В. Б. УТКИН, К. В. БАЛДИН, А. В. РУКОСУЕВ

МАТЕМАТИКА И ИНФОРМАТИКА

учебное пособие

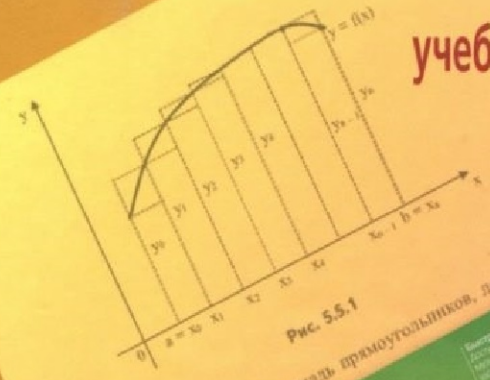
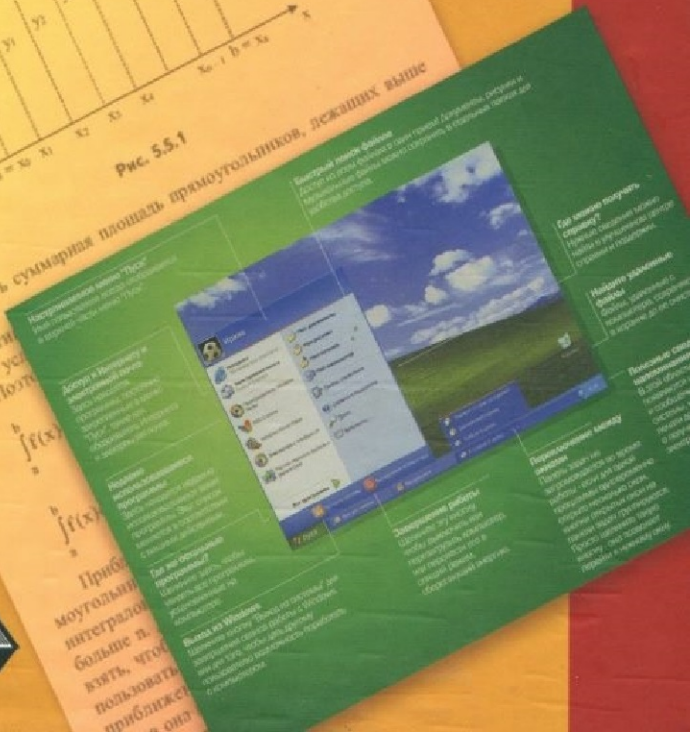


Рис. 5.5.1



«Дашков и К°»
Издательско-торговая корпорация

Издательско-торговая корпорация «Дашков и К°»

В. Б. Уткин, К. В. Балдин, А. В. Рукосуев

МАТЕМАТИКА И ИНФОРМАТИКА

Учебное пособие

4-е издание

Под общей редакцией
профессора В. Б. Уткина

Рекомендовано уполномоченным учреждением
Министерства образования и науки РФ —
Государственным университетом управления
в качестве учебного пособия для студентов
экономических вузов, обучающихся
по направлению подготовки «Экономика»

Регистрационный номер рецензии 540 от 29.12.2008 г.
(Федеральный институт развития образования)

Москва, 2016

УДК 517; 681
ББК 22.16; 32.96
У84

Рецензенты:

Кафедра “Проектирование вычислительных комплексов” “МАТИ” — РГТУ им. К. Э. Циолковского (заведующий кафедрой доктор физико-математических наук, профессор В. А. Зотов);

В. И. Бусов — доктор экономических наук, профессор.

Уткин В. Б.

У84 Математика и информатика: Учебное пособие / В. Б. Уткин, К. В. Балдин, А. В. Рукосуев. — 4-е изд. — М.: Издательско-торговая корпорация «Дашков и К°», 2016. — 468 с.

ISBN 978-5-394-01925-8

Учебное пособие содержит основы высшей математики и информатики. В него включены прикладные наработки авторов, примеры использования классических методов и заданий для самостоятельной работы обучаемых, вопросы для самопроверки. Материал учебника может послужить базой применения формальных методов для решения практических задач.

Для студентов высших учебных заведений, обучающихся по направлениям подготовки «Экономика» и «Менеджмент».

Подписано в печать 10.06.2015. Формат 60×84 1/16.
Печать офсетная. Бумага газетная.
Печ. л. 29,25. Тираж 500 экз.

Издательско-торговая корпорация «Дашков и К°»
129347, Москва, Ярославское шоссе, д. 142, к. 732.
E-mail: sales@dashkov.ru — отдел продаж;
office@dashkov.ru — офис;
http://www.dashkov.ru

Отпечатано в ГУП Академиздатцентр «Наука» РАН,
ОП «Производственно-издательский комбинат «ВИНИТИ»-«Наука»,
140014, Московская обл., г. Люберцы, Октябрьский пр-т, д. 403.

ISBN 978-5-394-01925-8

© Колл. авторов, 2006

Содержание

Введение

Настоящее учебное пособие содержит материал по двум различным, хотя и тесно связанным научным дисциплинам — математике и информатике. Математика — одна из древнейших наук — имеет широчайшее применение во всех сферах деятельности людей. Можно сказать, что без достижений математики не было бы современной цивилизации. В свою очередь, информатика сегодня — едва ли не самая развивающаяся наука, находящаяся на передовых рубежах общественного прогресса.

Название "математика" происходит от греческого слова "матема" (μα θη μα) — знание, наука. Математика относится к числу наиболее старых наук. В Вавилоне и Египте во втором тысячелетии до н. э. были известны многие сведения из арифметики и геометрии.

Академик А. Н. Колмогоров выделил четыре основных периода развития математики:

1. Период зарождения математики, который продолжался до VI–V вв. до н. э. Были известны разрозненные факты и формулы, которые использовались для решения сугубо практических задач: составление календарей, обмер земельных участков и т. д.

2. Период элементарной математики с VI–V вв. до н. э. до XVI в. н. э. Были заложены начала дедуктивного, аксиоматического методов. Развитие дедуктивной теории связано с именем Аристотеля, а первое систематизированное изложение геометрии было сделано Евклидом. Начало современной алгебры было положено в трудах итальянского ученого эпохи Возрождения Леонардо Пизанского (Фибоначчи).

3. Период создания математики переменных величин включает период с XVII в. по середину XIX в., который характеризуется созданием аналитической геометрии Р. Декартом, дифференциального и интегрального исчислений в работах И. Ньютона

и Г. Лейбница, а также современной алгебраической символики француза Вьета.

4. Современный период развития математики с середины XIX в. по наше время. Была создана теория действительных чисел, которая позволила строго выстроить математический анализ. В конце XIX вв. в работах Г. Кантора появилась теория множеств. В XIX и XX в. были заложены основы математической логики. В XX в. под влиянием успехов абстрактной алгебры появилось понимание математической структуры. Построению и исследованию математических структур были посвящены работы группы французских математиков, которые писали под псевдонимом Н. Бурбаки. Бурно развивались в XX в. теория вероятностей, математическая статистика, теория случайных функций. Здесь велик вклад российских и советских математиков П. Л. Чебышева, А. А. Маркова, А. Н. Колмогорова, Е. С. Вентцель.

В середине XIX в. Ф. Энгельс в своей работе "Анти-Дюринг" дал определение предмета математики: "Чистая математика имеет своим объектом пространственные формы и количественные отношения действительного мира" [30]. Это определение отражает развитие математики от ее рождения до середины XIX в.

Второе определение предмета математики было дано Н. Бурбаки в первой половине XX в. Оно было обусловлено современным периодом развития математики и новым подходом к аксиоматическому методу. По Н. Бурбаки, математика "скопление математических структур, не имеющих к действительности никакого отношения" [8].

Н. Бурбаки выделяет три основных типа структур: алгебраические, порядка, топологические. Многие ученые считали, что определение Ф. Энгельса устарело, но подход Н. Бурбаки встретил и негативное отношение, так как они не выяснили отношения рассматриваемых ими структур к действительному миру. Определение Ф. Энгельса не надо отбрасывать, его необходимо дополнить. Современное определение можно сформулировать, например, так [18]: математика — наука, которая исследует пространственные формы, количественные отношения, аксиоматические

структуры и вопросы доказательства путем построения абстрактных моделей действительного мира.

Математика проникла практически во все области общественной деятельности. Это объясняется, во-первых, тем, что она способна создавать модели изучаемых явлений, а во-вторых, тем, что математика используется для обработки числовых данных (как средство расчета).

В настоящее время различные численные и аналитические методы используются не только в естественных, но и в гуманитарных науках, например, в социологии, лингвистике, юриспруденции, экономике.

С помощью математических методов можно более глубоко анализировать сложные экономические явления и процессы, а с другой стороны, проблемы экономики стимулируют разработку новых математических теорий. Например, необходимость решения задач экономического планирования привела к разработке теории линейного программирования в 30-х годах XX в. [19]. Можно сделать вывод о том, что глубокое понимание экономических процессов и управление этими процессами невозможны без знания современного математического аппарата. Математическая подготовка современного специалиста в области экономики имеет свои специфические особенности, связанные со сложностью проведения финансово-экономических операций и принятия рациональных управленческих решений по ним.

Как наука математика имеет определенное математическое мировоззрение, однако для специалистов в области экономики, менеджмента, психологии и юриспруденции математика является прежде всего мощным инструментарием при проведении необходимых расчетов и исследований, а также фундаментом, на котором строится современное здание высшего профессионального образования.

Современный этап развития человеческой цивилизации характеризуется переходом к так называемому информационному обществу, в котором в результате процессов информатизации и компьютеризации информационные технологии во всех сферах деятельности играют более важную роль, нежели индустриальные,

аграрные и др. Как отмечал академик А. П. Ершов, информатизация — всеобщий неизбежный период развития цивилизации, период освоения информационной картины мира, осознания единства законов функционирования информации в природе и обществе, практического их применения, создания индустрии производства и обработки информации.

В связи с этим решение проблем рационального использования современных и перспективных методов и средств обработки информации в практической (профессиональной) деятельности людей приобретают первостепенное значение. Это обусловлено рядом причин. Во-первых, таковы актуальные потребности общества, связанные с необходимостью решения все более усложняющихся политических, экономических, военных и других проблем различного масштаба (глобальных, региональных, государственных, национальных и т. п.). Во-вторых, это — единственный путь значительного (а в ряде случаев — кардинального) повышения эффективности профессиональной деятельности человека. В-третьих, широкое распространение получили технические и программные средства, позволяющие реализовать новые технологии при приемлемом расходовании ресурсов. Наконец, пользователями этих технологий становится все большее число людей (по некоторым оценкам, к пользователям компьютерных технологий во многих странах может быть отнесено все трудоспособное население).

Естественно, что такой сложный и многообразный процесс, как информатизация, нуждается в методологическом обосновании, являющемся результатом исследований в рамках научно-технического направления и науки, получивших название “информатика”.

В широком смысле под информатикой понимается научно-техническое направление, охватывающее все аспекты разработки, проектирования, создания и функционирования систем обработки информации на базе ЭВМ, их применения и воздействия на различные области социальной практики.

Под информатикой в узком смысле понимается научная дисциплина, изучающая цели, способы и средства автоматизации человеческой деятельности на базе современных средств ЭВТ и свя-

зи при решении практических задач, связанных с накоплением, передачей, обработкой и представлением информации.

Предметом изучения информатики являются информационные технологии, которые реализуются на практике в автоматизированных информационных системах (АИС) различного назначения, выступающих в качестве объекта информатики. Таким образом, АИС позволяют автоматизировать ту или иную сферу профессиональной деятельности людей за счет использования компьютерных средств и технологий. Иными словами, в качестве основных средств (инструментария) автоматизации профессиональной деятельности людей сегодня выступают средства ЭВТ и связи.

В учебном пособии рассматриваются вопросы автоматизации таких важных видов профессиональной деятельности экономиста, как управленческая и научно-исследовательская. Возможность и необходимость применения именно в этих областях совершенных технических и программных средств, реализующих современные и перспективные математические методы, в том числе с использованием достижений теории искусственного интеллекта, позволяет в ряде случаев говорить об интеллектуализации профессиональной деятельности как важнейшем этапе ее автоматизации.

Содержание учебного пособия составляют два раздела — "Основы дискретной и высшей математики" и "Основы информатики".

Первый раздел состоит из семи глав. В первой главе "Основы дискретной математики" представлены основы теории множеств, введены элементы комбинаторики и основы теории графов. Вторая глава "Элементы линейной и векторной алгебры" посвящена матрицам, векторам, определителям и их свойствам, а также действиям над ними. Приведены методы решения систем линейных алгебраических уравнений (СЛАУ). В третьей главе "Функции и пределы" дано определение функции, способы ее задания и основные свойства, а также числовой последовательности и предела. Рассмотрены признаки существования предела, первый и второй замечательные пределы. Четвертая глава "Основы дифференциального исчисления" содержит сведения о производных, приложениях дифференциального исчисления и методике исследования

функций с помощью производных первого и второго порядков. В пятой главе "Основы интегрального исчисления" раскрыто содержание интегрального исчисления, приведены определения и свойства неопределенного, определенного и несобственного интегралов, а также способы их вычисления. Рассматриваются приложения интегрального исчисления. Шестая глава "Дифференциальные уравнения" написана на основе знаний, изложенных в предыдущих главах. В ней представлены обыкновенные дифференциальные уравнения первого порядка основных типов, а также методы их решения. Седьмая глава "Ряды" посвящена исследованию числовых, функциональных и степенных рядов.

Второй раздел "Основы информатики" содержит три главы. В восьмой главе "Методологические основы информатики" изложены основы создания и применения современных компьютерных систем и технологий в экономической практике: основные определения, классификация АИС, требования к специальному программному обеспечению и принципы его разработки, методика проведения информационного обследования объекта автоматизации, основы информационной безопасности экономических систем, современные технологии разработки АИС (в частности, CASE-технологии).

Девятая глава "Централизованная и распределенная обработка данных" посвящена вопросам автоматизации информационного обеспечения деятельности должностных лиц и содержит методические основы проектирования и использования баз (банков) данных и современных компьютерных сетей, а также организации процессов обработки информации в базе данных.

Десятая глава "Основы построения и использования систем искусственного интеллекта" содержит методические основы применения в деятельности специалистов систем искусственного интеллекта, прежде всего — экспертных систем с различными методами представления знаний. Подробно рассмотрены механизмы логического вывода в продукционных экспертных системах и диагностических экспертных системах байесовского типа.

Представленный в пособии материал по математике и информатике охватывает большинство разделов (не включены материалы

по теории вероятностей и математической статистике), изучаемых студентами гуманитарных вузов. При написании книги авторы придерживались современных точек зрения на понятия, о которых идет речь, и не отступали от общепринятых взглядов, стремились изложить материал в доступной для студентов форме. После каждого пункта помещены вопросы для самопроверки уровня усвоения читателем соответствующих знаний. Авторы надеются, что учебное пособие будет полезно студентам, изучающим психологию, менеджмент и юриспруденцию. Однако авторы издания не претендуют на исчерпывающую полноту и глубину охвата учебного материала из-за ограничений на объем книги.

Вклад авторов в данное издание распределился следующим образом: В. Б. Уткин — Введение, раздел II; К. В. Балдин — главы 4–6, подп. 8.1, 8.2, 9.1, 9.3, 10.1; А. В. Рукосуев — главы 1–3, 7.

Раздел I
ОСНОВЫ ДИСКРЕТНОЙ И ВЫСШЕЙ МАТЕМАТИКИ

1. Основы дискретной математики

1.1. Основы теории множеств

Понятие множества не определяется через другие понятия математики, т. е. оно является первичным. Появилось оно в конце XIX века в работах Г. Кантора (о сравнении мощностей множеств), который определил множество как "объединение в одно целое объектов, хорошо различимых нашей интуицией или нашей мыслью" [5, 8]. Разумеется, это определение не может рассматриваться как строго математическое, которого, впрочем, не существует, так как понятие множества является исходным, на его основе строятся остальные понятия математики.

Множество состоит из каких-то объектов. Например, существует множество натуральных чисел (N), множество всех звезд нашей Галактики, множество всех жителей РФ и т. д. Объекты, входящие в данное конкретное множество, являются его элементами. Различают конечные (состоящие из конечного числа элементов) и бесконечные множества.

Множества будем обозначать заглавными буквами A, B, C, ..., X, Y, Z, а их элементы — малыми буквами a, b, c, ..., x, y, z. Тот факт, что элемент x принадлежит множеству X, обозначают так: $x \in X$, а не принадлежит — $x \notin X$.

Если все элементы множества X являются также элементами множества Y, то множество X есть подмножество множества Y. Это записывается следующим образом: $X \subset Y$ или $Y \supset X$.

Множество всех подмножеств множества Y называется степенью этого множества и обозначается 2^Y или P(Y).

Множество X и Y являются равными (состоят из одних и тех же элементов) X = Y, если $X \subset Y$ или $Y \supset X$. Может использоваться следующая запись: $X \subseteq Y$, т. е. либо X = Y, либо $X \subset Y$ (является собственным подмножеством множества Y).

Вводится понятие пустого множества (∅), которое не содержит ни одного элемента. Например, множество решений уравнения $x^2 + 4 = 0$ есть пустое множество.

Способы задания множеств [5, 8]

1. Словесное описание.

Например, множество X есть множество всех прямых, проходящих через точку A плоскости α.

2. Перечисление элементов, входящих в множество.

Например, X = {–7, 0, 12, 123, 700}. Элементы в приведенном списке могут располагаться в любом порядке и должны быть различны, т. е. множества X = {5, 5, 7} и Y = {5, 7} равны между собой. Если во множестве есть совпадающие элементы, то его называют семейством Z = (5, 9, 9, 12, 12, 23).

3. Описание свойств элементов, входящих в множество.

X = {x | (x – 3)(x – 5) > 0}, т. е. элементами множества X будут только те числа, которые удовлетворяют неравенству (x – 3)(x – 5) > 0

Если обозначить через Q(x) свойства элементов, входящих в множество X, то для задания этого множества в общем случае можно использовать следующую запись: X = {x | Q(x)}, т. е. множество X состоит из тех элементов x, которые удовлетворяют свойству Q(x). Множество, которое содержит все рассматриваемые в некоторой задаче множества, называется универсальным и обозначается U.

Например, в качестве U можно взять множество N (заметим, что в некоторых монографиях оно начинается не с 1, а с 0).

Z = {z ∈ N | z < 6}, т. е. Z = {1, 2, 3, 4, 5}.

Для сокращения записи в математике используют кванторы всеобщности, существования, существования и единственности [17]:

∀ — квантор всеобщности (перевернутая первая буква английского слова All);

∃ — квантор существования (перевернутая первая буква английского слова Exists);

$\exists!$ — квантор существования и единственности.

Например, запись $(\forall x \in X)\ P(x)$ означает: для всех x из множества X справедливо P(x); запись $(\exists y \in Y)\ R(y)$ — существует y из множества Y такое, что справедливо R(y); запись $(\exists! z \in Z)\ M(z)$ — существует единственное z из множества Z такое, что справедливо M(z).

Операции над множествами [5, 26]

Пусть задано универсальное множество U. Множество всех его подмножеств есть 2^U. Заданы также множества X и Y, причем $X \in 2^U$ и $Y \in 2^U$.

Дополнением множества X называется множество $\overline{X}$ элементов множества U, которые не принадлежат X:

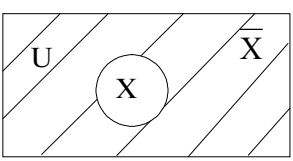
$$\overline{X} = \{x \in U \mid x \notin X\}.$$

Графически операции над множествами можно изображать с помощью кругов Эйлера (диаграмм Венна):

U — изображается прямоугольником;

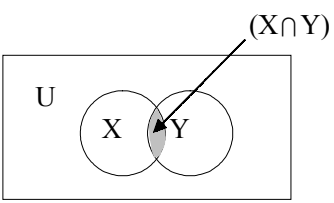
X — круг;

$\overline{X}$ — заштрихованная область прямоугольника.



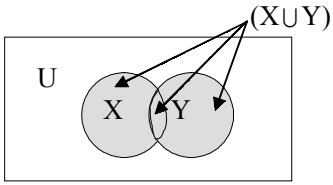
Пересечение $(X \cap Y)$ двух множеств X и Y состоит из элементов, принадлежащих обоим этим множествам:

$$(X \cap Y) = \{x \mid x \in X \text{ и } x \in Y\}$$



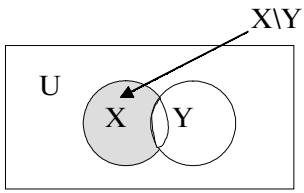
Объединение $(X \cup Y)$ двух множеств X и Y состоит из элементов, которые принадлежат хотя бы одному из множеств X и Y:

$$(X \cup Y) = \{x \mid x \in X \text{ или } x \in Y\}$$



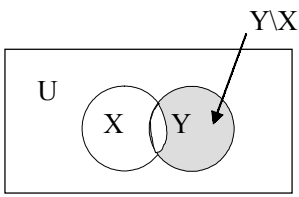
Разность (X/Y) двух множеств X и Y состоит из элементов, принадлежащих X, но не принадлежащих Y:

$X\backslash Y = \{x \mid x \in X \text{ и } x \notin Y\}$



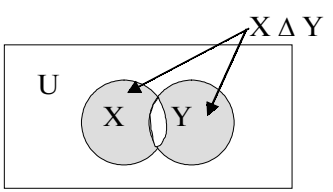
Аналогично определяется разность (Y\X) множеств Y и X:

$Y\backslash X = \{y \mid y \in Y \text{ и } y \notin X\}$



Симметрическая разность (X Δ Y) множеств X и Y состоит из элементов, которые принадлежат ровно одному из множеств X и Y:

$X \Delta Y = (X\backslash Y) \cup (Y\backslash X) = (X \cup Y)\backslash(X \cap Y)$



Пример 1.1.1

Дано множество: X = {–5, 0, 3, 17, 28, 33, 100}.

Y = {–7, 0, 5, 17, 33, 108}.

$X \cap Y = \{0, 17, 33\}$.

$X \cup Y = \{-7, -5, 0, 3, 5, 17, 28, 33, 100, 108\}$.

X/Y = {–5, 3, 28, 100}.

Y/X = {–7, 5, 108}.

$X \Delta Y = \{-7, -5, 3, 5, 28, 100, 108\}$.

Мощность множеств

Число элементов в конечном множестве X называют его мощностью и обозначают |X| или #X.

Например, X = {5, 12, 23, 111}, |X| = 4.

Если известны мощности множеств X и Y, то можно найти мощность их объединения по формуле

$|X \cup Y| = |X| + |Y| - |X \cap Y|$.

В общем случае имеем [5]:

$$|X_1 \cup X_2 \cup \ldots \cup X_n| = \sum_i |X_i| - \sum_{i<j} |X_i \cap X_j| + \sum_{i<j<k} |X_i \cap X_j \cap X_k| .$$

Для подсчета элементов в конечных множествах можно использовать комбинаторику.

Если между двумя множествами можно установить взаимно однозначное соответствие, то в них одинаковое количество элементов.

Взаимная однозначность означает, что каждому элементу первого множества соответствует один элемент второго, и наоборот.

Рассмотрим пример осуществления этого принципа.

Каких подмножеств больше у 100-элементного множества: мощности 60 или мощности 40.

Используем понятие числа сочетаний из n элементов по k (они отличаются только составом элементов) (более подробно в п. 1.2). Число сочетаний обозначается C_n^k и находится по формуле

$$C_n^k = \frac{n!}{k!(n-k)!},$$

где n! = 1 × 2 × 3 × ... × n (читается n-факториал).

В нашем случае имеем:

$$C_{100}^{60} = \frac{100!}{60!40!};\ C_{100}^{40} = \frac{100!}{40!60!}.$$

Поэтому у 100-элементного множества одинаковое количество подмножеств мощности 60 и 40 элементов.

Два множества называются равномощными, если между ними можно установить взаимно однозначное соответствие.

Для конечных множеств это означает, что в них одинаковое количество элементов.

Но это определение годится и для бесконечных множеств. Например, отрезки [0, 1] и [0, 10] равномощны, так как отображение $x \to 10x$ дает нужное соответствие [5, 8].

Можно также доказать, что интервал (0, 1) и луч $(0, +\infty)$ равномощны. Искомое взаимно однозначное соответствие имеет вид $x \to \frac{1}{x} - 1$ [5].

Также доказывается, что множество бесконечных последовательностей цифр 0, 1, 2, 3 равномощно множеству бесконечных последовательностей цифр 0 и 1.

Тот факт, что множество X равномощно (эквивалентно) множеству Y, записывается так: $|X| = |Y|$ $(X \sim Y)$.

Множество называется счетным, если оно равномощно множеству натуральных чисел (N).

Например, множество целых чисел (Z) равномощно N, т. е. $Z \sim N$.

Доказывается ([5]):

1) подмножество счетного множества конечно или счетно;

2) всякое бесконечное множество содержит счетное подмножество;

3) объединение конечного или счетного числа конечных или счетных множеств конечно или счетно.

Множество действительных чисел (R) или множество бесконечных последовательностей 0 и 1 несчетно. Мощность множества R больше мощности любого счетного множества и называется мощностью континуума.

Приведем без доказательства ***теорему Кантора-Бернштейна***.

Если множество X равномощно какому-то подмножеству множества Y, а множество Y равномощно какому-то подмножеству множества X, то множества X и Y *равномощны*.

Дадим также общую формулировку ***теоремы Кантора***.

Никакое множество Z не равномощно множеству всех своих подмножеств.

Наглядные представления о множествах могут приводить к противоречиям.

Приведем, например, ***парадокс Рассела*** [5].

Типичные множества не являются своими элементами. Например, множество целых чисел (Z) само не является целым числом и не будет своим элементом. Но, в принципе, можно представить себе множество, которое является своим элементом, например, множество всех множеств (U). Такие множества назовем "необычными". Теперь рассмотрим множество всех обычных множеств. Если оно обычное, то является своим элементом, и следовательно, оно — необычное, и наоборот.

В принципе, понятие множество не является непосредственно очевидным: разные люди (научные школы) могут понимать его по-разному.

Функции, прямые произведения, отношения

Сначала дадим традиционное определение функции [18].

Даны два множества X и Y. Функцией, которая определена на множестве X и принимает значение на множестве Y, называется закон (f), по которому каждому элементу x из множества X ($x \in X$) ставится в соответствие один элемент y из множества Y ($y \in Y$). Обычно это записывают в виде $y = f(x)$. Множество X есть область определения функции f, а множество $E = \{y \in Y \mid \exists x \in X\ y = f(x)\} \subseteq \subseteq Y$ — множество значений функции f. Функцию f, определенную на множестве X и принимающую значения на множестве Y, обозначают так $f : X \to Y$.

Элемент $x \in X$ называется независимой переменной (аргументом), элемент f(x) называется значением функции на элементе x.

Пусть задана однозначная функция f, т. е. различным значениям ее аргументов соответствуют различные значения функции, т. е. $(\forall x_1 \in X)\ (\forall x_2 \in X)\ (x_1 \neq x_2) \Leftrightarrow (f(x_1) \neq f(x_2))$. Знак "$\Leftrightarrow$" означает эквивалентность, например, $A \Leftrightarrow B$ — значит A эквивалентно B (A тогда и только тогда, когда B).

Тогда $\forall y \in E$ может быть поставлен в соответствие единственный элемент $x \in X$, такой что $y = f(x)$ и который обозначается $x = f^{-1}(y)$, это значит, что на множестве E определена функция f^{-1},

которая принимает значения на множестве X и называется обратной к функции f. Если задана функция f^{-1}: $E \to X$ и $Y = E$, т. е. когда множество значений функции f совпадает со всем множеством Y, функцию f называют взаимно однозначным соответствием между множествами X и Y.

Теперь сформулируем определение прямого, или декартова, произведения.

Прямым произведением множеств $X_1, X_2, X_3, ..., X_n$, $n \geq 2$ называется множество различных упорядоченных наборов (n-мерных векторов) $(x_1, x_2, x_3, ..., x_n)$, где $x_1 \in X_1$, $x_2 \in X_2$, $x_3 \in X_3$ ..., $x_n \in X_n$, обозначаемое $X_1 \times X_2 \times X_3 \times ... \times X_n = \prod_{i=1}^{n} X_i = \{(x_1, x_2, ..., x_n) | x_1 \in X_1, x_2 \in X_2, ..., x_n \in X_n\} = \{(x_1, x_2, ..., x_n) \mid x_i \in X_i, \forall i = \overline{1,n}\}$.

Заметим, что $X_1 \times X_2 \times ... \times X_n \neq X_n \times ... \times X_2 \times X_1$.

В том случае, если $X_i = X \; \forall i = \overline{1,n}$, то $X \times X \times ... \times X \equiv X^n$ ("≡" — тождественно равно) и называется n-й степенью множества X [14, 18].

В частном случае, если имеется два множества X и Y, их прямым произведением будет множество различных упорядоченных наборов (двумерных векторов) (x, y), где $x \in X$, а $y \in Y$, обозначаемые $X \times Y = \{(x, y) \mid x \in X, y \in Y\}$.

Например, имеем два множества $X = \{5, 6\}$, $Y = \{\ln 3, \ln 7\}$

$X \times Y = \{(5, \ln 3), (5, \ln 7), (6, \ln 3), (6, \ln 7)\}$

$Y \times X = \{(\ln 3, 5), (\ln 7, 5), (\ln 3, 6), (\ln 7, 6)\}$,

т. е. $X \times Y \neq Y \times X$.

Теперь дадим определение отношения. Даны множества $X_1, X_2, ..., X_n$, $n \geq 1$ n-местным отношением между элементами множеств $X_1, X_2, ..., X_n$ называется любое подмножество ρ прямого произведения этих множеств, т. е. $\rho \subseteq x_1 \times x_2 \times ... \times x_n$. Если $(x_1, x_2, ..., x_n) \in \rho$, говорят, что элементы $x_1, x_2, ..., x_n$ связаны отношением ρ.

Если $X_i = X \; \forall i = \overline{1,n} \; \rho \subseteq X^n$, говорят, что ρ есть n-местное отношение на множестве X [14, 18].

Для одноместных, двухместных и трехместных отношений часто используют специальные названия: унарные, бинарные, тернарные соответственно, т. е.

$n = 1$ — унарное отношение $\rho \subseteq X_1$

$n = 2$ — бинарное отношение $\rho \subseteq X_1 \times X_2$

$n = 3$ — тернарное отношение $\rho \subseteq X_1 \times X_2 \times X_3$

Если отношение совпадает с прямым произведением, оно называется полным, т. е. $\rho = X_1 \times X_2 \times ... \times X_n$.

Рассмотрим конкретные примеры отношений.

Пример 1.1.2

$x = \{1, 2, 7, 23, 35, 56\}$

$\rho = \{x \mid x \in X \text{ и } x < 23\}$

$\rho = \{1, 2, 7\}$, т. е. при $n = 1$ отношение ρ есть подмножество множества X.

Пример 1.1.3

$X_1 = \{0, 1, 2\}$; $X_2 = \{5, 6, 7\}$

$\rho = \{(x_1, x_2) \mid x_1 \in X_1, x_2 \in X_2, x_2 < 6\}$

$\rho = \{(0, 5), (1, 5), (2, 5)\}$.

Особый интерес представляют бинарные отношения, которые мы рассмотрим более подробно.

Если ρ есть отношение между элементами множеств X и Y, т. е. $\rho \subseteq X \times Y$, можно использовать запись $x \rho y$.

Обратным к отношению $\rho \subseteq X \times Y$ называется отношение

$\rho^{-1} = \{(y, x) \mid (x, y) \in \rho\} \subseteq Y \times X$.

Геометрически понятие бинарного отношения показано на рис. 1.1.1.

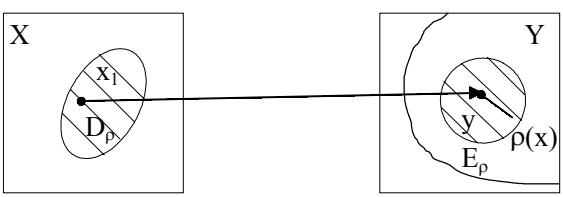


Рис. 1.1.1

$D_\rho = \{x \mid x \in X, (x, y) \in \rho\}$ — область определения отношения $\rho \subseteq X \times Y$.

$E_\rho = \{y \mid y \in Y, (x, y) \in \rho\}$ — множество значений отношений $\rho \subseteq X \times Y$.

Заметим, что для $\rho^{-1} \subset Y \times X$ D_ρ и E_ρ поменяются местами. Точка x — элемент множества D_ρ, а множество $\rho(x)$ есть ρ-образ

элемента x, а точка y является некоторым элементом из $\rho(x)$ (см. рис. 1.1.1).

Отсюда видно, что бинарное отношение является многозначной функцией [18].

Прямое произведение множества действительных чисел (R) само на себя $R \times R = R^2$ представляет собой систему координат на плоскости xOy.

А отношение $\rho \subseteq X \times Y \subseteq R^2$ является графиком отношения ρ.

Теперь рассмотрим конкретный пример бинарного отношения

Пример 1.1.4 [14]

Дано множество $X = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. Найдем отношение ρ, если оно задано так:

$\rho = \{(x_1, x_2) \mid x_1 \in X_1, x_2 \in X_2, x_1$ — делитель $x_2, x_1 \leq 5\}$

Отношение ρ имеет вид:

$\rho = \{(1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (1, 7), (1, 8), (1, 9), (1, 10), (2, 2), (2, 4), (2, 6), (2, 8), (2, 10), (3, 3), (3, 6), (3, 9), (4, 4), (4, 8), (5, 5), (5, 10)\}$.

Теперь найдем множество определения и множество значений отношения ρ.

$D_\rho = \{1, 2, 3, 4, 5\}$

$E_\rho = X$.

Представим полученное отношение ρ в графическом виде. Для этого зададим систему координат. Осью абсцисс будет D_ρ, а осью ординат E_ρ. По этим осям отложим элементы исходного множества X, затем соответствующие пары $(x_1\ x_2)$ отметим точками и получим графическое изображение нашего отношения (см. рис. 1.1.2).

Бинарные отношения можно представить в виде графа (множества вершин и множества ребер) (см. раздел “Элементы теории графов”).

Вершинами будут элементы исходного множества X, а ребрами пары $(x_1\ x_2)$. В данном случае важен порядок следования элементов, поэтому ребра будут ориентированными (обозначим их стрелками). В результате получим (см. рис. 1.1.3)

В случае $x_1 = x_2$ получим петлю.

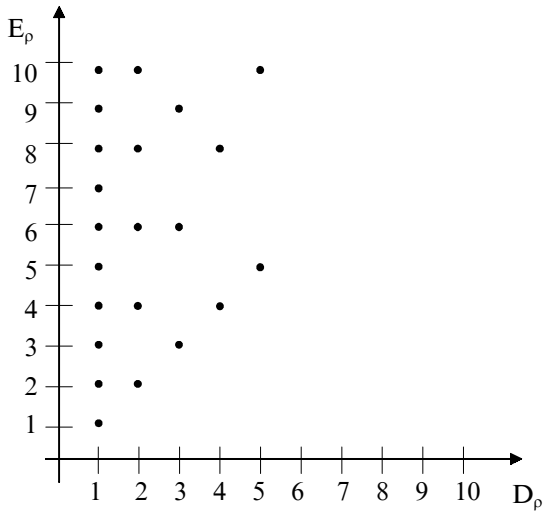


Рис. 1.1.2

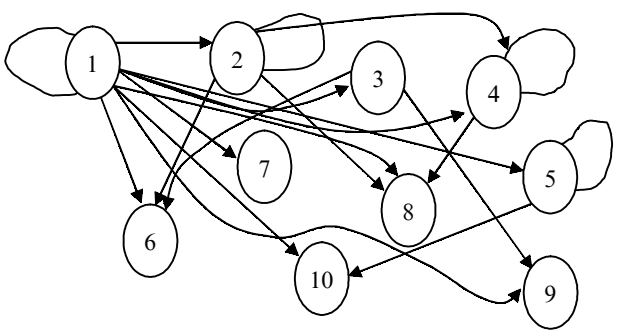


Рис. 1.1.3

Наконец, бинарное отношение можно представить в виде матрицы, т. е. прямоугольной таблицы размером $n \times m$, где n — количество строк, а m — количество столбцов. Например, если имеем два множества $X = \{x_1, x_2, ..., x_n\}$, $Y = \{y_1, y_2, ..., y_m\}$ и бинарное отношение на элементах этих множеств $\rho \subseteq X \times Y = \{(x, y) \mid x \in X, y \in Y\}$, то этому отношению соответствует матрица размера $n \times m$, состоящая из нулей и единиц. Единицами обозначаются пары

(x, y), входящие в отношение ρ (более подробно о матрицах см. в разделе “Элементы линейной алгебры”).

В данном примере мы имеем отношение $\rho \subset X^2$, т. е. ему соответствует матрица размера 10×10, число строк и столбцов которой равно числу элементов в множестве X.

	1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1	1
2	0	1	0	1	0	1	0	1	0	1
3	0	0	1	0	0	1	0	0	1	0
4	0	0	0	1	0	0	0	1	0	0
5	0	0	0	0	1	0	0	0	0	1
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0

В заключение рассмотрим функцию как отношение.

Отношение f между элементами множеств X и Y называется функцией, определенной на множестве X и принимающей значение на множестве Y, если

$$\forall x \in X \; \exists! \; y \in Y \mid (x, y) \in f \qquad (1.1.1)$$

В этом случае пишут $f: X \to Y$, а вместо $(x, y) \in f$ обычно пишется $y = f(x)$; y называют значением функции f на элементе x, так как для функциональных отношений в силу (1.1.1) f-образом одноэлементного множества $\{x\}$ будет одноэлементное множество $\{f(x)\}$.

Заметим, что (1.1.1) эквивалентно выполнению двух условий:

$$f^{-1}(Y) = X, \qquad (1.1.2)$$

$$y_1 = f(x) \text{ и } y_2 = f(x) \Rightarrow y_1 = y_2. \qquad (1.1.3)$$

Иногда отношения называют функцией, если выполнено только условие (1.1.3.), а если выполнено и условие (1.1.2), то *отображением*.

Часто слова функция и отображение используют как синонимы [18].

1.2. Элементы комбинаторики

Комбинаторика — часть математики, которая посвящена решению задач выбора и расположения элементов некоторого конечного множества в соответствии с заданными правилами, т. е. комбинаторика решает задачи выбора элементов из конечного множества и размещения этих элементов в каком-либо порядке [7, 8, 27].

Приведем правила сложения и умножения, которые применяются в комбинаторике [7].

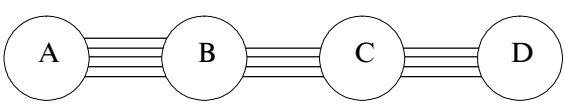
Правило сложения: если выбор каждого из объектов A_i $(i = \overline{1,k})$ можно сделать n_i способами, то выбор или A_1, или A_2, ..., или A_k можно произвести $n = \sum_{i=1}^{k} n_i$ способами.

Правило умножения: если выбор каждого из k объектов B_i $(i = \overline{1,k})$ можно сделать m_i способами, то выбор и B_1, и B_2, ..., и B_k можно осуществить $N = \prod_{i=1}^{k} m_i$ способами.

Приведем конкретные примеры применения этих правил.

Пример 1.2.1. Из Москвы в Санкт-Петербург можно добраться самолетом, поездом, автобусом. Есть пять автобусных маршрутов, два авиамаршрута, один железнодорожный. Поэтому общее число маршрутов между Москвой и Санкт-Петербургом равно: n = 5 + 2 + 1 = 8.

Пример 1.2.2. Из пункта A в пункт B можно доехать по 5 дорогам, из B в C можно доехать по трем дорогам, а из C в D можно доехать по четырем дорогам. Сколькими способами можно проехать из A в D через B и C?



По правилу произведения получаем N = 5 × 3 × 4 = 60.

Размещения

Дано множество, состоящее из n элементов. Размещением из n элементов по m элементам ($0 \le m \le n$) называется любое упорядоченное подмножество, содержащее m различных элементов исходного множества. Все эти подмножества отличаются друг от друга или составом элементов, или порядком их распределения [7, 8, 27].

Обозначим размещения из n элементов по m $A_n^m = n(n-1) \times$

$\times (n-2) \times ... \times (n-(m-1)) = \dfrac{n!}{(n-m)!}$, где $n! = 1 \times 2 \times 3 \times ... \times n$.

Принимается, что $0! = 1$ и $A_0^0 = 1$.

Пример 1.2.3. В футбольной премьер-лиге РФ участвуют 16 команд. Сколькими способами можно распределить три первых призовых места?

Так как в данном случае порядок команд имеет значение, то имеем дело с размещениями, т. е.

$$A_{16}^3 = \frac{16!}{(16-3)!} = \frac{1 \times 2 \times 3 \times ... \times 19}{1 \times 2 \times 3 \times ... \times 13} = 14 \times 15 \times 16 = 3360.$$

Число размещений по m элементов с повторениями из n элементов равно n^m, т. е.

$$(A_n^m)_{\text{повт}} = n^m.$$

Пример 1.2.4. Из цифр 5, 6, 7, 8 можно составить $(A_4^3)_{\text{повт}} = 4^3 = 64$ трехзначных числа.

Перестановки

Перестановкой из n элементов называется размещение из n элементов по n элементов [7, 8]. Так как каждая перестановка содержит все n элементов исходного множества, то различные перестановки отличаются друг от друга только порядком следования элементов, т. е.

$$P_n A_n^n = \frac{n!}{(n-n)!} = \frac{n!}{0!} = n!$$

Пример 1.2.5. Расставить четыре книги на полке можно $P_4 = 1 \times \times 2 \times 3 \times 4 = 24$ способами.

Если среди n элементов есть n_1 элементов одного вида, n_2 — элементов другого вида, ..., n_i — элементов i-го вида, то число перестановок с повторениями находится по формуле

$$P_n(n_1, n_2, ..., n_i) = \frac{n!}{n_1! n_2! ... n_i!}.$$

Пример 1.2.6. Сколько различных шестизначных чисел можно записать с помощью цифр: 2, 2, 3, 3, 4, 4. В данном случае $n = 6$, $n_1 = 2$, $n_2 = 2$, $n_3 = 2$, т. е.

$$P_6(2,2,2) = \frac{6!}{2!2!2!} = \frac{\cancel{1\times 2}\times 3\times \cancel{4}\times 5\times 6}{\cancel{1\times 2}\times \cancel{1\times 2}\times \cancel{1\times 2}} = 90.$$

Сочетания

Дано множество, состоящее из n элементов. Сочетанием из n элементов по m элементам ($0 \le m \le n$) называется любое подмножество, которое содержит m различных элементов исходного множества. Различными подмножествами считаются только те, которые отличаются по составу элементов [8, 27].

Обозначив число сочетаний через C_n^m, получим

$$C_n^m = \frac{A_n^m}{m!} = \frac{n!}{m!(n-m)!}.$$

Пример 1.2.7. В бригаде 25 человек. Надо найти четырех человек для работы в ночную смену. Сколькими способами это можно сделать?

Так как порядок выбранных четырех рабочих не имеет значения, то имеем $C_{25}^4 = \frac{25!}{4!(25-4)!} = \frac{25\times 24\times 23\times 22}{1\times 2\times 3\times 4} = 12\,650.$

Число сочетаний с повторениями из n элементов по m находится по формуле

$$(C_n^m)_{повт} = C_{n+m-1}^m = \frac{(n+m-1)!}{m!(n-1)!}.$$

Пример 1.2.8. Найдите число бросаний трех одинаковых кубиков. Число различных бросаний трех одинаковых кубиков равно

$$(C_6^3)_{повт} C_{6+3-1}^3 = C_8^3 = \frac{8!}{3!(8-3)!} = \frac{8\times 7\times 6}{1\times 2\times 3} = 56.$$

С числами C_n^m связано функциональное тождество, которое называется формулой *бинома Ньютона*

$$(a+b)^n = C_n^0 a^n + C_n^1 a^{n-1} b + \ldots + C_n^m a^{n-m} b^m + C_n^{m+1} a^{n-(m+1)} b^{m+1} + \\ + \ldots + C_n^n b^n,$$

при a = 1 имеем:

$$(1+b)^n = C_n^0 + C_n^1 b + \ldots + C_n^m b^m + \ldots + C_n^n b^n.$$

Пример 1.2.9. Разложить по формуле бинома Ньютона выражение $(1 + b)^7$.

$$(1+b)^7 = 1 + 7b + C_7^2 b^2 + C_7^3 b^3 + C_7^4 b^4 + C_7^5 b^5 + C_7^6 b^6 + C_7^7 b^7 = 1 + \\ + 7b + 21b^2 + 35b^3 + 35b^4 + 21b^5 + 7b^6 + b^7.$$

1.3. Основы теории графов

Впервые термин “граф” был употреблен венгерским математиком Д. Кенигом в 1936 г. Но начало теории графов было положено Л. Эйлером в 1736 г., когда он решил задачу о кенигсбергских мостах и нашел критерий существования в графе специального

маршрута (эйлерова цикла). Но как математическая дисциплина теория графов сформировалась именно в первой трети XX в. Эта теория располагает аппаратом решения различных прикладных задач из разных областей науки и техники, например, сетевое планирование и управление [10, 21]. В настоящее время теория графов — один из наиболее быстро развивающихся разделов математики.

Предположим, что V — это не пустое конечное множество, а $V^{(2)}$ — это множество всех его двухэлементных подмножеств. Множество E является произвольным подмножеством множества $V^{(2)}$, т. е. $E \subseteq V^{(2)}$.

Тогда графом (G) называется пара множеств (V, E), т. е. G = = (V, E), где VG — множество вершин графа, а EG — множество его ребер [10, 21, 25]. Любое ребро графа определяется парой его вершин. Если все пары вершин упорядоченные, то граф называется ориентированным (его ребра обозначают стрелками), в противном случае он — неориентированный. Если в графе есть ориентированные и неориентированные ребра, он называется смешанным. Ориентированный граф G можно задать как отношение, т. е. подмножество прямого произведения множества его вершин V само на себя

$G \subseteq V \times V$.

В этом случае множество всех двухэлементных подмножеств $V^{(2)}$ заменяется декартовым произведением $V \times V = V^2$ [10].

Графы обычно изображаются в виде рисунков, на которых вершины обозначаются кружками (точками), а ребра — отрезками (см. рис. 1.3.1, 1.3.2, 1.3.3).

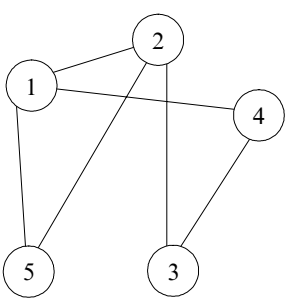


Рис. 1.3.1. Неориентированный граф

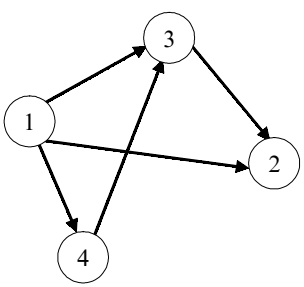


Рис. 1.3.2. Ориентированный граф

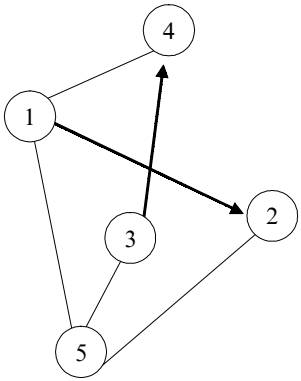


Рис. 1.3.3. Смешанный граф

Пример 1.3.1. Пусть задано множество V = {1, 2, 3}. Тогда $V^{(2)}$ = {(1, 2), (1, 3), (2, 3), (1, 1), (2, 2), (3, 3), (2, 1), (3, 1), (3, 2)}.

E = {(1, 2), (1, 3), (3, 2)}.

Предположив, что порядок вершин имеет значение, получаем следующий ориентированный граф (рис. 1.3.4).

Далее вершины графа мы будем обозначать буквами V_1, V_2, V_3, ..., V_n, а ребра e_1, e_2, ..., e_m. Вообще говоря, две вершины V_i и V_j определяют ребро e_k, т. е. $e_k = (V_i\ V_j)$ (рис. 1.3.5).

И в этом случае они будут концевыми вершинами ребра e_k. Но концевые вершины ребра не обязательно различны, т. е. начальная и конечная вершины могут совпадать. В этом случае ребро становится петлей (см. рис. 1.3.6).

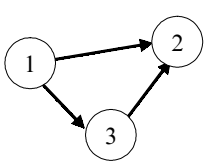


Рис. 1.3.4

Рис. 1.3.5

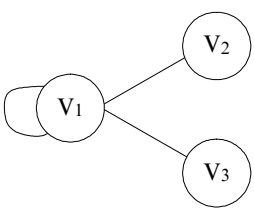


Рис. 1.3.6

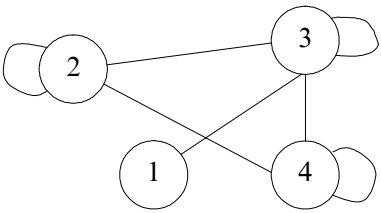


Рис. 1.3.7

Граф, имеющий петли, иногда называют *псевдографом* (см. рис. 1.3.7).

Между двумя вершинами может проходить и несколько ребер (ориентированных и неориентированных), их называют *параллельными*. А граф, имеющий такие ребра, называют *мультиграфом* (см. рис. 1.3.8).

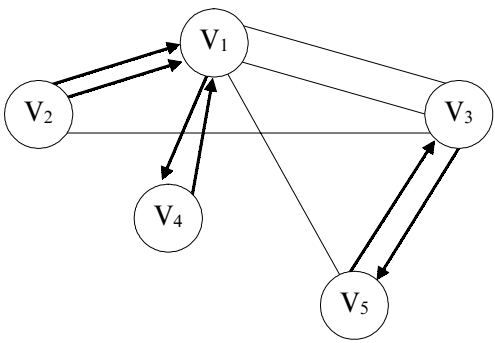


Рис. 1.3.8

Мультиграф — это пара (V, E), где V — множество вершин, а E — семейство подмножеств множества $V^{(2)}$. Употребление термина "семейство" говорит о том, что элементы множества $V^{(2)}$ могут повторяться (возможны параллельные ребра) [10].

Если граф не имеет петель и параллельных ребер, его называют простым (см. рис. 1.3.4). Граф G называется графом порядка n, если он содержит n вершин. На рисунке 1.3.9 имеем граф восьмого порядка [21].

Граф, который не имеет ребер (состоит только из вершин), называется пустым (см. рис. 1.3.10).

Граф, не имеющий вершин, называется ноль-графом. Две вершины называются смежными, если они являются концевыми вершинами какого-то ребра (например, вершины V_1 и V_3; V_2 и V_7 на рис. 1.3.9).

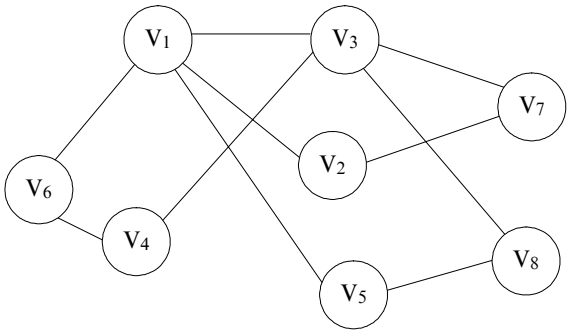


Рис. 1.3.9

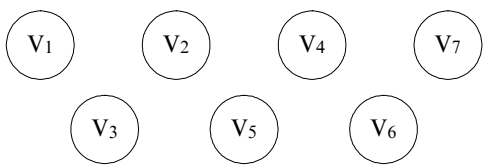


Рис. 1.3.10

Если два ребра имеют общую концевую вершину, то они являются смежными (например, ребра e_1 и e_2; e_2 и e_4 на рис. 1.3.11).

Если имеют в виду разные элементы графа (вершины и ребра), то используют понятия инцидентности, т. е. ребро инцидентно своим концевым вершинам (например, ребро e_3 инцидентно вершинам V_2 и V_3, см. рис. 1.3.11).

Число инцидентных вершине ребер называется степенью (валентностью) этой вершины и обозначается $d(V_i)$. Например, степень вершины V_2 на рис. 1.3.11 равна 3 ($d(V_2) = 3$) [10, 21, 27].

Вершина степени 1 называется висячей, вершина степени ноль — изолированной, а петля при вершине добавляет в степень этой вершины двойку.

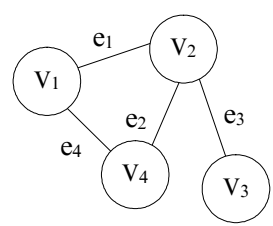


Рис. 1.3.11

Например, вершина V_4 на рис. 1.3.12 является висячей, вершина V_6 — изолированной, а степень вершины V_1 равна 4 ($d(V_1) = 4$ — два ребра и петля).

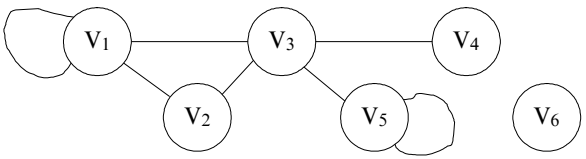


Рис. 1.3.12

Граф называют полным, если две любые его вершины смежные (см. рис. 1.3.4).

Приведем без доказательства две теоремы [10].

ТЕОРЕМА 1.3.1. *Сумма степеней вершин графа равна удвоенному числу его ребер.*

У графа на рис. 1.3.12 имеется 7 ребер, а сумма степеней его вершин равна 14.

ТЕОРЕМА 1.3.2. *Число вершин нечетной степени в любом графе четно.*

Например, у графа, изображенного на рис. 1.3.12, таких вершин две (V_4 и V_5).

Может оказаться, что один и тот же граф изображается разными рисунками. Говорят, что два графа G_1 и G_2 изоморфны, если существует такое взаимнооднозначное соответствие между множествами их вершин V_1 и V_2, что вершины соединены ребрами в одном из графов в том и только том случае, когда соответствующие им вершины соединены в другом графе. Если ребра ориентированы, то их направления также должны соответствовать друг другу.

На рис. 1.3.13 приведен пример изоморфных графов [10].

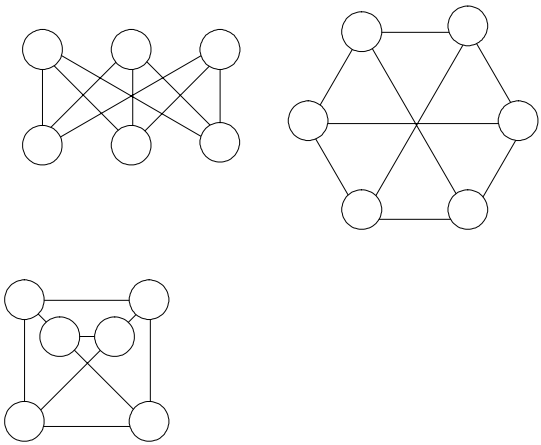


Рис. 1.3.13

В тех случаях, когда необходимо различать изоморфные графы, помечают их вершины и (или) ребра (см. рис. 1.3.14).

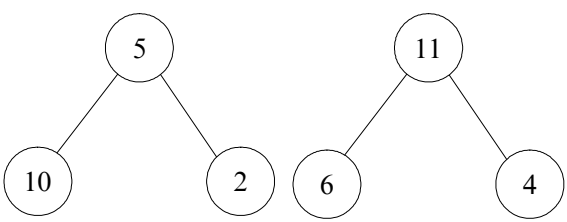


Рис. 1.3.14

Маршруты, цепи, пути, циклы [10, 21, 25]

Маршрут в графе — это конечная чередующаяся последовательность вершин и ребер, оканчивающаяся на вершине, причем одинаковые вершины и ребра в маршруте могут повторяться. Например, маршрут $V_1e_1\ V_2e_5\ V_4e_6\ V_1e_8\ V_6e_7\ V_4$ на рис. 1.3.15.

Маршрут называют открытым, если его конечные вершины различны, в противном случае он является замкнутым, например, маршрут $V_1e_1\ V_2e_5\ V_4e_6\ V_1e_8\ V_6e_7\ V_4e_6V_1$ на рис. 1.3.15.

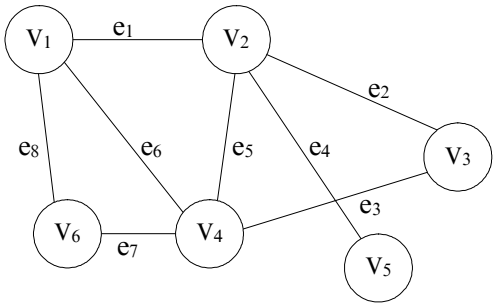


Рис. 1.3.15

Маршрут называют цепью, если все его ребра различны. Цепь является открытой, если ее конечные вершины различны, в противном случае — она замкнутая.

На рис. 1.3.16 $V_2e_1\ V_3e_3\ V_5e_{13}\ V_1e_5\ V_3e_8\ V_4$ — открытая цепь, а $V_1e_{13}\ V_5\ e_3V_3\ e_2V_2\ e_1V_3\ e_7V_1$ — замкнутая цепь.

Открытую цепь называют путем, если все ее вершины различны. Например, $V_4e_8\ V_3e_6\ V_1e_{11}\ V_6$ на рис. 1.3.16.

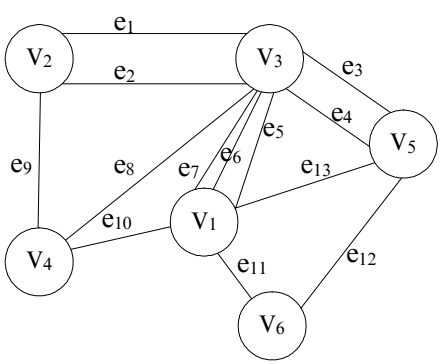


Рис. 1.3.16

Замкнутую цепь называют циклом, если все ее вершины за исключением концевых различны, например, $V_3e_3\ V_5e_{13}\ V_1e_7\ V_3$ на рис.1.3.16.

Две несовпадающие вершины V_i и V_j в графе G называются связными, если существует маршрут V_i — V_j.

Граф G называют связным, если две его любые несовпадающие вершины могут быть соединены маршрутом. Например, связными являются графы на рис. 1.3.15 и рис. 1.3.16, а несвязным — граф, изображенный на рис. 1.3.17.

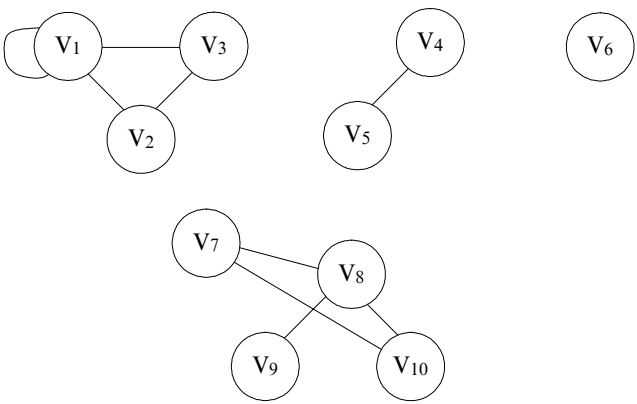


Рис. 1.3.17

Деревья и лес

Большую роль в различных отраслях науки имеют связные ациклические (не имеющие циклов) графы, которые на множестве V вершин имеют E = (V – 1) ребер, т. е. G = (V, (V – 1)). Эти графы носят название деревьев [10, 21]. Заметим, что (V – 1) — это минимальное количество ребер для того, чтобы граф был связным.

Примерами древовидной структуры являются генеалогический граф, схема вертикали управления любой организации, совокупность всех файлов, размещенных на диске ПЭВМ. Пример дерева приведен на рис. 1.3.19.

Несвязный граф, компонентами которого являются деревья, называется лесом (см. рис. 1.3.18).

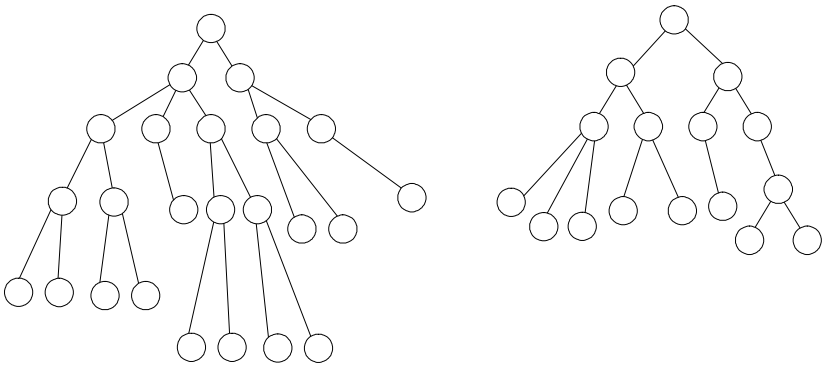


Рис. 1.3.18

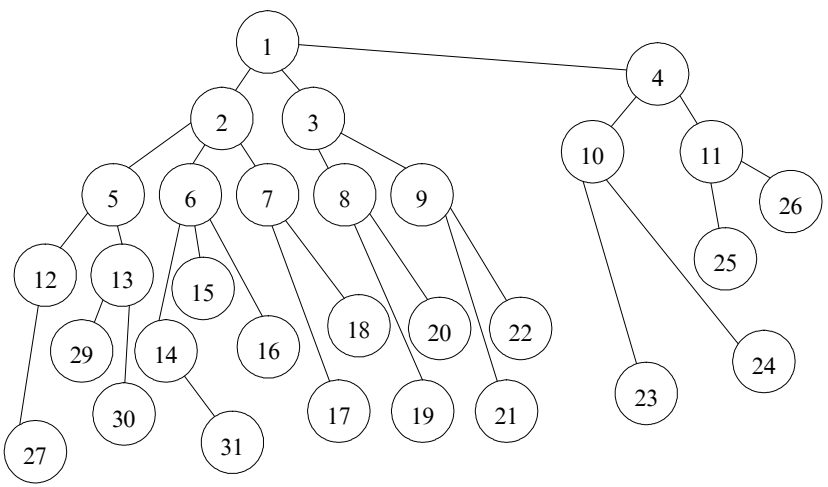


Рис. 1.3.19

Матрицы графов

1. Матрица инцидентности. Рассмотрим простой граф G (без петель и параллельных ребер), имеющий n вершин и m ребер. Ему соответствует матрица инцидентности размера n × m, т. е.

$A_I = [a_{ij}]$, где $i = \overline{1,n}$; $j = \overline{1,m}$.

Каждый элемент этой матрицы a_{ij} в случае ориентированного графа определяется следующим образом [10, 21]:

$$a_{ij} = \begin{cases} 1, \text{ если ребро j инцидентно вершине i и исходит из нее;} \\ -1, \text{ если ребро j инцидентно вершине i и входит в нее;} \\ 0, \text{ если ребро j не инцидентно вершине i.} \end{cases}$$

В случае неориентированного графа имеем:

$$a_{ij} = \begin{cases} 1, \text{ если ребро j инцидентно вершине i;} \\ \\ 0, \text{ если ребро j не инцидентно вершине i.} \end{cases}$$

Рассмотрим конкретный пример. Имеем ориентированный граф, имеющий 5 вершин и 7 ребер (см. рис. 1.3.20).

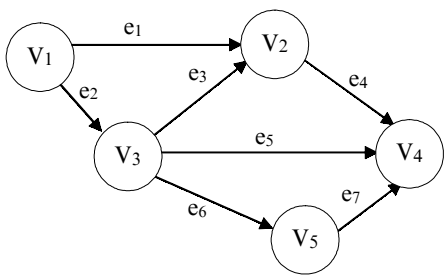


Рис. 1.3.20

Ему соответствует матрица инцидентности размера 5×7 следующего вида:

$$A_{1\ 5\times 7} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

В случае задания мультиграфа (имеет параллельные ребра) матрица инцидентности определяется по приведенным выше правилам.

Например, найдем матрицу инцидентности для графа, изображенного на рис. 1.3.21.

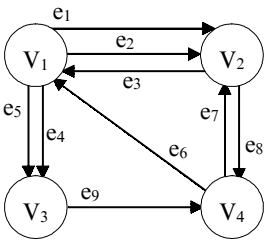


Рис. 1.3.21

Искомая матрица имеет размер (4 × 9) и выглядит следующим образом:

$$A_{1\ 4\times 9} = \begin{pmatrix} 1 & 1 & -1 & 1 & 1 & -1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \end{pmatrix}$$

2. Матрица смежности. Пусть задан псевдограф (имеет петли), содержащий n вершин и m ребер.

Матрицей смежности этого графа A_s называется матрица размера n × n, т. е.

$A_s = (a_{ij})$, $i = \overline{1,n}$, $j = \overline{1,n}$.

Любой элемент этой матрицы a_{ij} в случае ориентированного графа определяется следующим образом [10]:

$$a_{ij} = \begin{cases} 1, \text{если вершины } (V_iV_j) \in e_k \text{ и ребро исходят из вершины } V_i; \\ 0, \text{в противоположном случае.} \end{cases}$$

В случае, если граф G неориентированный, то любой элемент его матрицы смежности определяется так:

$$a_{ij} = \begin{cases} 1, \text{если вершины } V_i \text{ и } V_j \text{ пренадлежат ребру } e_k; \\ 0, \text{в противоположном случае.} \end{cases}$$

Матрица A_s в этом случае будет симметричной относительно главной диагонали.

Рассмотрим конкретный пример. Найдем матрицу смежности для графа, изображенного на рис. 1.3.22.

Искомая матрица смежности имеет размер 4 × 4 и выглядит так:

$$A_s = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

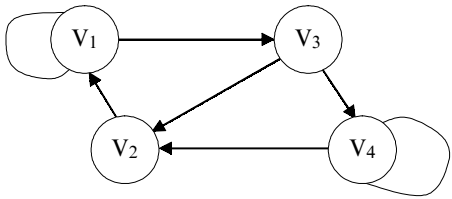


Рис. 1.3.22

В том случае, если граф кроме петель имеет параллельные ребра, матрица A_s находится по следующим правилам [7].

Если задан ориентированный граф, то каждый элемент его матрицы смежности находится так:

$$a_{ij} = \begin{cases} \Sigma \text{ числа ребер, выходящих из вершины } V_i \text{ и входящих в вершину } V_j; \\ 0, \text{ в противоположном случае.} \end{cases}$$

В случае неориентированного графа имеем:

$$a_{ij} = \begin{cases} \Sigma \text{ числа ребер между смежными вершинами} V_i \text{ и } V_j; \\ 0, \text{ в противоположном случае.} \end{cases}$$

Найдем матрицу смежности для графа, изображенного на рис 1.3.23.

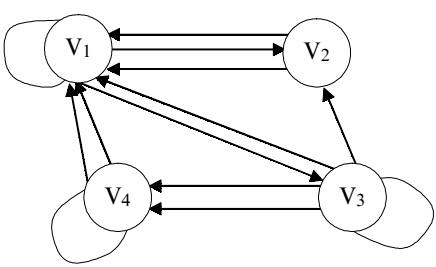


Рис. 1.3.23

Данному графу соответствует матрица смежности размера 4×4, имеющая вид:

$$A_s = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 2 & 0 & 0 & 0 \\ 1 & 1 & 1 & 2 \\ 2 & 0 & 0 & 1 \end{pmatrix}$$

Раскраски

Задачи раскраски вершин или ребер графа занимает важное место в теории графов, что связано с существованием объектов, которые по каким-то причинам могут быть объединены в одну группу.

Пусть $G = (V, E)$ — граф, $k \in N$. Произвольная функция вида $f : VG \to \{1, 2, 3, ..., k\}$ называется вершинной k-раскраской, или просто k-раскраской. Раскраска называется правильной, если для любых смежных вершин V_i и V_j выполняется неравенство $f(V_i) \neq f(V_j)$. Граф, для которого существует правильная k-раскраска, называется k-раскрашиваемым. В определении раскраски вместо множества $\{1, 2, 3, ..., k\}$ можно взять произвольное k-элементное множество. Правильную k-раскраску графа можно трактовать как окрашивание каждой его вершины в один из k цветов, при этом смежные вершины должны быть окрашены в разные цвета. При k-раскраске может быть использовано и менее k цветов. Правильную k-раскраску графа G можно рассматривать как разбиение $V_1 \cup V_2 \cup ... \cup V_t = VG$, $t \leq k$ множества вершин VG на не более чем k непустых классов, каждый из которых является независимым множеством. Классы этого разбиения называются цветными классами. Минимальное число k, при котором граф G является k-раскрашиваемым, называется хроматическим числом этого графа и обозначается X(G). Если $X(G) = k$, то граф G называется k-хроматическим. Правильная k-раскраска G при $k = X(G)$ называется минимальной [10, 27].

На рис. 1.3.24 приведена одна из правильных 4-раскрасок, причем меньшим числом цветов этот граф раскрасить нельзя [10].

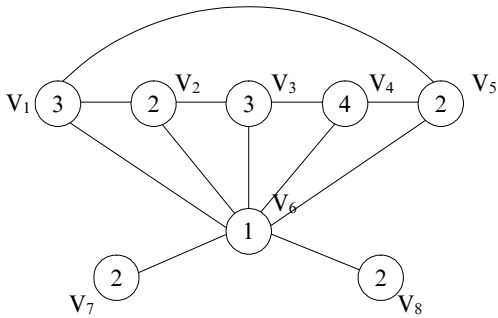


Рис. 1.3.24

Плоские и планарные графы

В некоторых случаях необходимо знать, можно ли изобразить граф на плоскости так, чтобы его изображение удовлетворяло некоторым условиям, например, в радиоэлектронике при изготовлении микросхем проводники электрического тока не должны пересекаться. Такая же задача возникает при проектировании железнодорожных трасс, когда нежелательны переезды. Поэтому вводится понятие плоского графа.

Плоским называют граф, вершины которого являются точками плоскости, а ребра непрерывными плоскими линиями без самопересечений, соединяющими соответствующие вершины так, что никакие два ребра не имеют общих точек, кроме инцидентной им обоим вершины [10].

Примеры плоских графов показаны на рис. 1.3.25.

Любой граф, изоморфный плоскому графу, называется *планарным* [10]. На рис. 1.3.26а приведен плоский граф, а на рис 1.3.26б — планарный граф, изоморфный графу на рис. 1.3.26а.

Проблема раскраски планарных графов является одной из самых знаменитых в этой теории. Первоначально вопрос формулировался так: достаточно ли четырех красок для такой раскраски произвольной географической карты, при которой любые две соседние страны окрашены в разные цвета. Рассматриваются только те карты, в которых граница любой страны состоит из одной

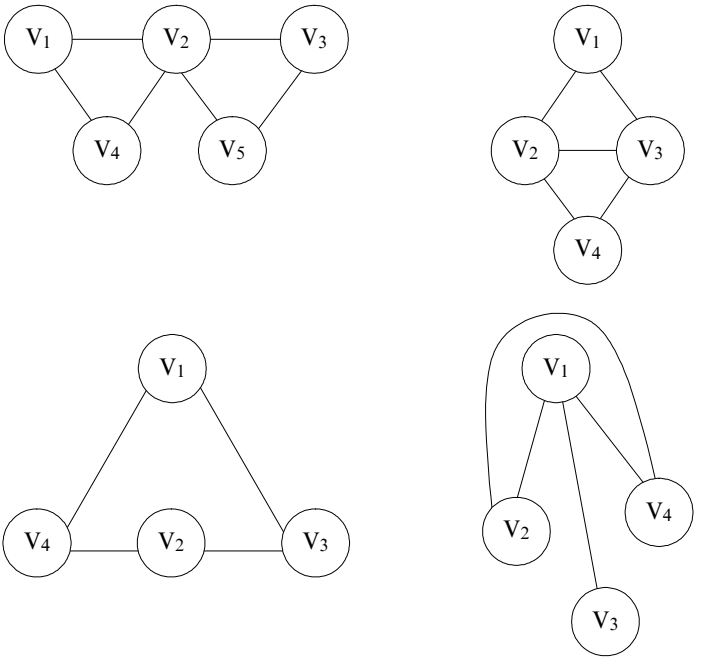


Рис. 1.3.25

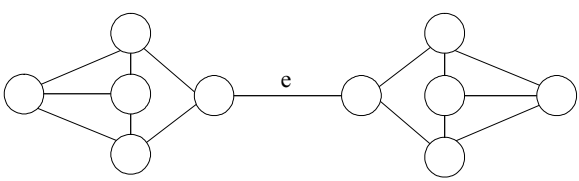


Рис. 1.3.26а

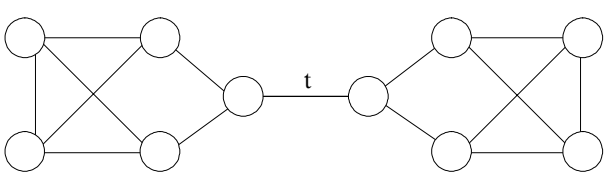


Рис. 1.3.26б

замкнутой линии, а соседними считаются страны, имеющие общую границу не нулевой длины.

Позднее понятие карты сформулировали так: карта — это связный плоский мультиграф без мостов (ребро графа называется мостом, если его удаление увеличивает число компонент графа, мостами являются ребра e и t в графах на рис. 1.3.26а и 1.3.26б).

В 1879 г. английский математик А. Кэли сформулировал гипотезу четырех красок так: всякая карта 4-раскрашиваема.

Часто пользуются другой формулировкой: всякий планарный граф 4-раскрашиваем.

В 1890 г. Р. Хивуд показал, что если 4 заменить на 5, то гипотеза легко доказывается, т. е. верна теорема: всякий планарный граф 5-раскрашиваем.

Эйлеровы цепи

Как мы уже упоминали, с задачи о кенигсбергских мостах началась теория графов. На рис. 1.3.27 показан план расположения семи мостов на реке Преголь в городе Кенигсберге (ныне Калининград). Задача состоит в том, чтобы пройти каждый мост по одному разу и вернуться в исходную точку [10, 21, 25].

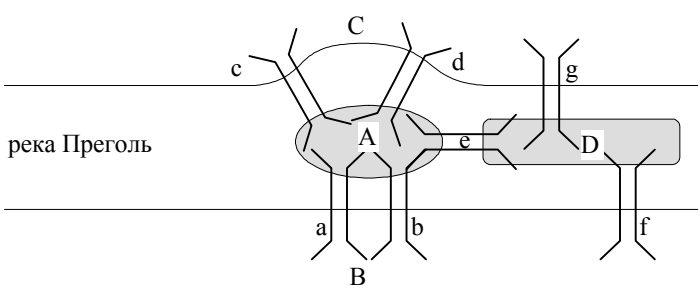


Рис. 1.3.27

Так как существенны только переходы через мосты, план города можно свести к изображению графа, в котором ребра соответствуют мостам, а вершины — различным разделенным мостами участкам города (см. рис. 1.3.28).

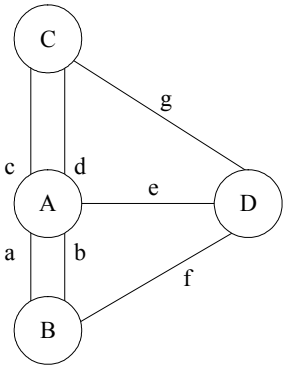


Рис. 1.3.28

Л. Эйлер обратился к общей задаче, касающейся теории графов: в каких случаях в конечном графе можно найти такой цикл, в котором каждое ребро графа участвовало бы один раз?

Если такой цикл (замкнутая цепь) есть, он называется Эйлеровым, а граф, содержащий его — Эйлеровым графом.

Л. Эйлер сформулировал и доказал теорему: *конечный граф G является Эйлеровым графом тогда и только тогда, когда: а) граф является связным; б) степени всех его вершин четные*.

Из теоремы Л. Эйлера следует, что задача о кенигсбергских мостах не имеет решения (граф на рис.1.3.28 не Эйлеров, так как степени всех его четырех вершин являются нечетными $d(A) = 5$; $d(B) = 3$; $d(C) = 3$; $d(D) = 3$).

Если обобщить задачу Л. Эйлера, то надо искать наименьшее число не пересекающихся по ребрам цепей N_i, которые необходимы для того, чтобы покрыть конечный связный граф G, т. е. включающий все его ребра в цепи N_i. Решение этой задачи сводится к задаче Л. Эйлера.

Задачи для самостоятельного решения

1. Дано: $X = \{-5, 17, 22, 34, 101\}$
$Y = \{-17, 0, 22, 34, 102, 505\}$.
Найти $X \cup Y$, $X \cap Y$, $X \setminus Y$, $Y \setminus X$, $X \Delta Y$.
2. Дано: $X = (-\infty;5]$; $Y = [-7;607)$.
Найти $X \cup Y$, $X \cap Y$, $X \setminus Y$, $Y \setminus X$.
3. Докажите, что отрезки [0, 1] и [0, 5] равномощны.
4. Дано: $X = \{1, 2, 3\}$; $Y = \{7, 8\}$
$\rho = \{(1, 7), (1, 8), (2, 8), (3, 7)\}$.
Построить матрицу и граф отношения ρ.
5. Сколькими способами в отделе, состоящем из 100 человек, можно выбрать начальника и его заместителей?

6. Есть шесть видов конвертов без марок. Сколькими способами можно выбрать конверт и марку для отправки письма?

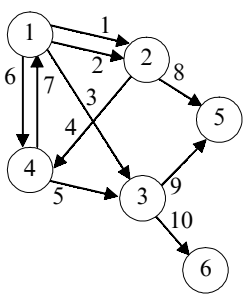
7. Из двенадцати человек надо выбрать пять и разместить их на занумерованных стульях (по одному человеку на стул). Сколькими способами это можно сделать?

8. Сколькими способами можно посадить за стол четырех мужчин и четырех женщин так, чтобы женщины и мужчины не сидели рядом?

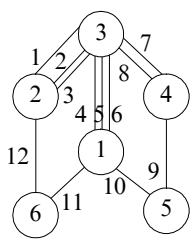
9. Сколько различных восьмизначных чисел можно составить, используя цифры 3, 4, 5?

10. Найти матрицы инцидентности для двух графов:

10.1

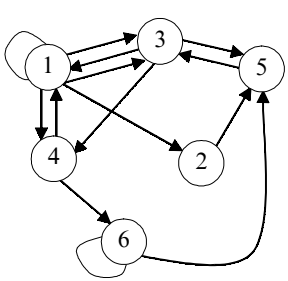


10.2

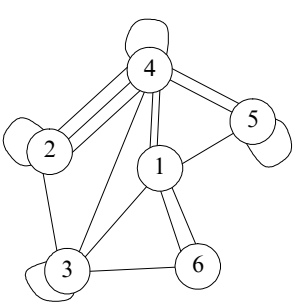


11. Найти матрицы смежности для графов:

11.1



11.2



Вопросы для самопроверки

1. Какие способы задания множеств вы знаете?
2. Какое множество называется универсальным?
3. Какое отношение называется бинарным?
4. Что называется числом сочетаний из n элементов по m?
5. Что называется числом размещений из n элементов по m?
6. Что называется перестановкой из n элементов?
7. Какие графы называются изоморфными?
8. Какие графы называются Эйлеровыми?
9. Что такое степень вершины графа?
10. Какие графы называют деревьями?
11. Что такое k-раскраски графа?
12. Какие графы называют планарными?

2. Элементы линейной и векторной алгебры

Линейная алгебра — это часть математики, посвященная в основном теории матриц и связанной с нею теории линейных преобразований, векторных пространств. Она включает теорию форм, теорию инвариантов, тензорную алгебру.

В данном пособии мы рассмотрим понятие матрицы, а также ее применение для решения систем линейных алгебраических уравнений (СЛАУ), рассмотрим определители, векторы, собственные числа и векторы матриц.

2.1. Матрицы, определители и их свойства

Матрицей называется прямоугольная таблица размером m (число строк) на n (число столбцов), заполненная некоторыми математическими объектами [28]. Мы будем рассматривать матрицы, элементами которых являются действительные числа.

Как правило, матрицы обозначают большими буквами (A, B...), а их элементы маленькими буквами с двумя индексами, указывающими номер строки и номер столбца (a_{ij}, b_{ij}, ...), т. е. прямоугольную матрицу размера m × n записывают следующим образом:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} =$$

$$= \begin{Vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{Vmatrix} = (a_{ij}) = [a_{ij}] = \|a_{ij}\|, \; i = \overline{1,m}, \; j = \overline{1,n}.$$

Если мы заменим строки матрицы ее столбцами (столбцы строками), то получим транспонированную матрицу, которую обозначают заглавной буквой с индексом Т наверху:

$$A^T = \begin{pmatrix} a_{11} & a_{21} & \dots & a_{m1} \\ a_{12} & a_{22} & \dots & a_{m2} \\ a_{1n} & a_{2n} & \dots & a_{mn} \end{pmatrix}$$

Рассмотрим некоторые типы матриц [19, 29].

Если число строк матрицы равно числу ее столбцов, то мы имеем *квадратную матрицу*:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}.$$

Элементы $a_{11}, a_{22}, \dots, a_{nn}$ называются *главной диагональю*, а их сумма — это след матрицы.

Элементы $a_{1n}, a_{2n-1}, \dots, a_{n1}$ составляют *побочную диагональ*.

Если все элементы матрицы, кроме элементов, стоящих на *главной диагонали*, равны нулю, то мы имеем *диагональную матрицу*:

$$D = \begin{pmatrix} d_{11} & 0 & \dots & 0 \\ 0 & d_{22} & \dots & 0 \\ 0 & 0 & \dots & d_{nn} \end{pmatrix}.$$

Если все ненулевые элементы диагональной матрицы равны 1, то мы имеем *единичную матрицу*:

$$E = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ 0 & 0 & \dots & 1 \end{pmatrix}.$$

Если ненулевые элементы располагаются выше главной диагонали, то имеем *верхнюю треугольную матрицу*, а если ниже — *нижнюю треугольную матрицу*:

$$B = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ 0 & b_{22} & \dots & b_{2n} \\ 0 & 0 & b_{33} & b_{3n} \\ 0 & 0 & 0 & b_{nn} \end{pmatrix}; \quad C = \begin{pmatrix} c_{11} & 0 & \dots & 0 \\ c_{21} & c_{22} & \dots & 0 \\ c_{n1} & c_{n2} & \dots & c_{nn} \end{pmatrix}.$$

Матрица размера $m \times 1$ — это матрица-столбец, а матрица размера $1 \times n$ — матрица-строка:

$$A = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{pmatrix}; \quad B = (b_1 b_2 \dots b_n).$$

Рассмотрим линейные операции над матрицами [13, 19, 29].

Для сложения двух матриц необходимо, чтобы они имели одинаковые размеры.

Сумму двух матриц обозначим $A + B$, а ее элементы равны $a_{ij} + b_{ij}$, т. е.

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} + \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ b_{m1} & b_{m2} & \dots & b_{mn} \end{pmatrix} =$$

$$= \begin{pmatrix} a_{11} + b_{11} & a_{12} + b_{12} & \dots & a_{1n} + b_{1n} \\ a_{21} + b_{21} & a_{22} + b_{22} & \dots & a_{2n} + b_{2n} \\ a_{m1} + b_{m1} & a_{m2} + b_{m2} & \dots & a_{mn} + b_{mn} \end{pmatrix}.$$

Например, $\begin{pmatrix} 2 & 3 \\ -1 & 6 \end{pmatrix} + \begin{pmatrix} 7 & 5 \\ 3 & -4 \end{pmatrix} = \begin{pmatrix} 9 & 8 \\ 2 & 2 \end{pmatrix}$.

Сложение матриц обладает следующими свойствами:

1. $A + B = B + A$.
2. $(A + B) + C = A + (B + C)$.

3. Для любых двух матриц одинакового размера всегда существует единственная матрица Z такая, что A + Z = B. Тогда Z есть разница матриц B и A, т. е. Z = B – A. Элементы матрицы Z равны $b_{ij} - a_{ij}$.

Произведением матрицы $A = (a_{ij})$ на число $k \in R$ называется матрица

$$kA = \begin{pmatrix} ka_{11} & ka_{12} & \dots & ka_{1n} \\ ka_{21} & ka_{22} & \dots & ka_{2n} \\ \hline ka_{m1} & ka_{m2} & \dots & ka_{mn} \end{pmatrix}.$$

Например, $5 \times \begin{pmatrix} 3 & -1 \\ 0 & 6 \end{pmatrix} = \begin{pmatrix} 15 & -5 \\ 0 & 30 \end{pmatrix}$.

Для умножения двух матриц необходимо, чтобы они были согласованными. Матрицы A и B называются *согласованными*, если число столбцов матрицы A равно числу строк матрицы B.

Пусть заданы матрицы:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \hline a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} = (a_{ij}), \text{ где } i = \overline{1,m},\ j = \overline{1,n};$$

$$B = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1p} \\ b_{21} & b_{22} & \dots & b_{2p} \\ \hline b_{n1} & b_{n2} & \dots & b_{np} \end{pmatrix} = (b_{jk}), \text{ где } j = \overline{1,n},\ k = \overline{1,p};$$

Тогда произведением матрицы A на матрицу B называется матрица C размера $m \times p$, элементы c_{ik} которой находятся по формуле

$$C_{ik} = \sum_{j=1}^{n} a_{ij} b_{jk}.$$

Из определения произведения матриц следует, что $A \times E = E \times A = A$.

Произведение матриц обладает следующими свойствами:

1. $(A \times B) \times C = A \times (B \times C)$.
2. $(A + B) \times C = AC + BC$.

В общем случае $A \times B \neq B \times A$.

Рассмотрим конкретный пример умножения двух матриц:

$$\begin{pmatrix} 3 & 1 & -2 \\ 4 & 0 & 5 \\ 7 & -6 & 9 \end{pmatrix} \times \begin{pmatrix} 4 & 5 \\ 3 & 10 \\ -1 & 4 \end{pmatrix} = \begin{pmatrix} 17 & 17 \\ 11 & 40 \\ 1 & 11 \end{pmatrix}.$$

Для квадратной матрицы размера $n \times n$ вводится понятие определителя.

Определителем квадратной матрицы порядка $n \times n$ (определителем порядка n) называется алгебраическая сумма всевозможных произведений элементов матрицы, взятых по одному из каждой строки, по одному из каждого столбца и снабженных знаками плюс и минус по некоторому определенному правилу. Это правило мы сформулируем позже [15, 29].

Определитель порядка n матрицы

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \hdashline a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}.$$

обозначается следующим образом:

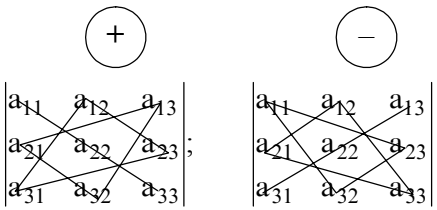
$$\det A = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \hdashline a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix}.$$

Приведем легко запоминающиеся правила для вычисления определителей второго и третьего порядков [3, 19]:

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{21}a_{12};$$

$$\begin{vmatrix} 3 & -6 \\ 2 & 1 \end{vmatrix} = 3 \times 1 - (-6) \times 2 = 15.$$

При вычислении определителя третьего порядка составляются три произведения со знаком плюс и три со знаком минус.



$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}a_{22}a_{33} + a_{31}a_{12}a_{23} + a_{13}a_{21}a_{32} - a_{13}a_{22}a_{31} -$$

$$- a_{11}a_{32}a_{23} - a_{33}a_{21}a_{12}$$

$$\begin{vmatrix} 2 & 0 & 1 \\ -3 & 4 & 7 \\ 0 & 6 & -8 \end{vmatrix} = 2 \times 4 \times (-8) + 0 \times 0 \times 7 + 1 \times 6 \times (-3) - 0 \times 4 \times 1 -$$

$$- (-8)(-3) \times 0 - 2 \times 6 \times 7 = -64 - 18 - 84 = -166.$$

Сформулируем свойства определителей [3, 19].

1. При транспонировании матрицы ее определитель не меняется.

2. При перестановке строк (столбцов) знак определителя меняется на противоположный:

$$\begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \hline a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} = - \begin{vmatrix} a_{21} & a_{22} & \dots & a_{2n} \\ a_{11} & a_{12} & \dots & a_{1n} \\ \hline a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix}$$

3. Если все элементы строки (столбца) матрицы равны нулю, то ее определитель равен нулю:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ 0 & 0 & & 0 \\ \hline a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}; \quad \begin{vmatrix} a_{11} & a_{21} & \dots & a_{1n} \\ 0 & 0 & & 0 \\ \hline a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} = 0.$$

4. Общий множитель всех элементов строки (столбца) определителя можно выносить за его знак:

$$\begin{vmatrix} ka_{11} & a_{12} & \dots & a_{1n} \\ ka_{21} & a_{22} & \dots & a_{2n} \\ \hline ka_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} = k \cdot \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \hline a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix}, \text{ где } k \in R.$$

5. Определитель равен нулю, если все элементы минимум двух его строк (столбцов) пропорциональны:

$$\begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ ka_{11} & ka_{12} & \dots & ka_{1n} \\ \hline a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} = 0, \ k \in R.$$

6. Если каждый элемент строки (столбца) определителя есть сумма двух слагаемых, то такой определитель можно представить в виде суммы двух определителей, у одного из которых соответствующая строка (столбец) составлена из первых слагаемых суммы, а у другого — из вторых:

$$\begin{vmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \dots & a_{1n}+b_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \hline a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \hline a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} +$$

$$+ \begin{vmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \hline a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix}.$$

7. Значение определителя не изменится, если к элементам его строки (столбца) прибавить соответствующие элементы другой строки (столбца), умноженные на одно и то же вещественное число:

$$\begin{vmatrix} a_{11} + ka_{21} & a_{12} + ka_{22} & \dots & a_{1n} + ka_{2n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \hline a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} =$$

$$= \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \hline a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix}, \text{ где } k \in R.$$

Дадим понятие минора и алгебраического дополнения [3, 19].

Рассмотрим матрицу размера $m \times n$:

$$A = \left(\begin{array}{cccc} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \hline a_{m1} & a_{m2} & \dots & a_{mn} \end{array} \right).$$

Выделим в ней k различных строк и k различных столбцов, причем $1 \le k \le$ минимального значения из m и n.

Элементы выделенных строк и столбцов образуют квадратную матрицу порядка k. Определитель выделенной квадратной матрицы порядка k называют *минором k-го* порядка матрицы A.

Если в выделенную квадратную матрицу порядка k включены строки и столбцы исходной матрицы, имеющие одинаковые номера, то такой минор называется главным.

Полное обозначение минора k-го порядка следующее:

$$M\begin{pmatrix} i_1 & i_2 & \dots & i_k \\ j_1 & j_2 & \dots & j_k \end{pmatrix},$$

где i — номера выделенных строк;

j — номера выделенных столбцов.

Общее число миноров порядка k прямоугольной матрицы размера m × n можно найти по формуле

$$N_k = C_m^k \times C_{n,}^k$$

где C_m^k — число сочетаний из m по k;

C_n^k — число сочетаний из n по k .

Число миноров первого порядка совпадает с общим числом элементов исходной матрицы

$$N_1 = C_m^1 \times C_n^1 = m \times n.$$

Рассмотрим конкретный пример:

$$A = \begin{pmatrix} 5 & 3 & 1 \\ 10 & 6 & 2 \end{pmatrix}.$$

Общее число миноров первого порядка данной матрицы A равно $N_1 = C_2^1 \times C_2^1 = 6$. Например,

$$M\begin{pmatrix} 1 \\ 1 \end{pmatrix} = 5; \quad m\begin{pmatrix} 2 \\ 1 \end{pmatrix} = 10.$$

Максимальный порядок миноров данной матрицы равен двум.

Общее число миноров второго порядка равно

$$N_2 = C_2^2 \times C_2^2 = 3.$$

$$M\begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix} = \begin{vmatrix} 5 & 3 \\ 10 & 6 \end{vmatrix}; \quad M\begin{pmatrix} 1 & 2 \\ 1 & 3 \end{pmatrix} = \begin{vmatrix} 5 & 1 \\ 10 & 2 \end{vmatrix};$$

$$M\begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix} = \begin{vmatrix} 3 & 1 \\ 6 & 2 \end{vmatrix}.$$

Рассмотрим теперь квадратную матрицу размера n × n.

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ a_{31} & a_{32} & \dots & a_{3n} \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}.$$

Вычеркнем в ней все элементы i-й строки и j-го столбца. Оставшиеся элементы образуют квадратную матрицу размера $(n-1) \times (n-1)$. Определитель этой матрицы будет минором $(n-1)$ порядка исходной матрицы A.

Например, вычеркнем в матрице A первую строку и второй столбец и получим следующий минор $(n-1)$ порядка исходной матрицы:

$$M_{12} = \begin{pmatrix} a_{21} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{33} & \dots & a_{3n} \\ a_{n1} & a_{n3} & \dots & a_{nn} \end{pmatrix}.$$

Минор обозначаем по номеру элемента, который стоит на пересечении вычеркиваемой строки и вычеркиваемого столбца. В нашем случае это элемент a_{12}.

Алгебраическим дополнением элемента a_{ij} квадратной матрицы порядка n называется число, вычисляемое по формуле $A_{ij} = (-1)^{i+j} M_{ij}$, т. е., если сумма номеров строки и столбца — четная, алгебраическое дополнение будет совпадать с соответствующим минором, а если нечетная, то алгебраическое дополнение и минор будут иметь разные знаки.

Используя понятие алгебраического дополнения, можно сформулировать общее правило вычисления определителей n-го порядка. Он вычисляется с помощью формул разложения по элементам какой-либо строки или какого-либо столбца. Всего существует 2xn формул разложения определителя (по элементам n-строк и n-столбцов) [3, 19, 29].

Например, приведем разложение по элементам первой строки и второго столбца:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} = a_{11}A_{11} + a_{12}A_{12} + \dots + a_{1n}A_{1n} = a_{12}A_{12} +$$

$+ a_{22}A_{22} + \dots + a_{n2}A_{n2}$.

Каждый элемент строки (столбца) умножается на соответствующее алгебраическое дополнение.

Теоретически с помощью формул разложения можно вычислить определитель квадратной матрицы любого порядка, но реально эти формулы используются для нахождения определителей не выше 4-го. Объем вычислений можно несколько сократить, если использовать свойства определителей.

Из формул разложения следуют приведенные нами выше правила вычисления определителей второго и третьего порядков.

Рассмотрим конкретные примеры вычисления определителей.

$$\begin{vmatrix} 2 & 5 & 1 \\ 3 & -1 & 6 \\ 4 & 3 & 7 \end{vmatrix} = 2\times(-1)^{1+1}\begin{vmatrix} -1 & 6 \\ 3 & 7 \end{vmatrix} + 5\times(-1)^{1+2}\begin{vmatrix} 3 & 6 \\ 4 & 7 \end{vmatrix} + 1\times(-1)^{1+3}\begin{vmatrix} 3 & -1 \\ 4 & 3 \end{vmatrix} =$$

$= 2(–7 – 18) – 5(21 – 24) + (9 – (–4)) = –50 + 15 + 13 = –22.$

В данном примере мы разложили определитель по элементам первой строки.

$$\begin{vmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 2 & 3 \\ 3 & 4 & 1 & 2 \\ 2 & 3 & 4 & 1 \end{vmatrix}.$$

К этому определителю сначала применим свойство номер 7. Первую строку определителя умножим последовательно на (–4); (–3); (–2) и сложим со 2-й; 3-й и 4-й строками. В результате получим:

$$\begin{vmatrix} 1 & 2 & 3 & 4 \\ 0 & -7 & -10 & -13 \\ 0 & -2 & -8 & -10 \\ 0 & -1 & -2 & -7 \end{vmatrix}.$$

Полученный определитель разложим по элементам 1-го столбца:

$$\begin{vmatrix} 1 & 2 & 3 & 4 \\ 0 & -7 & -10 & -13 \\ 0 & -2 & -8 & -10 \\ 0 & -1 & -2 & -7 \end{vmatrix} = 1 \times (-1)^{1+1} \begin{vmatrix} -7 & -10 & -13 \\ -2 & -8 & -10 \\ -1 & -2 & -7 \end{vmatrix}.$$

К полученному определителю вновь применим свойство номер 7. Умножим последовательно третью строку на (–2) и на (–7) и сложим со второй и первой строчками. Получим:

$$\begin{vmatrix} 0 & 4 & 36 \\ 0 & -4 & 4 \\ -1 & -2 & -7 \end{vmatrix}.$$

Последний определитель разложим по элементам первого столбца, т. е.

$$\begin{vmatrix} 0 & 4 & 36 \\ 0 & -4 & 4 \\ -1 & -2 & -7 \end{vmatrix} = (-1) \times (-1)^{3+1} \begin{vmatrix} 4 & 36 \\ -4 & 4 \end{vmatrix} = -[16 - (-144)] = -160.$$

Теперь рассмотрим обратную матрицу и правило ее вычисления.

Квадратная матрица A называется *вырожденной*, если ее определитель равен нулю, т. е. detA = 0. В противоположном случае (detA ≠ 0) матрица A является невырожденной. Любой *невырожденной* матрице A соответствует единственная обратная матрица A^{-1}, причем выполняется равенство

$A^{-1} \times A = A \times A^{-1} = E$.

Приведем алгоритм нахождения обратной матрицы [13, 19].

1. Вычислить определитель матрицы A и убедиться, что он не равен нулю.

2. Составить матрицу A_1 из алгебраических дополнений матрицы A:

$$A_1 = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ A_{n1} & A_{n2} & \dots & A_{nn} \end{pmatrix}.$$

3. Составить присоединенную матрицу (B), получаемую транспонированием матрицы A_1:

$$B = A_1^T = \begin{pmatrix} A_{11} & A_{21} & \dots & A_{n1} \\ A_{12} & A_{22} & \dots & A_{n2} \\ A_{1n} & A_{2n} & \dots & A_{nn} \end{pmatrix}.$$

4. Вычислить обратную матрицу по формуле

$$A^{-1} = \frac{1}{\det A} \times B = \begin{pmatrix} \frac{A_{11}}{\det A} & \frac{A_{21}}{\det A} & \dots & \frac{A_{n1}}{\det A} \\ \frac{A_{12}}{\det A} & \frac{A_{22}}{\det A} & \dots & \frac{A_{n2}}{\det A} \\ \frac{A_{1n}}{\det A} & \frac{A_{2n}}{\det A} & \dots & \frac{A_{nn}}{\det A} \end{pmatrix}.$$

5. Проверка полученного результата:

$A \times A^{-1} = A^{-1} \times A = E$.

Теперь рассмотрим конкретные примеры на обращение матриц.

Пример 2.1.1. Дано:

$$A = \begin{pmatrix} 3 & 2 \\ -1 & 5 \end{pmatrix}.$$

Найти: A^{-1}.

Находить A^{-1} будем в соответствии с приведенным алгоритмом. Найдем определитель исходной матрицы:

$$\det A = \begin{vmatrix} 3 & 2 \\ -1 & 5 \end{vmatrix} = 3 \times 5 - (-1) \times 2 = 17,$$

т. е. $\det A \neq 0$, поэтому у матрицы A есть обратная A^{-1}.

Теперь найдем алгебраическое дополнение:

$A_{11} = 5$; $A_{12} = 1$; $A_{21} = -2$; $A_{22} = 3$.

Составим из найденых алгебраических дополнений матрицу A_1.

$$A_1 = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} 5 & 1 \\ -2 & 3 \end{pmatrix}.$$

Найдем матрицу B (транспонируем матрицу A_1):

$$B = A_1^T = \begin{pmatrix} A_{11} & A_{21} \\ A_{12} & A_{22} \end{pmatrix} = \begin{pmatrix} 5 & -2 \\ 1 & 3 \end{pmatrix}.$$

Вычисляем обратную матрицу

$$A^{-1} = \frac{1}{\det A} \times B = \frac{1}{17} \times \begin{pmatrix} 5 & -2 \\ 1 & 3 \end{pmatrix} = \begin{pmatrix} \frac{5}{17} & -\frac{2}{17} \\ \frac{1}{17} & \frac{3}{17} \end{pmatrix}.$$

Проверим правильность вычисления обратной матрицы:

$$A \times A^{-1} = \begin{pmatrix} 3 & 2 \\ -1 & 5 \end{pmatrix} \times \begin{pmatrix} \frac{5}{17} & -\frac{2}{17} \\ \frac{1}{17} & \frac{3}{17} \end{pmatrix} = \begin{pmatrix} \frac{15}{17} + \frac{2}{17} & -\frac{6}{17} + \frac{6}{17} \\ -\frac{5}{17} + \frac{5}{17} & \frac{2}{17} + \frac{15}{17} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$

т. е. обратная матрица (A^{-1}) вычислена верно.

Пример 2.1.2. Дано:

$$A = \begin{pmatrix} 1 & -2 & 4 \\ 2 & 1 & -7 \\ 3 & -4 & 2 \end{pmatrix}.$$

Найти: A^{-1}.

Вычисляем определитель матрицы A:

$$\det A = \begin{vmatrix} 1 & -2 & 4 \\ 2 & 1 & -7 \\ 3 & -4 & 2 \end{vmatrix} = \begin{vmatrix} 1 & -2 & 4 \\ 0 & 5 & -15 \\ 0 & 2 & -10 \end{vmatrix}.$$

Умножаем первую строку последовательно на (–2) и на (–3) и складываем со второй и третьей, затем полученный определитель раскладываем по элементам первого столбца:

$$\begin{vmatrix} 1 & -2 & 4 \\ 0 & 5 & -15 \\ 0 & 2 & -10 \end{vmatrix} = 1 \times 1^{(1+1)} \begin{vmatrix} 5 & -15 \\ 2 & -10 \end{vmatrix} = -50 + 30 = -20.$$

detA ≠ 0, т. е. исходная матрица A — невырожденная и у нее есть обратная матрица.

Теперь найдем алгебраические дополнения всех элементов матрицы A:

$$A_{11} = \begin{vmatrix} 1 & -7 \\ -4 & 2 \end{vmatrix} = -26; \quad A_{12} = -\begin{vmatrix} 2 & -7 \\ 3 & 2 \end{vmatrix} = -25;$$

$$A_{13} = \begin{vmatrix} 2 & 1 \\ 3 & -4 \end{vmatrix} = -11; \quad A_{21} = -\begin{vmatrix} -2 & 4 \\ -4 & 2 \end{vmatrix} = -12;$$

$$A_{22} = \begin{vmatrix} 1 & 4 \\ 3 & 2 \end{vmatrix} = -10; \quad A_{23} = -\begin{vmatrix} 1 & -1 \\ 3 & -4 \end{vmatrix} = -2;$$

$$A_{31} = \begin{vmatrix} -2 & 4 \\ 1 & -7 \end{vmatrix} = 10; \quad A_{32} = -\begin{vmatrix} 1 & 4 \\ 2 & -7 \end{vmatrix} = 15;$$

$$A_{33} = \begin{vmatrix} 1 & -2 \\ 2 & 1 \end{vmatrix} = 5.$$

Из найденных алгебраических дополнений составляем матрицу A_1:

$$A_1 = \begin{pmatrix} -26 & -25 & -11 \\ -12 & -10 & -2 \\ 10 & 15 & 5 \end{pmatrix}.$$

Находим матрицу $B = A_1^T$:

$$B = \begin{pmatrix} -26 & -12 & 10 \\ -25 & -10 & 15 \\ -11 & -2 & 5 \end{pmatrix}.$$

Вычисляем обратную матрицу:

$$A^{-1} = \frac{1}{\det A} \times B = \frac{1}{-20} \begin{pmatrix} -26 & -12 & 10 \\ -25 & -10 & 15 \\ -11 & -2 & 5 \end{pmatrix} = \begin{pmatrix} \frac{26}{20} & \frac{12}{20} & -\frac{10}{20} \\ \frac{25}{20} & \frac{10}{20} & -\frac{15}{20} \\ \frac{11}{20} & \frac{2}{20} & -\frac{5}{20} \end{pmatrix}.$$

Делаем проверку правильности вычисления обратной матрицы $A \times A^{-1} = E$:

$$\begin{pmatrix} 1 & -2 & 4 \\ 2 & 1 & -7 \\ 3 & -4 & 2 \end{pmatrix} \times \begin{pmatrix} \frac{26}{20} & \frac{12}{20} & -\frac{10}{20} \\ \frac{25}{20} & \frac{10}{20} & -\frac{15}{20} \\ \frac{11}{20} & \frac{2}{20} & -\frac{5}{20} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Из проверки следует, что обратная матрица вычислена верно. Имеют место следующие равенства:

$\det A^{-1} = (\det A)^{-1}$;

$(A^{-1})^T = (A^T)^{-1}$;
$(A \times B \times C \times ... \times F)^{-1} = F^{-1} \times ... \times C^{-1} \times B^{-1} \times A^{-1}$.

Ранг матриц

Любая матрица кроме своего порядка должна характеризоваться еще одним показателем, который устанавливает количество ее независимых строк и столбцов. Этот показатель и называют рангом матрицы. Дадим его определение [13, 19].

Рангом (r(A)) *матрицы* A называют наивысший порядок отличных от нуля миноров этой матрицы. Ранг имеет любая матрица.

Ранг матрицы считается равным нулю, если все элементы матрицы равны нулю.

Для матриц высокого порядка разработаны специальные вычислительные методы определения ранга, например, методы жордановых исключений. Из приведенных выше свойств определителей следует, что ранг матрицы не изменяется: при ее транспонировании, при перестановке каких-либо строк или столбцов, при умножении каждого элемента строки или столбца на одно и то же число, при сложении элементов какой-то строки (столбца) с соответствующими элементами другой строки (столбца), умноженными на действительное число.

Без доказательства приведем теорему и следствия из нее [19].

ТЕОРЕМА 2.1. *Если ранг матрицы равен k, то существует k линейно-независимых строк, от которых линейно зависят все остальные строки матрицы.*

Следствие 1. Если ранг матрицы равен k, то она имеет k линейно-независимых столбцов, от которых линейно зависят остальные столбцы.

Следствие 2. Максимальное число линейно-независимых строк матрицы совпадает с максимальным числом линейно-независимых столбцов и равно рангу матрицы.

2.2. Системы линейных алгебраических уравнений

Рассмотрим системы линейных алгебраических уравнений (СЛАУ).

Линейная система m уравнений с n неизвестными — это система вида:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \hdashline a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m \end{cases} \qquad (2.1)$$

В (2.1) $i = \overline{1,m}$, $j = \overline{1,n}$, а a_{ij} — коэффициенты;

$x_1, x_2, \ldots, x_n$ — неизвестные;

$b_1, b_2, \ldots, b_m$ — свободные члены.

Систему (2.1) можно записать в матричном виде:

$AX = B$,

где $A = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \hdashline a_{m1} & a_{m2} & \ldots & a_{mn} \end{pmatrix}$ — матрица системы; (2.2)

$B = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$ — вектор свободных членов;

$X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$ — вектор неизвестных.

Решением СЛАУ называется любая совокупность n чисел $(x_1, x_2, \ldots, x_n)$, которая обращает каждое уравнение системы (2.1) в верное равенство.

Любая СЛАУ вида (2.1) может иметь одно решение, бесконечное множество решений, ни одного решения.

Если СЛАУ имеет хотя бы одно решение, то она называется *совместной*. Если СЛАУ не имеет решений, то она — *несовместная*.

Если все свободные члены СЛАУ (2.1) равны нулю, то система называется *однородной*, а если хотя бы один из свободных членов системы не равен нулю, то она называется *неоднородной*. Система однородных уравнений всегда совместна, т. е. она имеет хотя бы одно решение ($x_j = 0$).

Перед решением СЛАУ надо убедится в ее совместности. Поэтому приведем без доказательства теорему Кронекера-Капелли, которая позволяет это сделать.

Дополним матрицу A системы (2.2) столбцом свободных членов. В результате этого получим матрицу порядка $m \times (n + 1)$, которую называют расширенной матрицей системы (C):

$$C = \left(\begin{array}{cccc|c} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \hline a_{m1} & a_{m2} & \dots & a_{mn} & b_m \end{array}\right).$$

Через r(A) и r(C) обозначим ранги матриц A и C соответственно. Теперь сформулируем теорему [19].

ТЕОРЕМА 2.2. *Для того чтобы СЛАУ вида* (2.2) *была совместна, необходимо и достаточно, чтобы ранг матрицы системы* (r(A)) *был равен рангу расширенной матрицы системы* (r(C)), *т. е.* r(A) = r(C). *Здесь возможны два случая: 1) если* r(A) = r(C) = n, *где* n — *число неизвестных в системе* (2.1), *то СЛАУ имеет единственное решение; 2) если* r(A) = r(C) < n, *то СЛАУ имеет бесконечное множество решений.*

СЛАУ можно решать или прямым, или итерационным методом.

В прямых (точных) методах решение системы (2.2) находится за конечное число арифметических действий. К прямым методам

относятся метод Гаусса и его модификации, метод квадратного корня, метод Крамера и др.

Итерационные методы (методы последовательных приближений) состоят в том, что решение системы (2.2) находится как lim (предел) при $k \to \infty$ последовательных приближений $x^{(k)}$, где k — номер итерации. Обычно за конечное число итераций этот предел не достигается. Как правило, задается некоторое малое число $\varepsilon > 0$ (точность) и вычисления проводятся до тех пор, пока не будет выполнено условие:

$$\left| x_j^{(k)} - x_j^{(k-1)} \right| < \varepsilon.$$

К итерационным методам относятся: метод Якоби, метод Зейделя, метод релаксации, метод минимальных невязок, метод скорейшего спуска и др. [1, 24].

В предлагаемом учебном пособии мы рассмотрим прямые методы: метод Гаусса и метод Крамера.

Рассмотрим метод Гаусса решения СЛАУ. Он состоит из двух шагов. На первом шаге мы приводим исходную систему уравнений к верхнему треугольному виду, а на втором шаге находим неизвестные (x_j), начиная с последнего [1, 17].

Предположим, что мы имеем систему n уравнений с n неизвестными и она является совместной, т. е.

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \hdashline a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{cases} \qquad (2.3)$$

Исключаем неизвестное x_1 из всех уравнений, начиная со второго. Для этого из второго уравнения почленно вычтем первое, умноженное на $\frac{a_{21}}{a_{11}}$, из третьего почленно вычтем первое, умноженое на $\frac{a_{31}}{a_{11}}$, и т. д., причем $a_{11} \neq 0$; если $a_{11} = 0$, то переставляем местами уравнения системы (2.3). После этого система (2.3) примет вид:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{22}^{(1)}x_2 + \ldots + a_{2n}^{(1)}x_n = b_2^{(1)} \\ a_{n2}^{(1)}x_2 + \ldots + a_{nn}^{(1)}x_n = b_n^{(1)} \end{cases}, \qquad (2.4)$$

где $a_{ij}^{(1)} = a_{ij} - \frac{a_{i1}}{a_{11}}a_{1j}$; $b_i^{(1)} = b_i - \frac{a_{i1}}{a_{11}}b_1$, (i, j = 2, 3, ..., n).

В системе (2.4) исключаем неизвестное x_2 из всех уравнений, начиная с третьего, т. е. ведущим элементом становится $a_{22}^{(1)} \neq 0$, если он равен нулю, то мы переставляем уравнение местами. Затем из третьего уравнения системы (2.4) мы вычитаем второе, умноженное на коэффициент $\frac{a_{32}^{(1)}}{a_{22}^{(1)}}$, из четвертого уравнения системы (2.4) мы вычитаем второе, умноженное на коэффициент $\frac{a_{42}^{(1)}}{a_{22}^{(1)}}$, и т. д. В результате мы получаем следующую систему уравнений:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + + a_{13}x_3 + \ldots + a_{1n}x_n = b_1 \\ a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + \ldots + a_{2n}^{(1)}x_n = b_2^{(1)} \\ a_{33}^{(2)}x_3 + \ldots + a_{3n}^{(2)}x_n = b_3^{(2)} \\ a_{n3}^{(2)}x_3 + \ldots + a_{nn}^{(2)}x_n = b_n^{(2)} \end{cases}, \qquad (2.5)$$

где $a_{ij}^{(2)} = a_{ij}^{(1)} - \frac{a_{i2}^{(1)}}{a_{22}^{(1)}}a_{2j}^{(1)}$; $b_i^{(2)} = b_i^{(1)} - \frac{a_{i2}^{(1)}}{a_{22}^{(1)}}b_2^{(1)}$.

Аналогичный процесс мы продолжаем далее и на (n – 1) – М шаге приходим к следующей системе уравнений:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n = b_1 \\ a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + \ldots + a_{2n}^{(1)}x_n = b_2^{(1)} \\ a_{33}^{(2)}x_3 + \ldots + a_{3n}^{(2)}x_n = b_3^{(2)} \\ a_{nn}^{(n-1)}x_n = b_n^{(n-1)} \end{cases}. \qquad (2.6)$$

Таким образом исходную систему уравнений (2.3) мы привели к верхнему треугольному виду (первый шаг метода Гаусса завершен). Второй шаг (обратный ход) заключается в решении системы уравнений (2.6). Он осуществляется следующим образом: из последнего уравнения системы (2.6) находим $x_n = \frac{b_n^{(n-1)}}{a_{nn}^{(n-1)}}$, используя найденное значение x_n из предпоследнего (n – 1) уравнения системы (2.6) находим x_{n-1}, затем из (n – 2) уравнение системы (2.6) находим x_{n-2} и т. д. до x_1.

Алгоритм Гаусса состоит из однотипных операций, которые легко программируются.

Решим, используя метод Гаусса, систему уравнений.

Пример 2.3

$$\begin{cases} 3x_1 + x_2 - 5x_3 = 8 \\ 12x_1 + x_2 - 3x_3 = 24 \\ -15x_1 + 4x_2 + 0x_3 = -42 \end{cases} \quad (2.7)$$

Исключим x_1 из второго и третьего уравнений системы (2.7). Для этого умножим первое уравнение почленно на 4 и вычтем из второго, затем умножим первое уравнение на (–5) и вычтем из третьего. В результате получим следующую систему уравнений:

$$\begin{cases} 3x_1 + x_2 - 5x_3 = 8 \\ -3x_2 + 17x_3 = -8 \\ 9x_2 - 25x_3 = -2 \end{cases} \quad (2.8)$$

Теперь исключим из третьего уравнения системы (2.8) неизвестное x_2. Для этого умножим поэлементно второе уравнение системы (2.8) на (–3) и вычтем из третьего уравнения системы (2.8).

В результате получим следующую систему уравнений:

$$\begin{cases} 3x_1 + x_2 - 5x_3 = 8 \\ -3x_2 + 17x_3 = -8 \\ 26x_3 = -26 \end{cases} \quad (2.9)$$

Из системы уравнений (2.9) последовательно находим неизвестные x, начиная с последнего (x_3), т. е. $x_3 = -1$:

$$x_2 = \frac{17x_3 + 8}{3} = -3; \quad x_1 = \frac{8 - x_3 + 5x_3}{3} = 2.$$

Теперь рассмотрим метод Крамера для решения СЛАУ.

Рассмотрим систему уравнений (2.3), которую запишем в матричном виде

$$AX = B, \tag{2.10}$$

где $A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \hline a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$;

$B = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$;

$X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$.

Если определитель матрицы A (det A) не равен нулю, то существует A^{-1}.

Домножим слева систему (2.10) на A^{-1}, получим:

$A^{-1}AX = A^{-1}B$, так как $A^{-1}A = E$, то имеем $EX = A^{-1}B$, а так как $EX = X$, то окончательно получим $X = A^{-1}B$ (2.11), или в развернутом виде:

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \frac{1}{\det A} \begin{pmatrix} A_{11} & A_{21} & \dots & A_{n1} \\ A_{12} & A_{22} & \dots & A_{n2} \\ \hline A_{1n} & A_{2n} & \dots & A_{nn} \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \tag{2.12}$$

Из (2.12) следует:

$$\left.\begin{array}{l} x_1 = \dfrac{A_{11}b_1 + A_{21}b_2 + \ldots + A_{n1}b_n}{\det A} \\ \hline x_n = \dfrac{A_{1n}b_1 + A_{2n}b_2 + \ldots + A_{nn}b_n}{\det A} \end{array}\right\} \qquad (2.13)$$

Числители равенств (2.13) есть разложения по элементам 1, 2, ... n-го столбцов определителя, полученного из det A заменой в нем 1, 2, ..., n столбцов столбцом свободных членов, т. е.

$$\Delta_1 = \begin{pmatrix} b_1 & a_{12} & \ldots & a_{1n} \\ b_2 & a_{22} & \ldots & a_{2n} \\ \hline b_n & a_{n2} & \ldots & a_{nn} \end{pmatrix} \quad \Delta_n = \begin{pmatrix} a_{11} & a_{12} & \ldots & b_1 \\ a_{21} & a_{22} & \ldots & b_2 \\ \hline a_{n1} & a_{n2} & \ldots & b_n \end{pmatrix}$$

Таким образом, неизвестные x_i можно найти по формуле:

$$x_i = \frac{\Delta_i}{\det A}, \qquad (2.14)$$

где i = 1, 2, ..., n [19].

Решим систему уравнений, используя метод Крамера.

Пример 2.4.

$$\begin{cases} x_1 - 2x_2 + 2x_3 = 0 \\ 3x_1 - 4x_2 + 5x_3 = 0 \\ 2x_1 + 5x_2 - x_3 = 1 \end{cases}$$

Сначала найдем определитель исходной системы:

$$\det A = \begin{vmatrix} 1 & -2 & 2 \\ 3 & -4 & 5 \\ 2 & 5 & -1 \end{vmatrix} = 1\begin{vmatrix} -4 & 5 \\ 5 & -1 \end{vmatrix} + 2\begin{vmatrix} 3 & 5 \\ 2 & -1 \end{vmatrix} + 2\begin{vmatrix} 3 & -4 \\ 2 & 5 \end{vmatrix} = -21 - 26 +$$

$+ 46 = -1$.

Затем находим

$$\Delta_1 = \begin{vmatrix} 0 & -2 & 2 \\ 0 & -4 & 5 \\ 1 & 5 & -1 \end{vmatrix} = \begin{vmatrix} -2 & 2 \\ -4 & 5 \end{vmatrix} = -2;$$

$$\Delta_2 = \begin{vmatrix} 1 & 0 & 2 \\ 3 & 0 & 5 \\ 2 & 1 & -1 \end{vmatrix} = -\begin{vmatrix} 1 & 2 \\ 3 & 5 \end{vmatrix} = 1;$$

$$\Delta_3 = \begin{vmatrix} 1 & -2 & 0 \\ 3 & -4 & 0 \\ 2 & 5 & 1 \end{vmatrix} = \begin{vmatrix} 1 & -2 \\ 3 & -4 \end{vmatrix} = 2$$

и определяем неизвестные x_i; i = 1, 2, 3:

$$x_1 = \frac{\Delta_1}{\det A} = 2; \quad x_2 = \frac{\Delta_2}{\det A} = -1; \quad x_3 = \frac{\Delta_3}{\det A} = -2.$$

2.3. Собственные числа и собственные векторы матриц

Проблема собственных чисел играет существенную роль не только в линейной алгебре, но и в других разделах математики, а также во многих прикладных областях (в менеджменте, психологии, юриспруденции) [12].

Пусть задана квадратная матрица A размера (n × n), элементами которой являются действительные числа (R) и вектор неизвестных X размера (n × 1):

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}; \quad X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}.$$

Предположим, что λ — это некоторое неизвестное действительное число.

Если λ и ненулевой вектор X удовлетворяют уравнению

$$A \times X = \lambda \times X, \tag{2.15}$$

то λ называется собственным числом или собственным значением матрицы A, а X — собственным вектором этой же матрицы, соответствующим λ [12, 15].

Преобразуем уравнение (2.15) к следующему виду:

$$\lambda \times X - A \times X = 0, (\lambda E - A) \times X = 0, \tag{2.16}$$

где E — единичная матрица.

Матрица $(\lambda E - A) = \begin{pmatrix} \lambda - a_{11} & -a_{12} & \dots & -a_{1n} \\ -a_{21} & \lambda - a_{22} & \dots & -a_{2n} \\ \dots & \dots & \dots & \dots \\ -a_{n1} & -a_{n2} & \dots & \lambda - a_{nn} \end{pmatrix}$ называется

характеристической матрицей [18].

Так как по условию вектор неизвестных X не равен нулю, то среди его координат $x_1, x_2, \dots, x_n$ должна быть хотя бы одна ненулевая. Для того чтобы система линейных однородных уравнений (2.16) имела ненулевое решение, необходимо и достаточно, чтобы определитель этой системы был равен нулю (это следует из теоремы Кронекера-Капелли).

Поэтому получаем:

$$\det(\lambda E - A) = \begin{pmatrix} \lambda - a_{11} & -a_{12} & \dots & -a_{1n} \\ -a_{21} & \lambda - a_{22} & \dots & -a_{2n} \\ \dots & \dots & \dots & \dots \\ -a_{n1} & -a_{n2} & \dots & \lambda - a_{nn} \end{pmatrix} = 0. \tag{2.17}$$

Число $\lambda = \lambda_k$, где $k = \overline{1,n}$, будет собственным числом только в том случае, если матрица $(\lambda_k E - A)$ — вырожденная.

Уравнение (2.17) называется характеристическим уравнением матрицы A и представляет собой алгебраическое уравнение степени n относительно λ [18]:

$$\det(\lambda E - A) = \lambda^n + p_1\lambda^{n-1} + p_2\lambda^{n-2} + \dots + p_n = 0. \tag{2.18}$$

Уравнение имеет n корней $\lambda_1, \lambda_2, ..., \lambda_n$. Множество всех корней (2.18) называется спектром матрицы A.

Заметим, что уравнение $\det (A - \lambda E) = 0$ имеет те же корни, что и уравнение (2.17), т. е.

$$\det (A - \lambda E) = (-1)^n \det (\lambda E - A).$$

Каждому собственному значению спектра матрицы A ставится в соответствие собственный вектор, определенный с точностью до скалярного множителя. Если λ_k есть кратный корень характеристического уравнения, то для произвольной квадратной матрицы число соответствующих собственных векторов может быть не равно кратности корня. С геометрической точки зрения собственный вектор определяет в пространстве некоторое направление (прямую, проходящую через начало координат), которое в результате преобразования не изменяется и вдоль которого пространство испытывает растяжение или сжатие в λ раз [3, 18].

Полином $\lambda^n + p_1\lambda^{n-1} + ... + p_n = 0$ называют характеристическим полиномом. Коэффициенты p_k ($k = \overline{1,n}$) можно вычислить по следующим рекуррентным формулам [29]:

$$\begin{cases} p_1 = -SpA \\ 2p_2 = -SpA^2 - p_1 SpA \\ 3p_3 = -SpA^3 - p_1 SpA^2 - p_2 SpA \\ 4p_4 = -SpA^4 - p_1 SpA^3 - p_2 SpA^2 - p_3 SpA \\ np_n = -SpA^n - p_1 SpA^{n-1} - ... - p_{n-1} SpA \end{cases} \qquad (2.19)$$

Здесь $SpA = \sum_{k=1}^{n} a_{kk}$ — след матрицы (сумма элементов, стоящих на главной диагонали матрицы A). Заметим, что $P_n = (-1)^n \times$ $\times \det A$. При отыскании собственных чисел даже для матриц невысокого порядка неизбежно большое количество вычислений. Для общего случая нельзя предложить оптимальный способ нахождения собственных чисел и собственных векторов матрицы.

Рассмотрим случай, когда собственные числа находятся сразу исходя из вида матрицы (исходная матрица либо диагональная, либо верхняя или нижняя треугольная). В этом случае собственные числа $\lambda_1, \lambda_2, ..., \lambda_n$ совпадают с элементами главной диагонали исходной матрицы ($a_{11}, a_{22} ..., a_{nn}$).

Пусть задана верхняя треугольная матрица A размера ($n \times n$):

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ 0 & a_{22} & \dots & a_{2n} \\ 0 & 0 \; a_{33} & \dots & a_{3n} \\ \hline 0 & 0 \; 0 & \dots & a_{nn} \end{pmatrix}.$$

Тогда имеем:

$$\det(\lambda E - A) = \begin{vmatrix} \lambda - a_{11} & -a_{12} & \dots & -a_{1n} \\ & \lambda - a_{22} & \dots & -a_{2n} \\ \bigcirc & & \ddots & \\ & & & \lambda - a_{nn} \end{vmatrix} = (\lambda - a_{11})(\lambda - a_{22}) \times$$

$\times \dots \times (\lambda - a_{nn}) = 0.$

Отсюда видно, что собственные числа равны: $\lambda_1 = a_{11}, \lambda_2 = a_{22}, ..., \lambda_n = a_{nn}$.

С появлением ЭВМ получили распространение итерационные методы нахождения собственных чисел, которые не используют вычисление характеристического полинома. К этим способам относятся: степенной метод, метод обратных итераций, QR-алгоритм, метод вращений Якоби, QL-алгоритм и другие. Причем применение конкретного итерационного метода зависит от вида исходной матрицы A [1].

Теперь рассмотрим конкретные примеры.

Пример 2.5. Дана матрица A размера (3×3):

$$A = \begin{pmatrix} -4 & 4 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Найти собственные числа и собственные векторы матрицы A.

Из условия задачи видно, что матрица A является верхней треугольной матрицей. Поэтому собственными числами данной матрицы будут элементы ее главной диагонали.

$$\det(\lambda E - A) = \begin{vmatrix} \lambda + 4 & -4 & -1 \\ 0 & \lambda - 1 & 0 \\ 0 & 0 & \lambda - 1 \end{vmatrix} = (\lambda + 4)(\lambda - 1)(\lambda - 1) = 0 \Rightarrow \lambda_1 =$$

$= -4$, $\lambda_2 = \lambda_3 = 1$

Теперь найдем соответствующие найденным собственным числам собственные векторы. Для этого мы используем уравнение (2.16).

Для $\lambda_1 = -4$ получаем:

$$(-4E - A) \times X_1 = 0, \tag{2.20}$$

где $X_1 = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$.

Далее раскроем матричное уравнение (2.20)

$$\left[-4 \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} -4 & 4 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \right] \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

В результате получим

$$\begin{pmatrix} 0 & -4 & -1 \\ 0 & -5 & 0 \\ 0 & 0 & -5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \text{ или } \begin{cases} -4x_2 - x_3 = 0 \\ -5x_2 = 0 \\ -5x_3 = 0 \end{cases}$$

Так как матрица этой системы вырождена, то она имеет ненулевые решения, которые имеют вид:

$X_1 = \begin{pmatrix} x_1 \\ 0 \\ 0 \end{pmatrix}$, $x_1 \neq 0$, $x_1 \in R$, т. е. мы получили искомые собственные векторы для λ_1.

Для $\lambda_2 = \lambda_3 = 1$ получаем:

$(E - A)X_2 = 0$, где $X_2 = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$ или в подробной записи

$$\begin{pmatrix} 5 & -4 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

В результате получаем:

$5x_1 - 4x_2 - x_3 = 0$ или

$x_3 = 5x_1 - 4x_2$,

т. е. это уравнение имеет ненулевые решения, которые и будут искомыми собственными векторами для λ_2.

Эти решения запишем в виде:

$$X_2 = \begin{pmatrix} x_1 \\ x_2 \\ 5x_1 - 4x_2 \end{pmatrix}, \quad x_1^2 + x_2^2 \neq 0, \; x_1, x_2 \in R.$$

В заключение приведем два полезных правила [18]:

1. Сумма собственных чисел матрицы A равно следу этой матрицы, т. е.

$$\sum_{i=1}^{n} \lambda_i = a_{11} + a_{22} + \ldots + a_{nn}.$$

2. Произведение собственных чисел матрицы A равно определителю этой матрицы:

$$\prod_{i=1}^{n} \lambda_i = \det A.$$

2.4. Некоторые сведения о векторах

Цифровые данные, используемые в экономике, можно представить в виде списков чисел, каждое из которых имеет определенный смысл.

Например, списки цен различных товаров в магазинах, объемы продукции разных видов, выпущенных каким-либо предприятием за год, и т. д. В математике такие упорядоченные списки чисел называют векторами. Дадим определение n-мерного вектора (n = 1, 2 ...).

Упорядоченный набор n *чисел* $x_1, x_2, x_3, ..., x_n$ *называется* n-*мерным вектором*. Мы будем обозначать векторы заглавными буквами со стрелками над ними, т. е.

$\vec{X}(x_1, x_2, x_3, ..., x_n)$, числа $x_1, x_2, x_3, ..., x_n$ есть координаты вектора, а n — его размерность [12, 13].

Два n-мерных вектора называются равными, если их соответствующие координаты равны, например:

$\vec{X}(2, 3, 7, 12)$; $\vec{Y} = (2, 3, 7, 12) \Rightarrow \vec{X} = \vec{Y}$.

Вектор, все координаты которого нули, называется *ноль-вектором* и обозначается $\vec{0}$.

Алгебраической суммой двух n-мерных векторов $\vec{X} = (x_1, x_2, ..., x_n)$ и $\vec{Y} = (y_1, y_2, ..., y_n)$ называется вектор $\vec{X} \pm \vec{Y}$, каждая координата которого равна алгебраической сумме соответствующих координат векторов $\vec{X}$ и $\vec{Y}$, т. е.

$$\vec{X} \pm \vec{Y} = (x_1 \pm y_1, x_2 \pm y_2, ..., x_n \pm y_n). \qquad (2.21)$$

Произведением действительного числа k на n-мерный вектор $\vec{X} = (x_1, x_2, ..., x_n)$ называется n-мерный вектор $k\vec{X}$, каждая координата которого равна произведению *числа k на соответствующую координату вектора* $\vec{X}$, т. е.

$$k\vec{X} = (kx_1, kx_2, ..., kx_n). \qquad (2.22)$$

Множество n-мерных векторов, для которых определены действия алгебраического сложения (2.21) и умножения на число (2.22), называют n-мерным векторным пространством и обозначают R^n (в случае n = 1 оно совпадает с множеством действительных чисел R).

В случае n = 2 и n = 3 имеем соответственно двумерное (R^2) и трехмерное (R^3) векторные пространства. А двумерные и трехмерные векторы имеют геометрическую интерпретацию: они изображаются направленными отрезками на плоскости и в пространстве [3, 12, 13].

Пусть в R^n заданы векторы $\vec{X} = (x_1, x_2, ..., x_n)$, $\vec{Y} = (y_1, y_2, ..., y_n)$, $\vec{Z} = (z_1, z_2, ..., z_n)$; $k \in R$; $t \in R$.

Приведем свойства линейных действий векторами [3, 12]:

1) $\vec{X} + \vec{Y} = \vec{Y} + \vec{X}$;

2) $\vec{X} + (\vec{Y} + \vec{Z}) = (\vec{X} + \vec{Y}) + \vec{Z}$;

3) $\vec{X} + \vec{0} = \vec{X}$;

4) $kt(\vec{X}) = k(t\vec{X})$;

5) $k(\vec{X} + \vec{Y}) = k\vec{X} + k\vec{Y}$;

6) $(k + t)\vec{X} = k\vec{X} + t\vec{X}$;

7) $0\vec{X} = \vec{0}$;

8) $k\vec{0} = \vec{0}$;

9) $\vec{X} - \vec{Y} = \vec{X} + (-1)\vec{Y}$.

Длина (норма) вектора $\vec{X} = (x_1, x_2, ..., x_n)$ в пространстве R^n находится по формуле

$$|\vec{X}| = \sqrt{\sum_{i=1}^{n} x_i^2}. \qquad (2.23)$$

Например, задан вектор $\vec{X} = (5, 3, -2)$.

Используя (2.23), найдем, что его длина равна

$$|\vec{X}| = \sqrt{5^2 + 3^2 + (-2)^2} = \sqrt{38}.$$

Введем понятие скалярного произведения в действительном пространстве R^n.

Скалярным произведением двух векторов $\vec{X} = (x_1, x_2, ..., x_n)$ и $\vec{Y} = (y_1, y_2, ..., y_n)$ в R^n ($x_i \in R^n$, $y_i \in R^n$, $i = \overline{1,n}$) называется число, получаемое по формулам [3, 12, 13]:

$$(\vec{X}, \vec{Y}) = \sum_{i=1}^{n} x_i y_i; \tag{2.24}$$

$$(\vec{X}, \vec{Y}) = |\vec{X}| \times |\vec{Y}| \times \cos\alpha, \tag{2.25}$$

где α есть угол между n-мерными векторами $\vec{X}$ и $\vec{Y}$ (в случае $n = 2$ и $n = 3\alpha$ будет углом между направленными отрезками на плоскости и в пространстве, а при $n > 3$ векторы $\vec{X}$ и $\vec{Y}$ являются математическими абстракциями).

Из формулы (2.25) следует, что угол между n-мерными векторами $\vec{X}$ и $\vec{Y}$ равен

$$\cos\alpha = \frac{(\vec{X}, \vec{Y})}{|\vec{X}| \cdot |\vec{Y}|} = \frac{x_1 y_1 + x_2 y_2 + ... + x_n y_n}{\sqrt{x_1^2 + x_2^2 + ... + x_n^2}\sqrt{y_1^2 + y_2^2 + ... + y_n^2}}. \tag{2.26}$$

Если угол между векторами $\vec{X}$ и $\vec{Y}$ равен $\frac{\pi}{2}$, то скалярные произведения этих векторов равны нулю, т. е.

$$(\vec{X}, \vec{Y}) = \sum_{i=1}^{n} x_i y_i = x_1 y_1 + x_2 y_2 + ... x_n y_n = 0. \tag{2.27}$$

Пример 2.6

Например, заданы векторы:

$\vec{X} = (2, 3, 7)$; $\vec{Y} = (1, 6, 5)$ в 3-мерном пространстве R^3. Найти угол между ними.

По формуле (2.26) получим:

$$\cos(\hat{\vec{X}, \vec{Y}}) = \frac{x_1 y_1 + x_2 y_2 + x_3 y_3}{\sqrt{x_1^2 + x_2^2 + x_3^2}\sqrt{y_1^2 + y_2^2 + y_3^2}} = \frac{2 + 18 + 35}{\sqrt{4 + 9 + 49}\sqrt{1 + 36 + 25}} =$$

$$= \frac{55}{\sqrt{62}\sqrt{62}} = \frac{55}{62} \approx 0{,}887097;$$

$$\alpha = \overset{\wedge}{(\vec{X}, \vec{Y})} \approx 27°29'21,''5$$

Скалярное произведение в пространстве R^n обладает следующими свойствами [3, 12]:

1) $(\vec{X}, \vec{X}) \geq 0$ (при этом равенство нулю будет только в том случае, если $\vec{X} = \vec{0}$);

2) $(\vec{X}, \vec{Y}) = (\vec{Y}, \vec{X})$;

3) $(k\vec{X} + t\vec{Y}, Z) = k(\vec{X}, \vec{Z}) + t(\vec{Y}, \vec{Z})$.

Здесь $\vec{X}, \vec{Y}, \vec{Z}$ — векторы в R^n, а k и t — действительные числа.

Пространство R^n, в котором введено понятие скалярного произведения по формуле (2.24), называется *евклидовым n-мерным пространством* [3, 12].

Задачи для самостоятельного решения

1. Найти произведения матриц:

1.1. $\begin{pmatrix} 5 & 6 & -1 \\ -3 & 9 & 0 \\ 18 & 4 & 7 \end{pmatrix} \times \begin{pmatrix} 6 & -1 \\ 5 & 4 \\ 2 & 0 \end{pmatrix}$.

1.2. $\begin{pmatrix} 2 & -1 & 17 \\ & & \\ 4 & 3 & -8 \end{pmatrix} \times \begin{pmatrix} 4 & -2 \\ 7 & 10 \\ -6 & 0 \end{pmatrix}$.

1.3. $\begin{pmatrix} 10 & -3 & 8 \\ -5 & 4 & 3 \\ -7 & 2 & 8 \end{pmatrix} \times \begin{pmatrix} 11 & 10 & 0 \\ 4 & -3 & 5 \\ 6 & 2 & -7 \end{pmatrix}$.

2. Вычислить определители:

2.1. $\begin{vmatrix} 5 & 4 & -1 & 0 \\ 2 & -4 & 7 & 6 \\ 10 & 0 & -11 & 5 \\ -3 & -7 & 8 & 11 \end{vmatrix}$.

2.2. $\begin{vmatrix} 7 & -10 & 4 & 3 \\ 1 & -1 & 18 & 0 \\ 2 & -4 & 3 & 5 \\ -4 & -3 & 9 & 15 \end{vmatrix}$.

2.3. $\begin{vmatrix} 0 & -10 & 3 & 7 \\ 6 & 13 & -8 & -7 \\ 4 & -3 & 5 & 2 \\ 8 & -1 & 4 & 3 \end{vmatrix}$.

3. Найти матрицы, обратные данным:

3.1. $\begin{pmatrix} 10 & -3 & 5 \\ 6 & 8 & 7 \\ -2 & 1 & 4 \end{pmatrix}$.

3.2. $\begin{pmatrix} 5 & 4 & 1 \\ 2 & -3 & 6 \\ 7 & 1 & 2 \end{pmatrix}$.

3.3. $\begin{pmatrix} 7 & -6 & 5 \\ 1 & 4 & 2 \\ -10 & 11 & 8 \end{pmatrix}$.

4. Найти ранги матриц:

4.1. $\begin{pmatrix} 5 & -3 & 4 & 1 \\ 8 & -7 & 12 & 4 \\ -15 & 9 & -12 & -3 \end{pmatrix}$.

4.2. $\begin{pmatrix} 2 & 11 & -7 & 4 \\ -6 & 4 & -2 & -12 \\ 3 & -6 & 8 & -6 \\ 9 & 2 & 4 & 18 \end{pmatrix}$.

4.3. $\begin{pmatrix} 5 & 4 & -3 & 7 \\ 2 & -1 & 0 & 6 \\ 0 & -3 & 7 & 1 \end{pmatrix}$.

5. Решить СЛАУ методами Гаусса и Крамера:

5.1. $\begin{cases} x_1 + 2x_2 + x_3 = 3 \\ x_1 - x_2 - 2x_3 = -3 \\ 2x_1 - 3x_2 - x_3 = 0 \end{cases}$

5.2. $\begin{cases} x_1 + 2x_2 + x_3 = 1 \\ 2x_1 - 3x_2 - x_3 = -4 \\ 3x_1 + x_2 + 2x_3 = 1 \end{cases}$

5.3. $\begin{cases} x_1 + x_2 + 2x_3 = 0 \\ x_1 + 2x_2 + 5x_3 = -1 \\ 3x_1 - x_2 - 3x_3 = 1 \end{cases}$

6. Найти собственные числа и собственные векторы матриц:

6.1. $\begin{pmatrix} 1 & 0 & 5 \\ 0 & 1 & 1 \\ 2 & -1 & 1 \end{pmatrix}$.

6.2. $\begin{pmatrix} -1 & 0 & 0 \\ 2 & 1 & 0 \\ 5 & -2 & 4 \end{pmatrix}$.

6.3. $\begin{pmatrix} 2 & 3 & 7 \\ 1 & 2 & 4 \\ 1 & 1 & 3 \end{pmatrix}$.

6.4. $\begin{pmatrix} 2 & 12 & -4 \\ -1 & -3 & 1 \\ -1 & -12 & 6 \end{pmatrix}$.

7. Дано: $\vec{X} = (1, -5, 6, 7, 10)$;
$\vec{Y} = (-2, 7, 8, 11, -6)$.
Найти угол между векторами $\vec{X}$ и $\vec{Y}$.

8. Даны два перпендикулярных вектора
$\vec{X} = (3, x_2, 7)$ и $\vec{Y} = (1, 6, 8)$.
Найти координату x_2.

Вопросы для самопроверки

1. Что называется матрицей? Типы матриц.
2. Правило и свойства сложения матриц.
3. Правило и основные свойства перемножения двух матриц.
4. Как найти матрицу, обратную заданной? Любая ли матрица имеет обратную?
5. Что называется определителем?
6. Что такое ранг матрицы?
7. Как определить, совместна ли заданная СЛАУ?
8. В каких случаях однородные СЛАУ имеют нулевые решения?
9. В чем суть итерационных методов решения СЛАУ?
10. В чем состоит метод Гаусса решения СЛАУ?
11. В чем состоит метод Крамера решения СЛАУ?

12. Какие числа называются собственными значениями матрицы?

13. Что такое след матрицы?

14. Какое уравнение называется характеристическим уравнением матрицы?

15. Дать определение n-мерного векторного пространства.

16. Что называется нормой вектора?

17. Как найти угол между двумя векторами в n-мерном векторном пространстве?

18. Какое n-мерное пространство называется евклидовым?

3. Функции и пределы

3.1. Некоторые сведения о функциях

В любой области науки мы встречаемся с различными величинами. Под величиной понимают все то, что может быть измерено и (или) вычислено и выражено числом или числами [2].

В естественных, технических и гуманитарных науках имеют дело с различными величинами, например, скоростью, силой, температурой, себестоимостью, валовым внутренним продуктом какой-либо страны, количеством преступлений в каком-то регионе и др.

В математике конкретные величины не участвуют, т. е. в ней рассматривают величины вообще, не принимая во внимание их физический смысл.

Все величины можно разделить на переменные и постоянные.

Переменной называется такая величина, которая принимает различные числовые значения. Величина, которая не меняет свое числовое значение, называется *постоянной*.

Все процессы характеризуются взаимоизменяемостью нескольких переменных величин, а это приводит к важнейшему понятию математики — функциональной зависимости [2, 22].

Часто одни и те же величины могут в одних случаях быть переменными, а в других — постоянными.

Например, в формуле $F = ma$ величины m (масса) и a (ускорение) могут быть как постоянными, так и переменными.

Но существуют и фундаментальные постоянные, которые сохраняют свое значение, по крайней мере в нашей Метагалактике.

Например, в законе всемирного тяготения $F = G\frac{m_1 m_2}{r^2}$ величина $G = 6,672 \times 10^{-11}\ \frac{м^3}{кг \cdot с^2}$ — фундаментальная постоянная.

Установление и описание связей между величинами — одна из основных задач математического анализа, который включает в себя ряд дисциплин: теорию пределов, дифференциальное и интегральное исчисления, теорию рядов и др. [2]. Некоторые сведения из этих дисциплин мы рассмотрим в гл. 3–7.

Теперь приведем определение функции одного независимого аргумента.

Величина y называется функцией величины x на множестве определения D, если каждому значению $x \in D$ по какому-то закону поставлено в соответствие одно (несколько, бесконечно много) *значение (значений) y* [2, 20].

В первом случае функция называется однозначной, например, $y = x + 1$ (см. рис. 3.1).

Во втором случае — многозначной, например, $y = \operatorname{Arcsin} x$ (см. рис. 3.2).

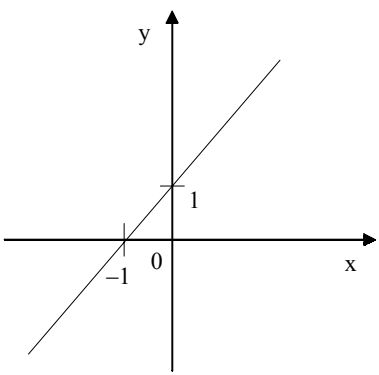


Рис. 3.1

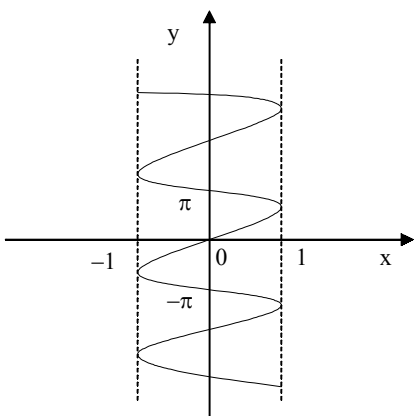


Рис. 3.2

Величину x из области D можно брать произвольно, поэтому она называется *аргументом*, или *независимой переменной*. Величина y будет зависеть от выбранной величины x, поэтому ее называют *зависимой переменной*, или *функцией*.

Область D может быть любой, но, как правило, используются области двух видов:

1) множество целых неотрицательных чисел или какие-то части этого множества;

2) один или несколько интервалов (конечных или бесконечных) числовой оси.

В первом случае имеем функцию целочисленного аргумента, а во втором — непрерывного.

Тот факт, что величина y есть функция аргумента x, обычно записывают так: $y = f(x)$.

Множество всех значений функции y обозначим через E.

Функцию можно задать с помощью таблицы, в виде графика (преимуществом этого способа является его наглядность) или аналитически (формулой). Последний способ является самым распространенным.

Все функции можно разделить на два класса: элементарные и неэлементарные.

К *элементарным функциям* относятся основные элементарные функции:

$y = x^n$ ($n \in R$), $y = a^x$ ($a > 0$, $a \neq 1$);

$y = \log_a x$ ($a > 0$, $a \neq 1$), $y = \sin x$, $y = \cos x$;

$y = \operatorname{tg} x$, $y = \operatorname{ctg} x$, $y = \sec x$, $y = \operatorname{cosec} x$;

$y = \arcsin x$, $y = \arccos x$, $y = \operatorname{arctg} x$, $y = \operatorname{arcctg} x$

и функции, полученные из основных элементарных функций при помощи конечного числа арифметических действий и конечного числа операций взятия функции от функции и заданные одной формулой [22].

Например:

$$y = \frac{6\sin^2 x - 5x^3}{\sqrt{14\operatorname{ctg} x + 2}}; y = \frac{2x^3}{\sqrt{1 - x^4}}; y = \frac{8^x - 7x^3}{\operatorname{tg} 2x}; y = \frac{e^x - 5^{2x}}{\ln 4x} \text{ и т. д.}$$

Все функции, не подходящие под данное определение, элементарными функциями не являются.

Например, функция

$$f(x) = \begin{cases} -1 \text{ при } x < 0 \\ 0 \text{ при } x = 0 \\ 1 \text{ при } x > 0 \end{cases}$$

не является элементарной функцией, так как задана тремя формулами [6, 22].

Функция $f(n) = 1 \times 2 \times 3 \times ... \times n = n!$ не будет элементарной, так как количество операций умножения, которое нужно совершить для получения f(n), не будет являться конечным.

3.2. Предел последовательности. Предел функции. Вычисление пределов

Прежде чем перейти к определению предела, напомним, что в математике используются три вида бесконечностей $+\infty$, $-\infty$, ∞. Бесконечность не является числом, она показывает, как меняется переменная величина, которая конечна в любой момент времени.

Теперь дадим понятие последовательности и ее предела.

Последовательностью называется множество чисел, которое перенумеровано с помощью целых чисел и расположено в порядке возрастания номеров [2].

Если задана последовательность $y_1, y_2, y_3, ...$, то тем самым любому целому неотрицательному значению n поставлено в соответствие значение $y_n = f(n)$.

Например, члены геометрической прогрессии $\frac{1}{3}, \frac{1}{9}, \frac{1}{27}...$ являются последовательными значениями функции

$$f(n) = \frac{1}{3^n},$$

где $n \in N$.

Может случиться так, что с увеличением n значения $y_n = f(n)$ будет неограниченно приближаться к какому-то числу a. В этом случае говорят, что число a является пределом функции f(n) целочисленного аргумента n или последовательности $y_1, y_2, ..., y_n, ...$ при $n \to \infty$ и пишут $\lim\limits_{n\to\infty} f(n) = a$, или $\lim\limits_{n\to\infty} y_n = a$.

Число a является пределом последовательности $y_1, y_2, ..., y_n, ...$ если для $\forall \varepsilon > 0$ можно найти такое $N > 0$, что для всех f(n) с номерами $n > N$ справедливо неравенство [2.22].

$|f(n) - a| < \varepsilon.$ (3.1)

Используя приведенное определение, докажем, что последовательность $\frac{1}{2}, \frac{2}{3}, \frac{3}{4}, ..., \frac{n}{n+1}, ...$ имеет предел, равный 1.

Согласно определению имеем:

$$\left|\frac{n}{n+1'} - 1\right| < \varepsilon \Rightarrow \left|\frac{-1}{n+1}\right| < \varepsilon \Rightarrow n+1 > \frac{1}{\varepsilon} \Rightarrow n > \left(\frac{1}{\varepsilon} - 1\right) = N.$$

Таким образом, мы доказали, что для любого наперед заданного $\varepsilon > 0$ можно найти такое $N = \frac{1}{\varepsilon} - 1$, что при всех $n > N$ будет выполняться (3.1), а это означает, что 1 есть предел исходной последовательности.

Теперь рассмотрим функцию $y = f(x)$ непрерывного аргумента x и предположим, что x неограниченно приближается к числу x_0 ($x \to x_0$). При этом может оказаться, что соответствующее значение f(x) неограниченно приближается к некоторому числу b. В этом случае говорят, что число b есть предел функции f(x) при $x \to x_0$.

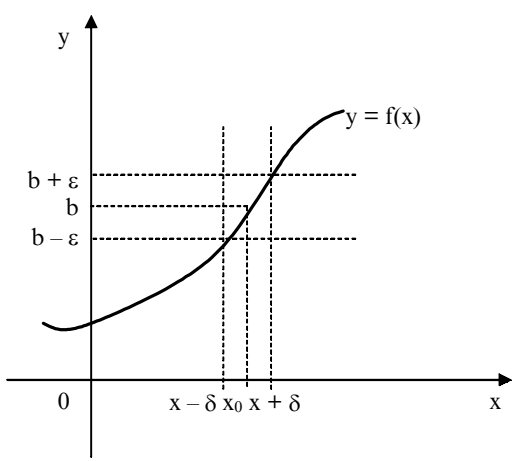


Рис. 3.3

Сформулируем определение *предела функции*.

Число b называется пределом функции $y = f(x)$ при $x \to x_0$, если для $\forall \varepsilon > 0$ можно найти такое $\delta > 0$, что для всех $x \neq x_0$, удовлетворяющих условию $|x - x_0| < \delta$, будет справедливо неравенство: $|f(x) - b| < \varepsilon$ [2, 20]. Заметим, что функция не обязательно должна быть определена в предельной точке x_0, она должна быть определена лишь в некоторой окрестности этой точки.

Тот факт, что b — предел функции $y = f(x)$ при $x \to x_0$, записывается так: $\lim\limits_{x \to x_0} f(x) = b$. Данное нами определение иллюстрируется рис. 3.3.

Используя приведенное определение предела, докажем, что

$$\lim_{x \to 3} \frac{x^2 - 9}{x - 3} = 6.$$

На основании определения имеем:

$$\left| \frac{x^2 - 9}{x - 3} - 6 \right| < \varepsilon \Rightarrow \left| \frac{(x - 3)(x + 3)}{x - 3} - 6 \right| < \varepsilon \Rightarrow |x - 3| < \varepsilon = \delta. \qquad (3.2)$$

Таким образом, мы доказали, что исходная функция будет отличаться от 6 меньше, чем на e, если будет выполняться неравенство (3.2). В данном случае $\varepsilon = \delta$.

Приведенное определение не дает способа вычисления пределов. Ниже мы рассмотрим некоторые из этих методов.

Дадим понятие о левых и правых пределах функции $y = f(x)$ и точках ее разрыва.

Если $f(x) \to b_1$ при $x \to x_0$ так, что x принимает только значения, меньшие x_0, то пишут $\lim\limits_{x \to x_0 -} f(x) = b_1$ и называют b_1 *левым пределом*.

Аналогично, если $f(x) \to b_2$ при $x \to x_0$ так, что x принимает только значения, большие x_0, то пишут $\lim\limits_{x \to x_0 +} f(x) = b_2$ и называют b_2 *правым пределом* [2, 20].

Геометрическая иллюстрация левого и правого пределов дана на рис. 3.4.

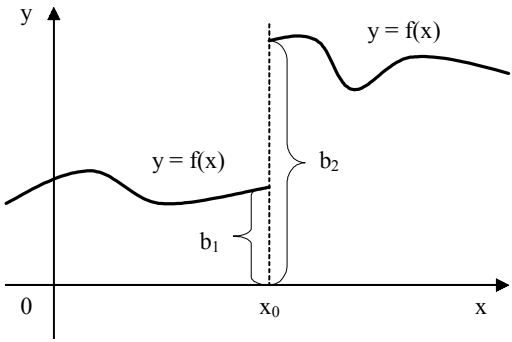


Рис. 3.4

Из рис. 3.4 следует, что в точке x_0 функция $y = f(x)$ имеет разрыв. Он носит название разрыва первого рода (в точке разрыва первого рода левый и правый пределы не равны ($b_1 \neq b_2$) и конечны). Все остальные точки разрыва называются точками разрыва второго рода [2, 20]. Примерами разрывов второго рода являются бесконечные разрывы (см. рис. 3.5).

$$\lim_{x \to 0+} \frac{1}{x} = +\infty$$

$$\lim_{x \to 0-} \frac{1}{x} = -\infty$$

В точке $x = 0$ функция $y = \frac{1}{x}$ имеет бесконечный разрыв.

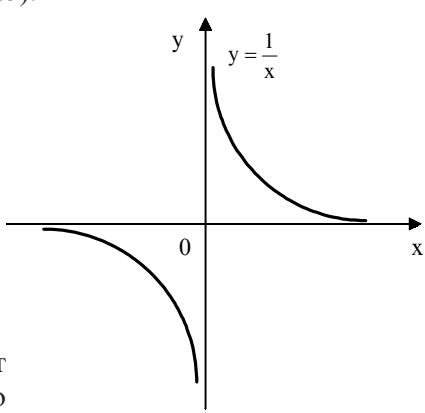


Рис. 3.5

Предположим, что аргумент функции $y = f(x)$ неограниченно возрастает ($x \to \infty$), т. е. является бесконечно большим аргументом.

Может оказаться, что при этом функция f(x) стремится к некоторому пределу b (см. рис. 3.6).

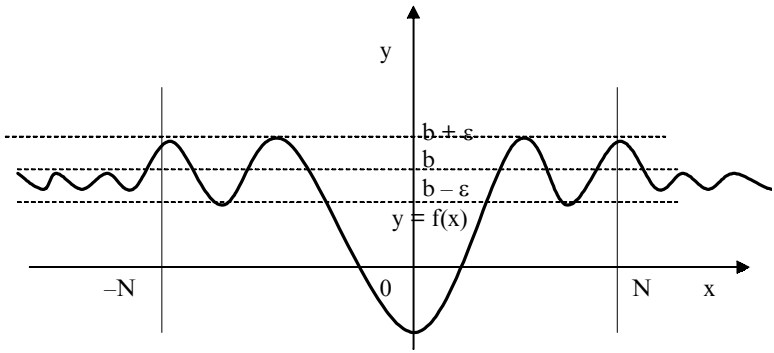


Рис. 3.6

Функция y = f(x) стремится к пределу b при $x \to \infty$, если для $\forall \varepsilon > 0$ можно найти такое N > 0, что для всех значений x, удовлетворяющих неравенству |x| > N, будет выполняться условие $|f(x) - b| < \varepsilon$ [2, 20, 22].

Теперь рассмотрим случай стремления функции y = f(x) к бесконечности при $x \to x_0$.

Функция y = f(x) стремится к бесконечности при $x \to x_0$, если для $\forall M > 0$ можно найти такое $\delta > 0$, что для всех значений $x \neq x_0$, удовлетворяющих условию $|x - x_0| < \delta$, выполняется неравенство |f(x)| > M [2, 20].

Это определение иллюстрируется рис. 3.7.

Напомним, что функция y = f(x) называется ограниченной в данной области изменения аргумента, если существует N > 0 такое, что для всех значений x, принадлежащих рассматриваемой области, будет выполняться неравенство $|f(x)| \le N$. Если такого числа N нет, то y = f(x) является неограниченной в данной области.

Например, функция y = sinx является ограниченной на своей области определения $x \in (-\infty; +\infty)$ (см. рис. 3.8).

$|\sin x| \le 1$, т. е. N = 1.

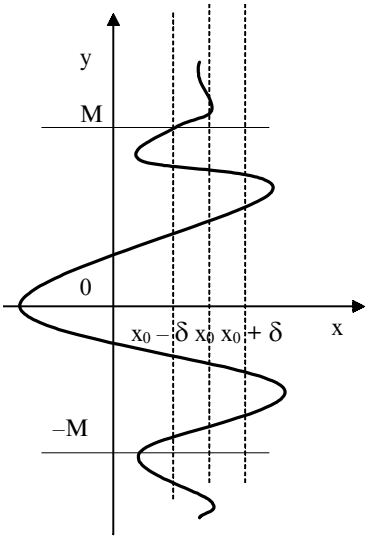


Рис. 3.7

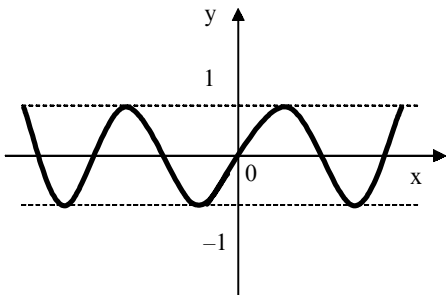


Рис. 3.8

Дадим определение бесконечно малой величины.

Функция б(x) называется *бесконечно малой* при $x \to x_0$ или $x \to \infty$, если $\lim\limits_{x \to x_0} \alpha(x) = 0$ или $\lim\limits_{x \to \infty} \alpha(x) = 0$.

Например, функция $y = (x - 3)^3$ при $x \to 3$ есть бесконечно малая величина, так как $\lim\limits_{x \to 3} (x - 3)^3 = 0$.

Постоянное очень малое число не является бесконечно малой величиной. Единственное число, которое рассматривается в качестве бесконечно малой величины, это ноль. Связь бесконечно малых и бесконечно больших величин можно проследить из теоремы 3.1: если $\alpha(x)$ — бесконечно малая величина, то $\dfrac{1}{\alpha(x)}$ — бесконечно большая величина, и наоборот [2].

Сравнение бесконечно малых

Если $\lim\limits_{x\to x_0}\dfrac{\alpha(x)}{\beta(x)}=0$, то $\alpha(x)$ есть бесконечно малая более высокого порядка, чем $\beta(x)$.

Например, $\lim\limits_{x\to 0}\dfrac{x^3}{x}=0$.

Если $\lim\limits_{x\to x_0}\dfrac{\alpha(x)}{\beta(x)}=\infty$, то $\alpha(x)$ есть бесконечно малая более низкого порядка, чем $\beta(x)$.

Например, $\lim\limits_{x\to 0}\dfrac{x^2}{x^5}=\infty$.

Если $\lim\limits_{x\to 0}\dfrac{\alpha(x)}{\beta(x)}=C$, где $C\in R$, то $\alpha(x)$ и $\beta(x)$ — бесконечно малые одного порядка.

Например, $\lim\limits_{x\to 1}\dfrac{3x}{x}=3$.

Если $\lim\limits_{x\to x_0}\dfrac{\alpha(x)}{\beta(x)}=1$, то $\alpha(x)$ и $\beta(x)$ есть эквивалентные бесконечно малые.

Например, $\lim\limits_{x\to 0}\dfrac{tgx}{x}=1$.

Теперь приведем *основные свойства пределов*, которые будем использовать при их вычислении [2, 9, 16].

1. Предел алгебраической суммы конечного числа функций равен алгебраической сумме пределов от этих функций, т. е.

$$\lim_{x\to x_0}(f_1(x)\pm f_2(x)\pm\ldots\pm f_n(x))=\lim_{x\to x_0}f_1(x)\pm\lim_{x\to x_0}f_2(x)\pm\ldots\pm$$

$$\times\lim_{x\to x_0}f_n(x).$$

2. Предел постоянной величины равен самой постоянной величине, т. е.

$$\lim_{x\to 0} C = C, \text{ где } C \in R.$$

3. Предел произведения конечного числа функций равен произведению пределов этих функций, т. е.

$$\lim_{x\to x_0} (f_1(x)\cdot f_2(x)\cdot \ldots \cdot f_n(x)) = \lim_{x\to x_0} f_1(x)\cdot \lim_{x\to x_0} f_2(x)\cdot \ldots \cdot \lim_{x\to x_0} f_n(x).$$

Следствие. Постоянный множитель можно выносить за знак предела, т. е.

$$\lim_{x\to x_0} Cf(x) = C \lim_{x\to x_0} f(x), \text{ где } C \in R.$$

4. Предел частного двух функций равен частному их пределов, если предел знаменателя не равен нулю, т. е.

$$\lim_{x\to x_0} \frac{f_1(x)}{f_2(x)} = \frac{\lim\limits_{x\to x_0} f_1(x)}{\lim\limits_{x\to x_0} f_2(x)}, \quad \lim_{x\to x_0} f_2(x) \neq 0.$$

5. Предел целой положительной степени функции равен той же степени предела этой функции, т. е.

$$\lim_{x\to x_0} [f(x)]^n = [\lim_{x\to x_0} f(x)]^n, \text{ где } n \in N.$$

6. Предел целой положительной n-й степени корня функции равен корню n-й положительной степени предела этой функции, т. е.

$$\lim_{x\to x_0} \sqrt[n]{f(x)} = \sqrt[n]{\lim_{x\to x_0} f(x)}, \text{ где } n \in Z^+.$$

Приведем два замечательных предела, которые можно использовать при решении пределов:

1) $\lim\limits_{x\to 0} \dfrac{\sin x}{x} = 1;$

2) $\lim\limits_{x\to\infty} \left(1 + \dfrac{1}{x}\right)^x = e$ (основание натуральных логарифмов).

Стремление к бесконечности всегда можно заменить стремлением к нулю и наоборот. Заменим во втором замечательном пределе $\frac{1}{x} = y$, а $x = \frac{1}{y}$. Тогда согласно теореме 3.1 при $x \to \infty$ $y \to 0$ и второй замечательный предел принимает вид:

$$\lim_{y \to 0} (1 + y)^{\frac{1}{y}} = e.$$

Кратко рассмотрим понятие непрерывности функции. Для этого напомним, что приращением функции $y = f(x)$ в точке x_0 называется величина $\Delta y = \Delta f(x) = f(x_0 + \Delta x) - f(x_0)$ [2, 16], где Δx есть приращение аргумента (см. рис. 3.9).

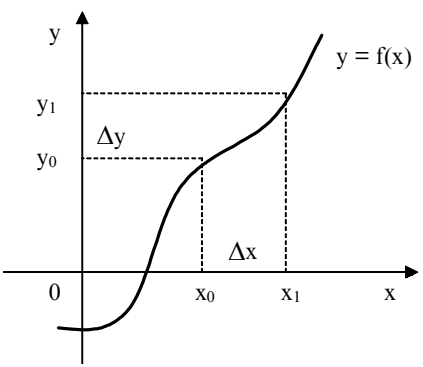


Рис. 3.9

Функция $y = f(x)$ называется *непрерывной* в точке x_0, если она определена в некоторой окрестности этой точки и если выполняется следующее равенство:

$$\lim_{\Delta x \to 0} \Delta y = 0. \qquad (3.3)$$

Докажем, например, что функция $y = \cos x$ непрерывна в любой точке x_0 своей области определения.

Согласно определению непрерывности функции в точке x_0 получим:

$$\Delta y = \cos(x_0 + \Delta x) - \cos x_0 = -2 \sin \frac{x_0 + \Delta x - x_0}{2} \times \sin \frac{x_0 + \Delta x + x_0}{2} =$$

$$= -2 \sin \frac{\Delta x}{2} \sin \frac{2x_0 + \Delta x}{2};$$

$$\lim_{\Delta x \to 0} \Delta y = \lim_{\Delta x \to 0} \left(-2 \sin \frac{\Delta x}{2} \sin \frac{2x_0 + \Delta x}{2} \right) =$$

$$= -2 \lim_{\Delta x \to 0} \sin \frac{\Delta x}{2} \lim_{\Delta x \to 0} \sin \frac{2x_0 + \Delta x}{2} = 0.$$

Пользуясь выражением для приращения функции, формулу (3.3) можно переписать так:

$$\lim_{\Delta x \to 0} [f(x_0 + \Delta x) - f(x_0)] = 0, \text{ или } \lim_{\Delta x \to 0} f(x_0 + \Delta x) = f(x_0).$$

Обозначим $x_0 + \Delta x = x$, тогда x будет стремиться к x_0 при $\Delta x \to 0$, и окончательно получим:

$$\lim_{x \to x_0} f(x) = f(x_0). \qquad (3.4)$$

Таким образом, функция $y = f(x)$ непрерывна в точке x_0, если она определена в некоторой окрестности этой точки и предел функции при стремлении аргумента к x_0 существует и равен значению функции в этой точке [2].

Заметим, что функция является непрерывной на некотором интервале, если она непрерывна в каждой его точке, а все основные элементарные функции непрерывны на тех интервалах, в которых они определены.

Приведем основные *свойства непрерывных функций* [9].

1. Алгебраическая сумма конечного числа непрерывных функций есть функция непрерывная.

2. Произведение конечного числа непрерывных функций есть функция непрерывная.

3. Частное двух непрерывных функций есть функция непрерывная в тех точках, в которых делитель не равен нулю.

4. Если $y = f(u)$ и $u = \varphi(x)$ — непрерывные функции своих аргументов, то сложная функция $y = f(\varphi(x))$ также непрерывна.

5. Если функция $y = f(x)$ непрерывна и имеет обратную функцию $x = \varphi(y)$, то последняя также непрерывна.

Если функция $y = f(x)$ непрерывна, то в формуле (3.4) можно поменять местами знаки функции и предела [16], т. е.

$$\lim_{x \to x_0} f(x) = f \lim_{x \to x_0} x. \qquad (3.5)$$

Формула (3.5) означает, что если функция непрерывна, то для отыскания предела надо вместо аргумента x подставить предельное значение x_0. Это правило неприменимо в том случае, когда

при подстановке предельного значения мы получаем неопределенности вида: $\frac{0}{0}$; $\frac{\infty}{\infty}$; $(0\cdot\infty)$; $(\infty-\infty)$; 1^{∞}; 0^{0}; ∞^{0} и др.

Теперь приведем конкретные примеры вычисления некоторых пределов.

Пример 3.2.1

$$\lim_{x\to 4}\frac{x^2+2}{x-3}=\frac{4^2+2}{4-3}=\frac{18}{1}=18.$$

Пример 3.2.2

$$\lim_{x\to 1}\frac{x^3-1}{x-1}=\lim_{x\to 1}\frac{(x-1)(x^2+x+1)}{x-1}=\lim_{x\to 1}(x^2+x+1)=3.$$

Пример 3.2.3

$$\lim_{x\to\infty}\frac{6x^3-3x^2+2x-4}{7x^5+6x^4-5x^2+7}.$$

Если подставить предельное значение, то получим неопределенность $\left(\frac{\infty}{\infty}\right)$, поэтому для решения подобных примеров используют следующий прием: делим числитель и знаменатель на x в максимальной степени, в данном случае на x^5. Тогда получим:

$$\lim_{x\to\infty}\frac{\frac{6}{x^2}-\frac{3}{x^3}+\frac{2}{x^4}-\frac{4}{x^5}}{7+\frac{6}{x}-\frac{5}{x^3}+\frac{7}{x^5}}=\frac{0-0+0-0}{7+0-0-0}=0.$$

Пример 3.2.4

$$\lim_{x\to 0}\frac{\sqrt{2+x}-\sqrt{2-x}}{5x}=\lim_{x\to 0}\left(\frac{\sqrt{2+x}-\sqrt{2-x}}{5x}\right)\left(\frac{\sqrt{2+x}+\sqrt{2-x}}{\sqrt{2+x}+\sqrt{2-x}}\right)=$$

$$= \lim_{x\to 0}\left[\frac{2+x-2+x}{5x(\sqrt{2+x}+\sqrt{2+x})}\right] = \lim_{x\to 0}\frac{2}{5(\sqrt{2+x}+\sqrt{2+x})} = \frac{\sqrt{2}}{10}.$$

Пример 3.2.5

$$\lim_{x\to\infty}\left(1+\frac{3}{x}\right)^{2x} = \lim_{x\to\infty}\left(1+\frac{3}{x}\right)^{\frac{x}{3}\cdot 2x\cdot\frac{3}{x}} = \left[\lim_{x\to\infty}\left(1+\frac{3}{x}\right)^{\frac{x}{3}}\right]^6 = e^6.$$

(Предел в квадратных скобках — это второй замечательный предел).

Пример 3.2.6

$$\lim_{x\to 0}\frac{1-\cos x}{x^2} = \lim_{x\to 0}\left[\frac{1-\cos x}{x^2}\times\frac{1+\cos x}{1+\cos x}\right] =$$

$$= \lim_{x\to 0}\frac{1-\cos^2 x}{x^2(1+\cos x)} = \lim_{x\to 0}\frac{\sin^2 x}{x^2(1+\cos x)} =$$

$$= \left[\lim_{x\to 0}\frac{\sin x}{x}\right]^2\cdot\lim_{x\to 0}\frac{1}{1+\cos x} = \frac{1}{2}.$$

Пример 3.2.7

$$\lim_{x\to 0}\left[\frac{\log_5(1+x)}{x}\right] = \lim_{x\to 0}\left[\frac{1}{x}\log_5(1+x)\right] =$$

$$= \lim_{x\to 0}\log_5(1+x)^{\frac{1}{x}} = \log_5\lim_{x\to 0}(1+x)^{\frac{1}{x}} = \log_5 e.$$

Так как логарифмическая функция непрерывна, то можно воспользоваться формулой (3.5).

Пример 3.2.8

$$\lim_{x\to 0}\frac{\sin 8x}{x}.$$

Данный предел можно свести к первому замечательному пределу путем замены переменной, т. е. $8x = y \Rightarrow x = \frac{y}{8}$, при $x \to 0$ $y \to 0$. Тогда получим:

$$\lim_{x\to 0} \frac{\sin 8x}{x} = \lim_{y\to 0} 8\frac{\sin y}{y} = 8\lim_{y\to 0} \frac{\sin y}{y} = 8.$$

Пример 3.2.9. Подставляем предельное значение (∞) вместо x и получаем:

$$\lim_{x\to\infty}\left(\frac{2}{3}+\frac{5}{x}\right)^{x^3} = \left(\frac{2}{3}\right)^{\infty} = 0.$$

Пример 3.2.10

$$\lim_{x\to\infty} \frac{6x+\sin x}{x-\cos x} = \lim_{x\to\infty} \frac{6+\frac{\sin x}{x}}{1-\frac{\cos x}{x}} = \frac{\lim\limits_{x\to\infty} 6+\lim\limits_{x\to\infty}\frac{\sin x}{x}}{\lim\limits_{x\to\infty} 1-\lim\limits_{x\to\infty}\frac{\cos x}{x}} = \frac{6+0}{1-0} = 6.$$

$$\lim_{x\to\infty} \frac{\sin x}{x} = 0; \quad \lim_{x\to\infty} \frac{\cos x}{x} = 0.$$

$\lim\limits_{x\to\infty} \sin x$ и $\lim\limits_{x\to\infty} \cos x$ не стремятся ни к какому пределу, они колеблются между –1 и +1, поэтому имеем:

$$\lim_{x\to\infty} \frac{\sin x}{x} = 0 \quad \lim_{x\to\infty} \frac{\cos x}{x} = 0.$$

Пример 3.2.11

$$\lim_{x\to\infty} \frac{2^{x+1}+3^{x+1}}{2^x+3^x} = \lim_{x\to\infty} \frac{\left(\frac{2}{3}\right)^{x+1}+1}{\left(\frac{2}{3}\right)^{x}\times\frac{1}{3}+\frac{1}{3}} = \frac{0+1}{0+\frac{1}{3}} = 3.$$

Пример 3.2.12

$$\lim_{x\to 0}\frac{\ln(3+x)-\ln 3}{x}=\lim_{x\to 0}\frac{1}{x}\ln\left(\frac{3+x}{3}\right)=$$

$$=\lim_{x\to 0}\ln\left(1+\frac{x}{3}\right)^{\frac{1}{x}}=\lim_{x\to 0}\ln\left(1+\frac{x}{3}\right)^{\frac{3}{3x}}=$$

$$=\ln\left[\lim_{x\to 0}\left(1+\frac{x}{3}\right)^{\frac{3}{x}}\right]^{\frac{1}{3}}=\ln e^{\frac{1}{3}}=\frac{1}{3}.$$

Пример 3.2.13. Для того чтобы убрать иррациональность в знаменателе, умножаем и делим исходное выражение на $(\sqrt{x+8}+4)$:

$$\lim_{x\to 8}\frac{64-x^2}{\sqrt{x+8}-4}=\lim_{x\to 8}\left[\frac{64-x^2}{\sqrt{x+8}-4}\times\frac{\sqrt{x+8}+4}{\sqrt{x+8}+4}\right]=$$

$$=\lim_{x\to 8}\frac{(64-x^2)(\sqrt{x+8}+4)}{x+8-16}=$$

$$=\lim_{x\to 8}\left(\frac{-(x-8)(x+8)(\sqrt{x+8}+4)}{x-8}\right)=$$

$$=-\lim_{x\to 8}(x+8)(\sqrt{x+8}+4)=-128.$$

Пример 3.2.14

$$\lim_{x\to 0}\frac{\arcsin x}{x}.$$

Необходимо свести данный предел к первому замечательному пределу. Для этого делаем замену переменной, т. е. $\arcsin x = y$, $x = \sin y$, при $x \to 0$ $y \to 0$. Тогда получим:

$$\lim_{x\to 0} \frac{\arcsin x}{x} = \lim_{y\to 0} \frac{y}{\sin y} = 1.$$

Задачи для самостоятельного решения

1. Найти области определения функций:

1.1. $y = \sqrt{2x^2 + 7x - 5}$;

1.2. $y = \log_5(6\cos x - 2)$;

1.3. $y = \dfrac{x^4 + 6x^3 - 8x + 2}{x^2 - 5}$;

1.4. $y = 5x^3 - 16x^2 + 2x - 7$.

2. Найти пределы функций:

2.1. $\lim\limits_{x\to\infty} \dfrac{5x^4 - 6x^3 + x^2 - 9}{7x^5 + 11x^4 + 2x^2 - 6x}$;

2.2. $\lim\limits_{x\to\infty} \dfrac{12x^3 - 4x^2 - 5x + 2}{4x^2 - x + 7x^3 + 18}$;

2.3. $\lim\limits_{x\to 0} \dfrac{3\sin 5x}{32x}$;

2.4. $\lim\limits_{x\to 0} \dfrac{2x^3 - x^2 + 4x}{x^2 + 8x}$;

2.5. $\lim\limits_{x\to 0} \dfrac{3\arcsin x}{11x}$;

2.6. $\lim\limits_{x\to 1} \dfrac{\sqrt{x} - \sqrt{2 - x}}{x - 1}$;

2.7. $\lim\limits_{x\to\infty}\left(1+\frac{7}{x}\right)^{x}$;

2.8. $\lim\limits_{x\to\infty}(-5x^2-7x+18)$;

2.9. $\lim\limits_{x\to 2}\frac{x^2-5x+6}{x^2-3x+2}$;

2.10. $\lim\limits_{x\to\infty}(x-\sqrt{x^2-3})$;

2.11. $\lim\limits_{x\to 4}\frac{16-x^2}{x^3-64}$;

2.12. $\lim\limits_{x\to 0}\frac{\sin 10\pi x}{\operatorname{tg}5x}$.

Вопросы для самопроверки

1. Что называется функцией одной независимой переменной?
2. Перечислить основные элементарные функции.
3. Какие функции называются элементарными? Приведите примеры.
4. Что такое предел функции $y = f(x)$ при $x \to x_0$?
5. Дайте определение правого и левого пределов функции $y = f(x)$.
6. Дайте определение предела последовательности.
7. Какая функция называется бесконечно большой величиной при $x \to x_0$ и $x \to +\infty$?
8. Какова связь между бесконечно большой и бесконечно малой величинами?
9. Сформулировать правила предельного перехода в случае арифметических действий.
10. В чем состоит правило предельного перехода для непрерывной функции?

4. Основы дифференциального исчисления

Дифференциальное исчисление — это раздел математического анализа, связанный в основном с понятиями производной и дифференциала функции.

4.1. Производная первого порядка. Дифференциал. Производная второго порядка

Производной функции $y = f(x)$ называется предел отношения приращения функции к приращению аргумента при произвольном стремлении последнего к нулю [2, 20, 22].

$$y' = f'(x) = \frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x} = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x},$$

т. е. производная функции $f(x)\left(f'(x), y', \frac{dy}{dx}\right)$ есть некоторая функция, полученная по определенным правилам из заданной функции.

Значение производной функции $y = f(x)$ в какой-то точке x_0 обозначают обычно так:

$f'(x_0)$ или $y'_{x = x_0}$.

Механический смысл производной — это предел средней скорости за бесконечно малый промежуток времени.

Геометрический смысл производной вытекает из следующей теоремы 4.1: если значение производной от функции $y = f(x)$ при $x = x_0$ равно $f'(x_0)$, то прямая, проведенная через точку $M_0(x_0, y_0)$ с угловым коэффициентом, равным $f'(x_0)$, является касательной к графику функции в точке M_0.

Геометрический смысл производной иллюстрируется на рис. 4.1.

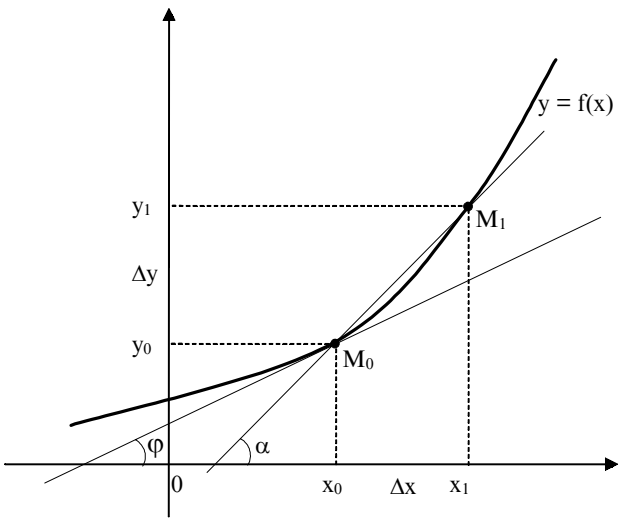


Рис. 4.1

Проведем через точки M_0 и M_1 секущую, тангенс угла α, образованного между секущей и положительным направлением оси 0X, равен:

$$\operatorname{tg}\alpha = \frac{\Delta y}{\Delta x}.$$

Будем перемещать точку M_1 по кривой в сторону точки M_0, т. е. устремим Δx к нулю. Предельным значением секущей будет касательная, проходящая через точку M_0. Тогда получим [2, 16, 17]:

$$y' = \lim_{\Delta x \to 0} \operatorname{tg}\alpha = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x} = \operatorname{tg}\varphi.$$

Установим связь между непрерывностью и дифференцируемостью функции. Она видна из следующей теоремы 4.2: если функция дифференцируема в некоторой точке, то в этой точке она непрерывна. Обратное утверждение неверно. В качестве примера возьмем функцию $y = |x|$. Ее график показан на рис. 4.2.

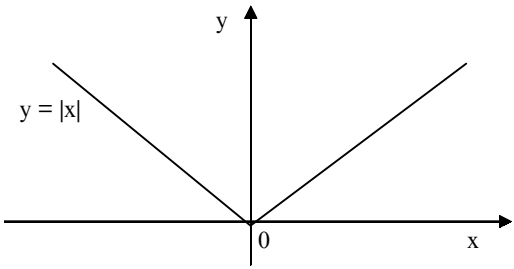


Рис. 4.2

Из него видно, что в точке x = 0 данная функция не имеет определенной касательной, а значит, не имеет в этой точке и производной.

Из определения производной следует способ ее вычисления.

Найдем производную функции $y = x^n$, где $n \in Z$, исходя из определения производной.

$$y = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x} = \lim_{\Delta x \to 0} \frac{(x + \Delta x)^n - x^n}{\Delta x} =$$

$$= \lim_{\Delta x \to 0} \frac{x^n + nx^{n-1} \times \Delta x + \frac{n(n-1)}{2} \times x^{n-2} \times (\Delta x)^2 + \ldots + (\Delta x)^n - x^n}{\Delta x} =$$

$$= \lim_{\Delta x \to 0} \frac{\Delta x \left(nx^{n-1} + \frac{n(n-1)}{2} \times x^{n-2} \times \Delta x + \ldots + (\Delta x)^{n-1} \right)}{\Delta x} =$$

$$= \lim_{\Delta x \to 0} nx^{n-1} + \lim_{\Delta x \to 0} \frac{n(n-1)}{2} \times x^{n-2} \times \Delta x + \ldots + \lim_{\Delta x \to 0} (\Delta x)^{n-1} = nx^{n-1}.$$

Итак, $(x^n)' = nx^{n-1}$, например $(x^8)' = 8x^7$.

Можно доказать, что полученная формула верна для всех $x \in R$ [22] .

Из приведенного примера видно, что использовать определение производной для ее вычисления дело достаточно трудоемкое.

Поэтому гораздо проще, используя определение производной, вывести производные основных элементарных функций и сформулировать правила дифференцирования алгебраической суммы, произведения, частного функций, сложной функции, обратной функции. По полученным формулам и правилам можно будет находить производные любых элементарных функций [2].

Производные основных элементарных функций

$(x^n)' = nx^{n-1}$; $(\ln x)' = \frac{1}{x}$; $(\log_a x)' = \frac{1}{x \ln a}$;

$(a^x)' = a^x \ln a$; $(e^x)' = e^x$; $(\sin x)' = \cos x$; $(\cos x)' = -\sin x$;

$(\operatorname{tg} x)' = \frac{1}{\cos^2 x}$; $(\operatorname{ctg} x)' = -\frac{1}{\sin^2 x}$; $(\arcsin x)' = \frac{1}{\sqrt{1-x^2}}$;

$(\arccos x)' = -\frac{1}{\sqrt{1-x^2}}$; $(\operatorname{arctg} x)' = \frac{1}{1+x^2}$; $(\operatorname{arcctg} x)' = -\frac{1}{1+x^2}$.

Дополним таблицу производных производными от гиперболических и обратных гиперболических функций, которые не являются основными элементарными функциями, но часто используются в различных приложениях [2, 22].

К гиперболическим функциям относятся гиперболические синус (sh x), косинус (ch x) и тангенс (th x), которые находятся по формулам:

$$\operatorname{sh} x = \frac{e^x - e^{-x}}{2}; \quad \operatorname{ch} x = \frac{e^x + e^{-x}}{2}; \quad \operatorname{th} x = \frac{\operatorname{sh} x}{\operatorname{ch} x} = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$

Все эти функции определены на множестве действительных чисел (R) и связаны между собой следующими соотношениями:

$\operatorname{ch}^2 x - \operatorname{sh}^2 x = 1$; $\operatorname{ch} 2x = \operatorname{ch}^2 x + \operatorname{sh}^2 x$; $\operatorname{sh} 2x = 2 \operatorname{sh} x \operatorname{ch} x$; $\operatorname{ch}^2 x = \frac{1}{1 - \operatorname{th}^2 x}$;

$$\operatorname{sh}^2 x = \frac{\operatorname{tg}^2 x}{1 - \operatorname{th}^2 x}.$$

Функции, обратные к shx, chx, thx являются обратными гиперболическими функциями и обозначаются:

Arshx (ареа-синус гиперболический), Archx (ареа-косинус гиперболический), Arth (ареа-тангенс гиперболический).

$$\mathrm{Arshx} = \ln\left(x + \sqrt{x^2 + 1}\right);$$

$$\mathrm{Archx} = \ln\left(x \pm \sqrt{x^2 - 1}\right);$$

$$\mathrm{Arthx} = \frac{1}{2}\ln\frac{1+x}{1-x}.$$

Производные гиперболических и обратных гиперболических функций находятся по формулам

$$(shx)' = chx;\ (chx)' = shx;\ (thx)' = \frac{1}{ch^2 x};\ (Arshx)' = \frac{1}{\sqrt{x^2+1}};$$

$$(Archx)' = \pm\frac{1}{\sqrt{x^2-1}};\ (Arthx)' = \frac{1}{1-x^2}.$$

Правила дифференцирования

1. Алгебраической суммы функций.

$(f_1(x) \pm f_2(x) \pm \ldots \pm f_n(x))' = f_1'(x) \pm f_2'(x) \pm \ldots \pm f_n'(x)$.

2. Произведения функций.

$(f_1(x) \times f_2(x))' = f_1'(x) \times f_2'(x) + f_1(x) \times f_2'(x)$.

Исходя из этого правила, для трех функций получим:

$(f_1(x) \times f_2(x) \times f_3(x))' = (f_1(x) \times f_2(x))' \times f_3(x) + f_1(x) \times f_2(x) \times f_3'(x) =$
$= f_1'(x) \times f_2(x) \times f_3(x) + f_2'(x) \times f_1(x) \times f_3(x) + f_3'(x) \times f_1(x) \times f_2(x)$.

3. Частного двух функций.

$$\left(\frac{f_1(x)}{f_2(x)}\right)' = \frac{f_1'(x) \times f_2(x) - f_2'(x) \times f_1(x)}{[f_2(x)]^2};\quad f_2(x) \neq 0.$$

4. Сложной функции.

Сформулируем ТЕОРЕМУ 4.3: *производная сложной функции равна производной заданной функции по промежуточному аргументу, умноженной на производную этого аргумента по независимой переменной*, т. е. если y = f(u), а u = j(x)и y = f(j(x)), то согласно данной теореме:

$$y' = f'(u) \times u'(x) = \frac{dy}{du} \times \frac{du}{dx}.$$

Аналогично выводится формула при любом числе промежуточных аргументов, т. е. производная сложной функции равна произведению производных от функций ее составляющих.

Например, найдем производную функции

$y = \cos^2 4x.$

$y' = 2\cos 4x(-\sin 4x)4 = -8\cos 4x \times \sin 4x = -4\sin 8x.$

5. Производная обратной функции находится по формуле

$$y'(x) = \frac{1}{x'(y)} \text{ или } x'(y) = \frac{1}{y'(x)},$$ т. е. производные от взаимно обратных функций обратны по величине. В качестве примера найдем производную функции

$$y = \arcsin x \Rightarrow x = \sin y,\ y \in \left[-\frac{\pi}{2}; \frac{\pi}{2}\right];$$

$$x_y' = \cos y \Rightarrow y_x' = \frac{1}{\cos y} = \frac{1}{\sqrt{1-\sin^2 y}} = \frac{1}{\sqrt{1-x^2}}.$$

Примеры нахождения производных

Пример 4.1. $y = \log_{2x} \sin^2 x = \frac{\ln \sin^2 x}{\ln 2x}$

Прежде чем найти производную от заданной функции, мы перешли к другому основанию и теперь найдем производную исходной функции по правилу производной частного.

$$y' = \frac{\frac{1}{\sin^2 x} 2\sin x \cos x \ln 2x - \frac{1}{2x} 2\ln \sin^2 x}{(\ln 2x)^2} =$$

$$= \frac{2\operatorname{ctg} x \ln 2x - x^{-1} \ln \sin^2 x}{(\ln 2x)^2}.$$

Пример 4.1.2. $y = x^{\sin x}$

Данная функция называется сложной показательной функцией. Чтобы найти производную от такой функции прологарифмируем ее левую и правую части, а затем продифференцируем полученные выражения, помня, что y есть функция от x [2].

$\ln y = \sin x \ln x;$

$$\frac{1}{y} y' = \cos x \ln x + \frac{1}{x} \sin x;$$

$$y' = y\left(\cos x \ln x + \frac{\sin x}{x}\right);$$

$$y' = x^{\sin x}\left(\cos x \ln x + \frac{\sin x}{x}\right).$$

Дадим понятие дифференциала функции.

Если задана непрерывная функция $y = f(x)$, имеющая производную $f'(x) = \lim\limits_{\Delta x \to 0} \frac{\Delta y}{\Delta x}$, то имеем:

$\frac{\Delta y}{\Delta x} = f'(x) + \alpha(x)$, где $\alpha(x)$ — бесконечно малая величина при $\Delta x \to 0$.

Далее получаем:

$\Delta y = f'(x)\Delta x + \alpha(x)\Delta x.$

Дифференциалом (от латинского слова *differentia* — разность) функции называется главная часть приращения этой функции, линейная относительно приращения аргумента x, т. е.

$dy = f'(x) \times \Delta x.$

Геометрический смысл дифференциала виден из рис. 4.3.

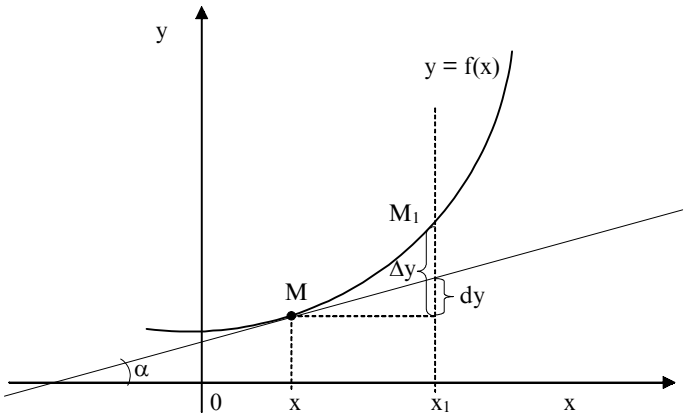


Рис. 4.3

Дифференциал функции геометрически изображается приращением ординаты касательной, проведенной в точке M(x, y) при данных значениях x, Δx [16, 22].

Рассмотрим функцию y = x.

Для нее получим, что dy = Δx, а так как y можно заменить на x по условию, то имеем dx = Δx.

Следовательно, дифференциал функции равен dy = y'dx. Отсюда следует формула

$$y' = \frac{dy}{dx}.$$

Дадим понятие производной второго порядка. Предположим, что нам задана функция y = f(x), имеющая производную y' = f'(x).

Производная от функции f'(x) также является функцией, и если она дифференцируема, то от нее можно взять производную. Она будет называться производной второго порядка:

$$y'' = f''(x) = (f'(x))' = \lim_{\Delta x \to 0} \frac{f'(x + \Delta x) - f'(x)}{\Delta x}.$$

Пример 4.3. $y = 3x^2 + \sin^3 x$.

$y' = 6x + 3\sin^2 x \cos x$

$y'' = 6 + 3(2\sin x \times \cos^2 x - \sin^3 x)$.

С помощью первой и второй производных можно исследовать функцию на экстремум (max, min), находить точки перегиба и участки выпуклости и вогнутости функции.

4.2 Некоторые сведения о функциях многих переменных. Понятие о частной производной

Ранее мы рассмотрели функции, которые зависели от одного независимого аргумента. Но в действительности чаще приходится иметь дело с функциями, которые зависят от двух, трех и большего числа независимых аргументов.

Например, площадь прямоугольника со сторонами a и b будет функцией двух независимых аргументов. Функция эта имеет вид $S_{пр} = a \times b$, где a и b могут быть любыми действительными положительными числами, так как и стороны прямоугольника, и его площадь не могут быть отрицательными величинами.

Положение какого-либо объекта на поверхности планеты определяется тремя координатами: широтой, долготой и высотой, т. е. является функцией трех независимых аргументов.

Положение космического аппарата (КА), движущегося по невозмущенной эллиптической орбите вокруг Земли, есть функция шести аргументов (трех координат (x, y, z) и трех составляющих скорости $(\dot{x}, \dot{y}, \dot{z})$). Эта функциональная зависимость имеет следующий вид:

$$F = \varphi(x, y, z, \dot{x}, \dot{y}, \dot{z}).$$

В гуманитарных науках, например в юриспруденции и экономике, жестко детерминированные функциональные связи встречаются нечасто. Там используются многофакторные статистические взаимосвязи вида:

$$Z = f(x_1, x_2, ..., x_n) + \Delta f(\Delta x_1, \Delta x_2, ..., \Delta x_n) + \varepsilon(y_1, y_2, ..., y_m),$$

где $x_1, x_2, ..., x_n$ — учтенные нами признаки, под влиянием которых меняется функция Z;

$\Delta x_1, \Delta x_2, ..., \Delta x_n$ — ошибки учтенных признаков;

$y_1, y_2, ..., y_m$ — неучтенные нами признаки, которые могут влиять на функцию Z.

Сначала рассмотрим функцию двух независимых аргументов x и y. Переменная величина Z называется функцией переменных величин x и y на множестве D, если каждой точке этого множества соответствует одно определенное значение величины Z [2, 3, 6, 20].

Множество D называется областью определения функции Z. Обычно она представляет собой часть плоскости x0y, ограниченной одной или несколькими линиями. Тот факт, что Z есть функция независимых аргументов x и y, записывают так:

$Z = f(x, y)$.

Функция двух аргументов может задаваться следующими способами:

1) аналитическим, т. е. приводится формула, при помощи которой по заданным значениям аргументов x и y находят значения функции Z. Например:

$$Z = \frac{1}{2}xy;\ Z = \sin^2(x + 2y);\ Z = \sqrt[3]{\sin^3 x - \ln y};\ Z^2 = x^2 + y^2.$$

2) табличным, т. е. для некоторого количества пар аргументов (x, y) приводятся значения функции (Z).

3) графическим.

x \ y	y_1	y_2	...	y_n
x_1	Z_{11}	Z_{12}	...	Z_{1n}
x_2	Z_{21}	Z_{22}	...	Z_{2n}
⋮	...	...	...	...
x_n	Zn_1	Z_{n2}	...	Z_{nm}

Графиком функции двух независимых аргументов в системе прямоугольных координат называется множество точек, абсциссы и ординаты которых являются значениями x и y, а аппликаты — соответствующими значениями Z. Графиком функции двух непрерывных аргументов обычно служит поверхность. Например, графиком функции $Z = x^2 + y^2$ является параболоид вращения (см. рис. 4.4).

Теперь дадим определение функции n независимых аргументов $x_1, x_2, ..., x_n$.

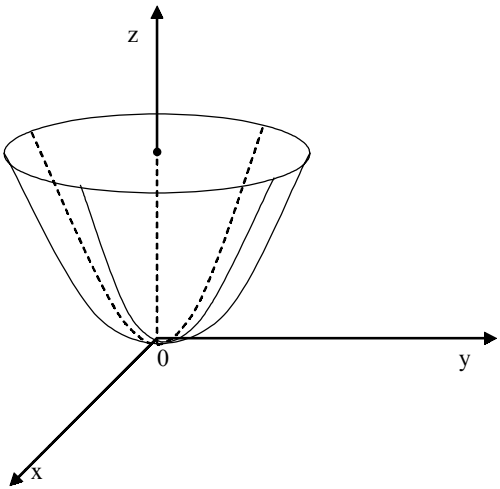


Рис. 4.4

Переменная величина W называется функцией переменных величин $x_1, x_2, ..., x_n$, если каждой рассматриваемой совокупности этих величин соответствует одно определенное значение W.

Тот факт, что W есть функция аргументов $x_1, x_2, ..., x_n$, записывают так:

$$W = f(x_1, x_2, ..., x_n).$$

Геометрическая иллюстрация функций от n независимых аргументов теряет наглядность при $n > 2$.

При исследовании поверхностей 2-го порядка часто применяют метод сечений, который заключается в том, что определение вида поверхности по ее уравнению производится путем изучения кривых, образованных при сечении этой поверхности плоскостями, параллельными координатным плоскостям.

Дадим определение предела функции двух независимых аргументов.

Число b называется пределом функции $Z = f(x, y)$ при $x \to x_0$ и $y \to y_0$, если для всех значений x и y достаточно мало отличающихся от x_0 и y_0 соответствующие значения функции f(x, y) как угодно мало отличаются от числа b [6, 22].

Тот факт, что b есть предел функции $Z = f(x, y)$ при $x \to x_0$ и $y \to y_0$, записывают так:

$$\lim_{\substack{x \to x_o \\ y \to y_o}} f(x, y) = b.$$

Теперь введем понятия частных производных по независимым аргументам.

Рассмотрим функцию двух независимых аргументов x и y. Z = f(x, y). Предположим, что y = const и рассмотрим эту функцию как функцию одного независимого аргумента x.

Если эта функция дифференцируема, то существует предел

$$\lim_{\Delta x \to 0} \frac{f(x + \Delta x, y) - f(x, y)}{\Delta x} = f'_x(x, y).$$

Нижний индекс (x) указывает на то, что производная берется по аргументу x.

Частной производной по x от функции Z = f(x, y) называется функция переменных величин x и y, которая получается при дифференцировании f(x, y) по x в предположении, что y = const.

Она обозначается так:

$$\frac{\partial Z}{\partial x}; \frac{\partial f(x, y)}{\partial x}; Z'_x.$$

Аргумент y считается постоянным только в процессе дифференцирования. После нахождения частной производной функция

$\frac{\partial Z}{\partial x}$ будет зависеть от двух аргументов x и y.

Аналогично определим частную производную по y от функции Z = f(x, y) при x = const как предел:

$$\lim_{\Delta y \to 0} \frac{f(x, y + \Delta y) - f(x, y)}{\Delta y} = f'_y(x, y).$$

Она обозначается так:

$$\frac{\partial Z}{\partial y}; \frac{\partial f(x, y)}{\partial y}; Z'_y.$$

При нахождении частных производных используются формулы и правила дифференцирования функции одного независимого аргумента. Рассмотрим конкретные примеры.

Пример 4.4

$Z = 5x^3 \cos y$.

$$\frac{\partial Z}{\partial x} = 15x^2 \cos y; \quad \frac{\partial Z}{\partial y} = -5x^3 \sin y.$$

Пример 4.5

$Z = 6x^4 tgy + 5x\ln y;$

$$\frac{\partial Z}{\partial x} = 24x^3 tgy + 5\ln y;$$

$$\frac{\partial Z}{\partial y} = \frac{6x^4}{\cos^2 y} + \frac{5x}{y}.$$

Аналогично можно определить частные производные от любого числа независимых аргументов.

Например, имеем функцию n независимых переменных

$W = f(x_1, x_2, ..., x_n)$.

Найдем частную производную по аргументу x_1:

$$\frac{\partial W}{\partial x_1} = \lim_{\Delta x_1 \to 0} \frac{f(x_1 + \Delta x_1, x_2, x_3, ..., x_n) - f(x_1, x_2, ..., x_n)}{\Delta x_1}.$$

Аналогично определим частную производную по аргументу x_2:

$$\frac{\partial W}{\partial x_2} = \lim_{\Delta x_2 \to 0} \frac{f(x_1, x_2 + \Delta x_2, x_3, ..., x_n) - f(x_1, x_2, ..., x_n)}{\Delta x_2}$$ и т. д.

Пример 4.6

$W = 2x_1 \cos x_2 \ln x_3.$

$$\frac{\partial W}{\partial x_1} = 2\cos x_2 \ln x_3; \quad \frac{\partial W}{\partial x_2} = -2x_1 \sin x_2 \ln x_3;$$

$$\frac{\partial W}{\partial x_3} = \frac{2x_1 \cos x_2}{x_3}.$$

Дифференцирование сложных функций

Пусть имеем функцию двух независимых аргументов $Z = f(u, v)$, причем аргументы являются функциями независимых переменных x и y, т. е. $u = \varphi(x, y)$; $v = \psi(x, y)$, т. е.

$Z = f[\varphi(x, y), \psi(x, y)]$.

В этом случае частные производные функции Z по аргументам x и y будут вычисляться по формулам [2, 22].

$$\frac{\partial Z}{\partial x} = \frac{\partial Z}{\partial u} \times \frac{\partial u}{\partial x} + \frac{\partial Z}{\partial v} \times \frac{\partial v}{\partial x};$$

$$\frac{\partial Z}{\partial y} = \frac{\partial Z}{\partial u} \times \frac{\partial u}{\partial y} + \frac{\partial Z}{\partial v} \times \frac{\partial v}{\partial y}.$$

Пример 4.7

$Z = e^{3xy} \times \sin(5x + y)$.

$u = 3xy$; $v = 5x + y$.

$z = e^u \times \sin v$.

$$\frac{\partial Z}{\partial x} = e^{3xy} \times 3y \times \sin(5x + y) + e^{3xy} \times \cos(5x + y) \times 5 = e^{3xy}(3y \times$$

$\times \sin(5x + y) + 5\cos(5x + y))$.

$$\frac{\partial Z}{\partial y} = e^{3xy} \times 3x \times \sin(5x + y) + e^{3xy} \times \cos(5x + y) = e^{3xy}(3x \times \sin(5x +$$

$+ y) + \cos(5x + y))$.

Пример 4.8

$Z = \sin(x^3y^2) \times \ln(3y + x^2)$.

$u = x^3y^2$.

$v = 3y + x^2$.

$z = \sin u \ln v$.

$$\frac{\partial Z}{\partial x} = \cos(x^3y^2)3x^2 \times y^2 \times \ln(3y + x^2) + \frac{\sin(x^3y^2)}{(3y + x^2)}2x.$$

$$\frac{\partial Z}{\partial y} = \cos(x^3y^2)2yx^3 \times \ln(3y + x^2) + \frac{\sin(x^3y^2)}{(3y + x^2)} \times 3.$$

4.3. Некоторые приложения дифференциального исчисления

4.3.1. Формула Тейлора

Пусть некоторая функция $y = f(x)$ имеет все производные до $(n + 1)$-го порядка включительно на некотором интервале, включающем точку x_0. Найдем многочлен $y = P_n(x)$ степени не выше n, значение которого в точке $x = x_0$ равно значению функции $y = f(x)$ в этой точке, а значение его производных до n-го порядка в точке $x = x_0$ равны значениям соответствующих производных от функции $y = f(x)$ в этой точке, т. е.

$$P_n(x_0) = f(x_0);\ P_n'(x_0) = f'(x_0);\ P_n''(x_0) = f''(x_0);$$

$$P_n'''(x_0) = f'''(x_0),\ \ldots,\ P_n^{(n)}(x_0) = f^{(n)}(x_0). \tag{4.1}$$

Искомый многочлен (более подробно см., например, [20, 22]) будет иметь вид:

$$P_n(x) = f(x_0) + \frac{(x - x_0)}{1} f'(x_0) + \frac{(x - x_0)^2}{1 \cdot 2} f''(x_0) + \\ + \frac{(x - x_0)^3}{1 \times 2 \times 3} f'''(x_0) + \ldots + \frac{(x - x_0)^n}{1 \times 2 \times 3 \times \ldots \times n} f^{(n)}(x_0). \tag{4.2}$$

Через $R_n(x)$ обозначим разность значений данной функции $y = f(x)$ и многочлена, находимого по формуле (4.2), т. е.

$R_n(x) = f(x) - P_n(x)$,

отсюда $f(x) = P_n(x) + R_n(x)$ или

$$f(x) = f(x_0) + \frac{(x - x_0)}{1!} f'(x_0) + \frac{(x - x_0)^2}{2!} f''(x_0) + \\ + \frac{(x - x_0)^3}{3!} f'''(x_0) + \ldots + \frac{(x - x_0)^n}{n!} f^{(n)}(x_0) + R_n(x), \tag{4.3}$$

где $R_n(x)$ — это остаточный член, который может быть записан в разных формах.

Мы приведем так называемую форму Лагранжа, которая имеет вид:

$$R_n(x) = \frac{(x - x_0)^{n+1}}{(n+1)!} f^{(n+1)}(\eta). \tag{4.4}$$

Здесь $\eta \in [x, x_0]$ и ее можно представить в виде $\eta = x_0 + \lambda(x - x_0)$, где $0 < \lambda < 1$. Тогда формула для остаточного члена примет вид:

$$R_n(x) = \frac{(x - x_0)^{n+1}}{(n+1)!} f^{(n+1)}[x_0 + \lambda(x - x_0)].$$

А формула

$$\begin{aligned} f(x) &= f(x_0) + \frac{(x - x_0)}{1!} f'(x_0) + \frac{(x - x_0)^2}{2!} f''(x_0) + \\ &+ \frac{(x - x_0)^3}{3!} f'''(x_0) + \ldots + \frac{(x - x_0)^n}{n!} f^{(n)}(x_0) + \\ &+ \frac{(x - x_0)^{n+1}}{(n+1)!} \cdot f^{(n+1)}[x_0 + \lambda(x - x_0)] \end{aligned} \tag{4.5}$$

называется формулой Тейлора для функции $y = f(x)$.

Если в формуле (4.5) принять $x_0 = 0$, то она примет вид:

$$\begin{aligned} f(x) &= f(0) + \frac{x}{1!} f'(0) + \frac{x^2}{2!} f''(0) + \frac{x^3}{3!} f'''(0) + \ldots + \\ &+ \frac{x^n}{n!} f^{(n)}(0) + \frac{x^{n+1}}{(n+1)!} \cdot f^{(n+1)}(\lambda x). \end{aligned} \tag{4.6}$$

Здесь $0 < \lambda < 1$, а формулу (4.6) часто называют формулой Маклорена. Теперь найдем разложение функции $y = e^x$ по формуле (4.6):

$f(x) = e^x$; $f(0) = 1$; $f'(x) = e^x$; $f'(0) = 1$; $f''(x) = e^x$;

$f''(0) = 1$, ..., $f^{(n)}(x) = e^x$; $f^{(n)}(0) = 1$.

Эти данные подставляем в формулу (4.6) и получаем:

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!} + \frac{x^{n+1}}{(n+1)!} e^{\lambda x}, \text{ где } 0 < \lambda < 1.$$

Если $|x| \le 1$, то, взяв $n = 8$, найдем оценку остаточного члена $R_n < \frac{1}{9!}3$. А если $x = 1$, то получим формулу для приближенного вычисления числа e [22], т. е.

$$e = 1 + 1 + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \ldots + \frac{1}{8!} \approx 2{,}71827.$$

Здесь верны первые четыре знака после запятой, так как ошибка не превосходит числа $\frac{3}{9!}$, или 10^{-5}. Заметим, что какое бы ни было x, остаточный член $R_n = \frac{x^{n+1}}{(n+1)!}e^{\lambda x} \to 0$ при $n \to \infty$, т. е. $\lim\limits_{n\to\infty} R_n = 0$. Поэтому при $\forall x$, взяв достаточное число членов разложения, по формуле (4.6) мы получим e^x с любой необходимой степенью точности.

4.3.2. Правило Лопиталя

Данное правило помогает раскрывать неопределенности вида: $\frac{0}{0}, \frac{\infty}{\infty}$, его суть выражается теоремой [2, 20, 22].

ТЕОРЕМА ЛОПИТАЛЯ 4.4. *Пусть функции* $f(x)$ *и* $\varphi(x)$ *при* $x \to x_0$ *или* $x \to \infty$ *совместно стремятся к нулю или к бесконечности. Если отношение производных этих функций имеет предел, то отношение самих функций тоже имеет предел, который равен пределу отношения производных*, т. е.

$$\lim_{x\to x_0} \frac{f(x)}{\varphi(x)} = \lim_{x\to x_0} \frac{f'(x)}{\varphi'(x)}; \tag{4.7}$$

$$\lim_{x\to \infty} \frac{f(x)}{\varphi(x)} = \lim_{x\to \infty} \frac{f'(x)}{\varphi'(x)}. \tag{4.8}$$

Теперь рассмотрим конкретные примеры применения этого правила.

Пример 4.9

$$\lim_{x\to 0}\frac{\ln(1+x)}{x}=\lim_{x\to 0}\frac{\frac{1}{1+x}}{1}=1.$$

Ранее мы сводили этот предел ко второму замечательному пределу и пользовались тем, что lnx является непрерывной функцией. Заметим, что простота взятия данного предела кажущаяся, так как дифференцирование функций само опирается на знание пределов [2].

Пример 4.10

$$\lim_{x\to 0}\frac{\sin 7x}{2\sin 3x}=\lim_{x\to 0}\frac{7\cos 7x}{6\cos 3x}=\frac{7}{6}.$$

Пример 4.11

$$\lim_{x\to 0}\frac{\arcsin x}{3x}=\lim_{x\to 0}\frac{\frac{1}{\sqrt{1-x^2}}}{3}=\frac{1}{3}.$$

Пример 4.12

$$\lim_{x\to 2}\frac{x^2-5x+6}{x^2-3x+2}=\lim_{x\to 2}\frac{2x-5}{2x-3}=-1.$$

Заметим, что если производные числителя и знаменателя одновременно стремятся к нулю или к бесконечности, можно применять правило Лопиталя еще раз, а в случае необходимости и далее.

Пример 4.13

$$\lim_{x\to\infty}\frac{5x^4-3x^3+2x-6}{7x^4+20x^2-7x+12}=\lim_{x\to\infty}\frac{20x^3-9x^2+2}{28x^3+40x-7}=$$

$$=\lim_{x\to\infty}\frac{60x^2-18x}{84x^2+40}=\lim_{x\to\infty}\frac{120x-18}{168x}=\lim_{x\to\infty}\frac{120}{168}=\frac{5}{7}.$$

Пример 4.14

$$\lim_{x\to\infty}\frac{\sin\frac{6}{x}}{\frac{1}{x}}=\lim_{x\to\infty}\frac{\cos\left(\frac{6}{x}\right)\left(-\frac{6}{x^2}\right)}{\left(-\frac{1}{x^2}\right)}=\lim_{x\to\infty}6\cos\frac{6}{x}=6.$$

$$\lim_{x\to+\infty}\frac{e^x}{7x}=\lim_{x\to\infty}\frac{e^x}{7}=+\infty.$$

Формулы (4.7) и (4.8) справедливы только в том случае, если предел, стоящий справа (конечный или бесконечный), существует. Приведем пример, когда отношение функций имеет предел, а отношение их производных не стремится ни к какому пределу.

Пример 4.15

$$\lim_{x\to\infty}\left(\frac{x+\sin x}{x}\right)=\lim_{x\to\infty}\left(1+\frac{\sin x}{x}\right)=1.$$

Предел производных равен:

$$\lim_{x\to\infty}\frac{(x+\sin x)'}{x'}=\lim_{x\to\infty}\frac{1+\cos x}{1},$$

при $x\to\infty$ — предел колеблется между 0 и 2, поэтому отношение производных функций не имеет предела, т. е. к данному примеру правило Лопиталя применить нельзя. Оно не является универсальным.

При помощи правила Лопиталя можно раскрывать другие неопределенности, например: $0\times\infty$; $\infty-\infty$; 1^∞; ∞^0; 0^0. Эти случаи сводятся к рассмотренным нами неопределенностям $\frac{0}{0}, \frac{\infty}{\infty}$.

Рассмотрим некоторые примеры.

Пример 4.16

$\lim\limits_{x\to0}x^3\ln5x$. Это случай $0\times\infty$.

Преобразуем данный предел к виду

$\lim\limits_{x\to 0}\dfrac{\ln 5x}{\dfrac{1}{x^3}}$, т. е. мы привели исходный предел к случаю $\dfrac{\infty}{\infty}$. Теперь можно применить правило Лопиталя:

$$\lim_{x\to 0}\frac{\ln 5x}{\dfrac{1}{x^3}}=\lim_{x\to 0}\frac{\dfrac{1}{5x}\cdot 5}{-\dfrac{3}{x^4}}=\lim_{x\to 0}\left(-\frac{x^3}{3}\right)=0.$$

Пример 4.17

$\lim\limits_{x\to 1}\left(\dfrac{3x}{x-1}-\dfrac{2}{\ln x}\right)$, т. е. имеем случай $\infty-\infty$. Исходный предел преобразуем к виду

$\lim\limits_{x\to 1}\dfrac{3x\ln x-2x+2}{(x-1)\ln x}$, видно что мы пришли к случаю $\dfrac{0}{0}$, поэтому применяем правило Лопиталя:

$$\lim_{x\to 1}\frac{3x\ln x-2x+2}{(x-1)\ln x}=\lim_{x\to 1}\frac{3\ln x+\dfrac{3x}{x}-2}{\ln x+\dfrac{x-1}{x}}=$$

$$=\lim_{x\to 1}\frac{3\ln x+1}{\ln x+1-\dfrac{1}{x}}=\frac{1}{0}=\infty.$$

Пример 4.18

$\lim\limits_{x\to 0}(\cos 2x)^{\frac{1}{x}}$, т. е. имеем случай 1^{∞}.

Рассмотрим предел

$\lim\limits_{x\to 0} \ln(\cos 2x)^{\frac{1}{x}} = \lim\limits_{x\to 0} \dfrac{\ln\cos 2x}{x}$, а это случай $\dfrac{0}{0}$, поэтому к последнему пределу применимо правило Лопиталя.

$$\lim_{x\to 0} \frac{\ln\cos 2x}{x} = \lim_{x\to 0} \frac{\frac{1}{\cos 2x}(-\sin 2x)\cdot 2}{1} = \lim_{x\to 0}(-2\operatorname{tg}2x) = 0.$$

Поэтому исходный предел

$$\lim_{x\to 0}(\cos 2x)^{\frac{1}{x}} = e^0 = 1.$$

Пример 4.19

$\lim\limits_{x\to 0} x^{7x}$, т. е. имеем случай 0^0.

Рассмотрим предел $\lim\limits_{x\to 0} \ln x^{7x} = \lim\limits_{x\to 0} 7x\ln x = \lim\limits_{x\to 0} \dfrac{\ln x}{\frac{1}{7x}}$, т. е. пришли к случаю $\dfrac{\infty}{\infty}$. Теперь к последнему примеру применяем правило Лопиталя.

$$\lim_{x\to 0} \frac{\ln x}{\frac{1}{7x}} = \lim_{x\to 0} \frac{\frac{1}{x}}{-\frac{1}{7x^2}} = \lim_{x\to 0}(-7x) = 0.$$

Поэтому исходный предел равен:

$$\lim_{x\to 0} x^{7x} = e^0 = 1.$$

4.3.3. Асимптоты

Прямая L называется асимптотой графика функции $y = f(x)$, если расстояние δ от переменной точки A функции до этой прямой при удалении точки A в бесконечность стремится к нулю (см. рис. 4.5) [2, 6, 22].

Различают вертикальные асимптоты (параллельные оси 0y) и наклонные.

Сначала рассмотрим вертикальные асимптоты.

Из определения асимптоты следует, что если

$\lim\limits_{x \to x_0 +} f(x) = \pm\infty$, или

$\lim\limits_{x \to x_0 -} f(x) = \pm\infty$, или

$\lim\limits_{x \to x_0} f(x) = \infty$, то прямая $x = x_0$ является асимптотой функции $y = f(x)$ и наоборот, если прямая $x = x_0$ есть асимптота кривой $y = f(x)$, то существуют указанные выше пределы. То есть для нахождения вертикальных асимптот надо найти такие значения $x = x_0$, при приближении к которым функция $y = f(x)$ стремится к бесконечности. Например, функция $y = tgx$ имеет бесконечное число вертикальных асимптот (см. рис. 4.6) $x = \pm\frac{\pi}{2}$; $x = \pm\frac{3\pi}{2}$; $x = \pm\frac{5\pi}{2}$...

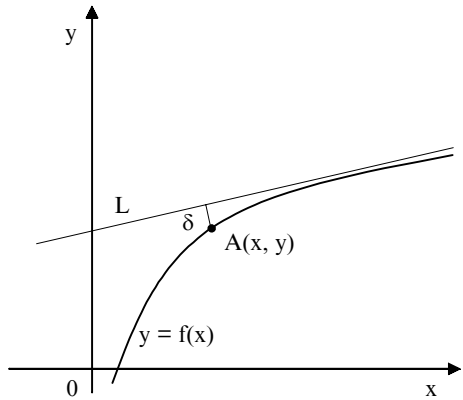


Рис. 4.5

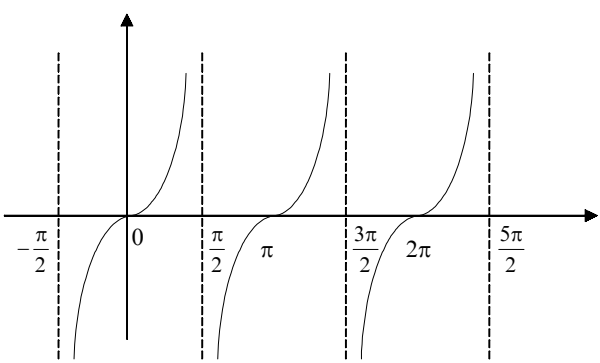


Рис. 4.6

Теперь рассмотрим наклонные асимптоты. Предположим, что функция y = f(x) имеет наклонную асимптоту y = ax + b (см. рис. 4.7).

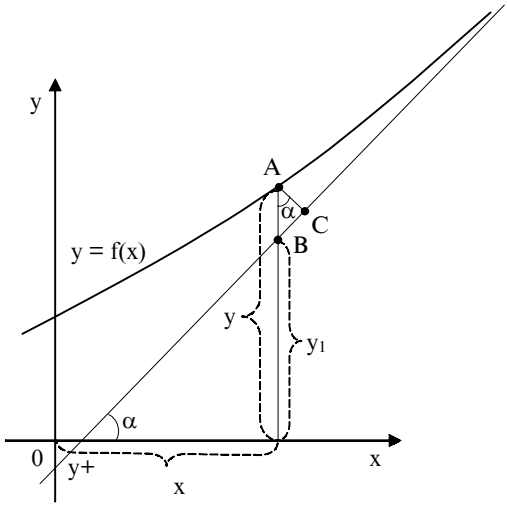


Рис. 4.7

Нам нужно найти коэффициенты a и b. Точка A(x, y) принадлежит графику функции y = f(x), а точка B(x, y_1) — асимптоте. Длина отрезка AC — это расстояние от точки A до асимптоты и по условию $\lim\limits_{x\to+\infty}(AC)=0$ (4.9). Обозначим через α угол наклона асимптоты к положительному направлению оси 0x и из Δ ABC найдем:

$$(AC)=(AB)\cos\alpha \Rightarrow (AB)=\frac{(AC)}{\cos\alpha}.$$

Так как $\alpha\neq\frac{\pi}{2}$ и α = const, то в силу (4.9) имеем $\lim\limits_{x\to+\infty}(AB)=0$ (4.10), так как (AB) = |y – y_1| = |f(x) – ax – b|, то (4.10) принимает следующий вид:

$$\lim_{x\to+\infty}(f(x)-ax-b)=0. \qquad (4.11)$$

Следовательно, если $y = ax + b$ есть асимптота, то выполняется (4.11) и наоборот, если при коэффициентах a и b выполняется (4.11), то прямая $y = ax + b$ является асимптотой. Теперь найдем коэффициенты a и b. Из (4.11) получаем (выносим x в формуле (4.11) за скобки):

$$\lim_{x\to+\infty} x\left(\frac{f(x)}{x} - a - \frac{b}{x}\right) = 0.$$

Так как $x \to +\infty$, то

$$\lim_{x\to+\infty}\left(\frac{f(x)}{x} - a - \frac{b}{x}\right) = 0,$$ а так как b есть число, то $\lim\limits_{x\to+\infty} \frac{b}{x} = 0$,

поэтому получаем $\lim\limits_{x\to+\infty}\left(\frac{f(x)}{x} - a\right) = 0$, или $a = \lim\limits_{x\to+\infty} \frac{f(x)}{x}$. (4.12)

Получив a из (4.11), находим b по формуле

$$b = \lim_{x\to+\infty} (f(x) - ax). \qquad (4.13)$$

Следовательно, если $y = ax + b$ является асимптотой, то a и b находятся по формулам (4.12) и (4.13). Если хотя бы один из пределов (4.12) или (4.13) не существует, то функция $y = f(x)$ наклонной асимптоты не имеет [22].

Все приведенные рассуждения справедливы и при $x \to -\infty$. Так как асимптотическое изменение функции может быть различным при стремлении x к положительной и отрицательной бесконечности, то надо раздельно рассматривать случаи $x \to -\infty$ и $x \to +\infty$. Если существует асимптота в первом случае, то ее называют левосторонней, а во втором случае — правосторонней.

Если при $x \to -\infty$ и $x \to +\infty$ пределы (4.12) и (4.13) совпадают, то левосторонняя и правосторонняя асимптоты являются частями одной и той же прямой.

Заметим, что если функция дробно-рациональная, то при нахождении a и b сразу можно рассматривать произвольное стремление к бесконечности.

Рассмотрим примеры нахождения наклонных асимптот.

Пример 4.20

$$y = \frac{x^3}{2x^2 - 1}.$$

Так как данная функция дробно-рациональная, то сразу рассматриваем произвольное стремление x к ∞.

$$a = \lim_{x\to\infty} \frac{x^3}{x(2x^2 - 1)} = \lim_{x\to\infty} \frac{x^3}{2x^3 - x} = \lim_{x\to\infty} \frac{1}{2 - \frac{1}{x^2}} = \frac{1}{2}.$$

$$b = \lim_{x\to\infty}\left(\frac{x^3}{2x^2 - 1} - \frac{1}{2}x \right) = \lim_{x\to\infty} \frac{2x^3 - x(2x^2 - 1)}{2(2x^2 - 1)} =$$

$$= \lim_{x\to\infty} \frac{2x^3 - 2x^3 + x}{4x^2 - 2} = \lim_{x\to\infty} \frac{\frac{1}{x}}{4 - \frac{2}{x^2}} = 0.$$

Поэтому получаем $y = \frac{1}{2}x$.

Пример 4.21
$y = 2x + \ln x$.

$$a = \lim_{x\to+\infty} \frac{2x + \ln x}{x} = \lim_{x\to+\infty} 2 + \lim_{x\to+\infty} \frac{\ln x}{x} = 2 + \lim_{x\to+\infty} \frac{\frac{1}{x}}{1} = 2$$

(по правилу Лопиталя);

$$b = \lim_{x\to+\infty} (2x + \ln x - 2x) = \lim_{x\to+\infty} \ln x = \infty.$$

Из последнего равенства следует, что исходная функция наклонной асимптоты не имеет.

4.3.4. Исследование функций с помощью производных первого и второго порядков и построение их графиков

Приведем ряд теорем, позволяющих находить: участки монотонности (возрастания, убывания) функции, экстремумы функции, участки выпуклости и вогнутости функции и точки перегиба.

Сначала сформулируем достаточный признак монотонности [2, 16]:

1) если $f'(x) > 0$ на некотором интервале, то $f(x)$ на этом интервале возрастает;

2) если $f'(x) < 0$ на некотором интервале, то $f(x)$ на этом интервале убывает;

3) если $f'(x) = 0$ на некотором интервале, то $f(x)$ на этом интервале постоянна.

Геометрическая интерпретация этого признака показана на рис. 4.8.

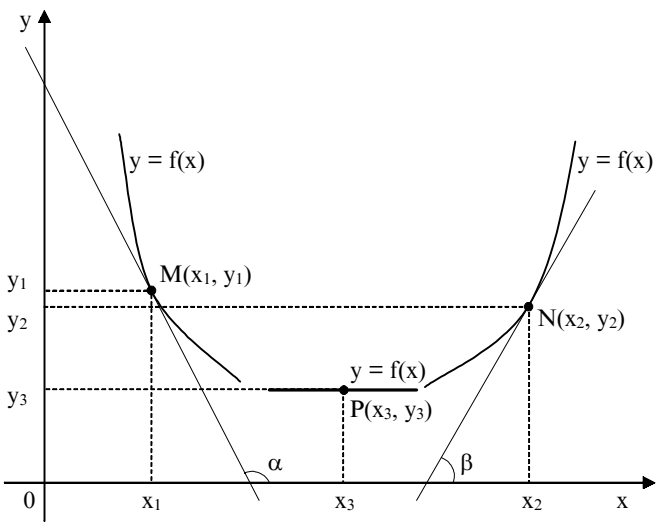


Рис. 4.8

$tg\alpha = y'(x = x_1) < 0$; $tg\beta = y'(x = x_2) > 0$; $y'(x = x_3) = 0$.

Важную роль в исследовании функций играют точки, отделяющие интервалы ее возрастания от интервалов ее убывания. Эти точки носят название экстремумов функции или ее локальных максимумов и минимумов. Слово “локальный” означает, что точка будет максимальной (минимальной) лишь на каком-то интервале.

Теперь приведем определение:

1) точка $M(x_1, y_1)$ есть точка локального максимума функции $y = f(x)$, если $f(x_1)$ — наибольшее значение функции $y = f(x)$ в некоторой окрестности точки $M(x_1, y_1)$;

2) точка $N(x_2, y_2)$ есть точка локального минимума функции $y = f(x)$, если $f(x_2)$ — наименьшее значение функции $y = f(x)$ в некоторой окрестности точки $N(x_2, y_2)$.

Функция на своей области определения может иметь несколько экстремумов. Наибольшее и наименьшее значения функции ее области определения обычно называют абсолютным максимумом и абсолютным минимумом.

Понятие экстремума функции иллюстрируется рис. 4.9.

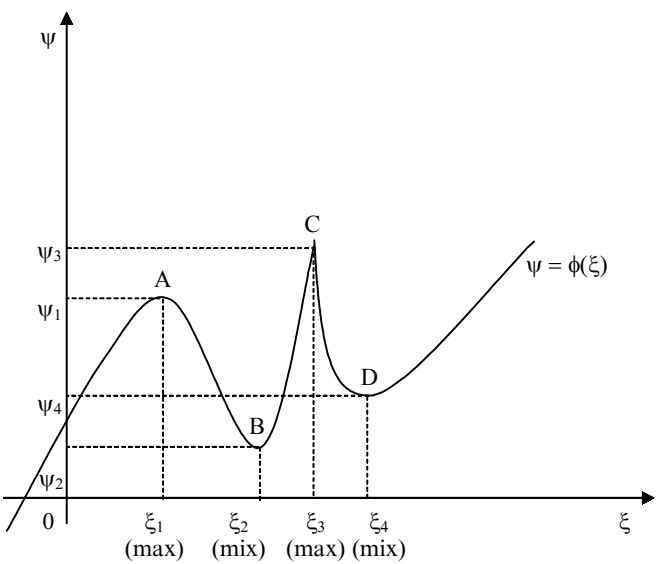


Рис. 4.9

Теперь сформулируем необходимый признак экстремума [2, 16]: *если в точке (A, B, C, D на рис. 4.9) функция* $\psi = \varphi(\xi)$ *достигает экстремума, то ее производная в этой точке либо равна нулю (A, B, D), либо не существует (C).*

Приведенный признак не является достаточным, т. е. из того факта, что производная в данной точке равна нулю или не существует, еще не следует, что эта точка есть экстремум функции.

Недостаточность данного признака проиллюстрируем примером 4.22. Рассмотрим функцию $y = x^3$ и найдем ее экстремум, используя приведенный признак (найдем производную данной функции, приравняем ее к нулю и найдем координаты экстремума, если он существует).

$y' = 3x^2;\ 3x^2 = 0 \Rightarrow \left.\begin{matrix} x = 0 \\ y = 0 \end{matrix}\right\}$ — экстремум данной функции в соответствии с необходимым признаком.

Но из графика функции $y = x^3$ (см. рис. 4.10) следует, что экстремума в точке с координатами $x = 0$, $y = 0$ у данной функции нет.

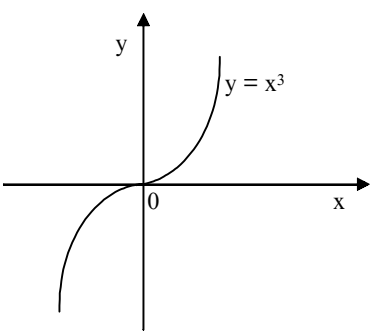


Рис. 4.10

Сформулируем теперь достаточный признак экстремума: точка (A, B, C, D на рис. 4.9) *есть точка экстремума функции* $y = f(x)$, *если производная этой функции* $y' = f'(x)$ *при переходе x через критическую точку (точку, где производная равна нулю или не существует) меняет знак. Если знак меняется с плюса на минус (A, C на рис. 4.9), то имеем локальный максимум, а если знак меняется с минуса на плюс (B, D на рис 4.9), то имеем локальный минимум.*

Заметим, что функция $y = f(x)$ должна быть непрерывна на интервале, содержащем критическую точку. Например, (пример 4.23) производная функции

$y = \frac{2}{x^2}$; $y' = \frac{-4}{x^3}$ меняет знак при переходе через точку x = 0, но экстремума в ней не имеет, так как в этой точке она разрывна [2].

Вернемся к примеру 4.22 и проверим, удовлетворяется ли там достаточный признак экстремума.

Видно (см. рис. 4.11), что производная функции $y = x^3$ не меняет знака при переходе x через точку x = 0, поэтому данная функция не имеет экстремума в точке с координатами x = 0, y = 0.

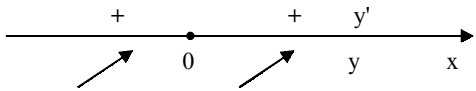


Рис. 4.11

В качестве еще одного примера (пример 4.24) рассмотрим функцию

$y = 3x^3 + 6x^2 - x + 2$.

$y' = 9x^2 + 12x - 1$

$9x^2 + 12x - 1 = 0$

$x_1 \approx 0{,}06$; $x_2 \approx -1{,}4$.

$y' = 9(x - 0{,}06)(x + 1{,}4)$.

Применяя достаточный признак экстремума, находим, что в точке x = –1,4 — максимум, а в точке x = 0, 06 — минимум (см. рис. 4.12).

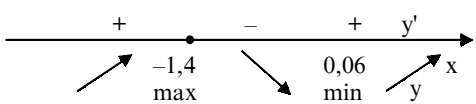


Рис. 4.12

Точки экстремума можно находить и с помощью второй производной. Для этого сформулируем второй достаточный признак экстремума: *некоторая точка с координатами x_0, y_0 будет точкой экстремума функции $y = f(x)$, если $f'(x_0) = 0$, а $f''(x_0) \neq 0$, при этом если $f''(x_0) > 0$, то данная точка будет точкой минимума функции $y = f(x)$, а если $f''(x_0) < 0$ — точкой максимума; в том случае, если $f''(x_0) = 0$, данный признак не применим* [2, 16].

Используем приведенный признак для нахождения экстремумов функции $y = 3x^3 + 6x^2 – x + 2$ на примере 4.24.

$y'' = 18x + 12$.

$y''(x = 0{,}06) = 18 \times 0{,}06 + 12 \approx 13{,}1$.

$y''(x = –1{,}4) = 18(–1{,}4) + 12 \approx –13{,}2$.

Следовательно, в точке $x = 0, 06$ исходная функция будет иметь минимум, а в точке $x = –1, 4$ — максимум.

Теперь покажем, как применять вторую производную для нахождения участков выпуклости и вогнутости функции и ее точек перегиба.

Сначала приведем соответствующие определения.

График дифференцируемой функции $y = f(x)$ называется вогнутым (в положительном направлении оси ординат) на некотором интервале, если на этом интервале он расположен выше касательной, проведенной в любой точке графика в этом интервале (см. рис 4.13).

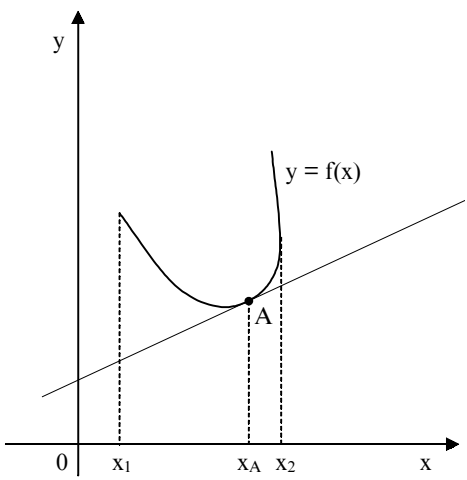


Рис 4.13

График дифференцируемой функции $y = f(x)$ называется выпуклым вверх (в положительном направлении оси ординат) на некотором интервале, если на этом интервале он расположен ниже касательной, проведенной *к любой точке графика в этом интервале* (см. рис. 4.14).

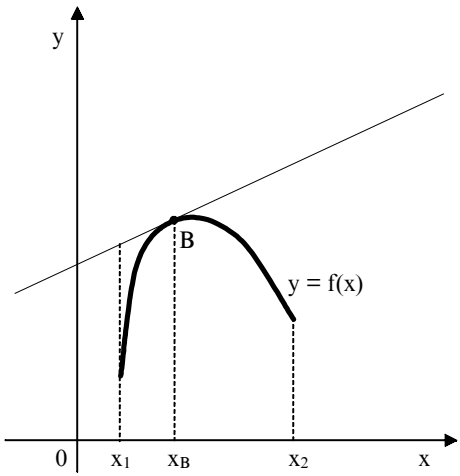


Рис. 4.14

Точки, отделяющие участки выпуклости функции от участков ее вогнутости (и наоборот), называются точками перегиба.

Теперь сформулируем теорему 4.5: *если вторая производная функции $y = f(x)$ всюду на некотором интервале меньше нуля, то функция $y = f(x)$ на этом интервале — выпуклая; если вторая производная функции $y = f(x)$ всюду на некотором интервале больше нуля, то функция $y = f(x)$ на этом интервале — вогнутая* [2, 16, 20].

Приведем также необходимый признак существования точки перегиба: *если точка с координатами x_0, y_0 является точкой перегиба функции $y = f(x)$, то вторая производная данной функции в этой точке либо равна нулю, либо не существует* [2, 16].

Недостаточность данного признака мы проиллюстрируем примером (пример 4.25).

Рассмотрим функцию $y = x^4$. Воспользовавшись приведенным выше признаком, проверим, есть ли у этой функции точки перегиба.

$$y' = 4x^3;\ y'' = 12x^2.$$

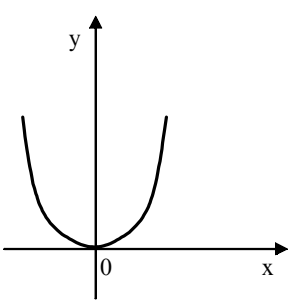


Рис. 4.15

$12x^2 = 0 \Rightarrow \left.\begin{matrix} x = 0 \\ y = 0 \end{matrix}\right\}$ должна быть точкой перегиба по необходимому признаку, но если взглянуть на график этой функции (см. рис. 4.15), видно, что в данной точке перегиба нет.

Поэтому сформулируем достаточный признак существования точки перегиба: *точка с координатами x_0, y_0 является точкой перегиба функции $y = f(x)$, если $f''(x)$,*

меняет знак при переходе x через x_0; если знак меняется с минуса на плюс, то слева от данной точки лежит участок выпуклости, а справа — участок вогнутости, а если знак меняется с плюса на минус, то наоборот [2, 16].

Применим данный признак к функции из примера 4.25.

Видно (см. рис. 4.16), что достаточный признак не выполняется, поэтому в точке с координатами x = 0, y = 0 перегиба нет.

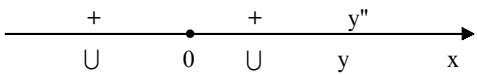


Рис. 4.16

Теперь применим достаточный признак существования точки перегиба к функции из примера 4.24.

$y'' = 18x + 12;\ 18x + 12 = 0;\ x = -\frac{12}{18} = -\frac{2}{3}.$

В данном случае достаточный признак свидетельствует о том, что точка с абсциссой $\left(-\frac{2}{3}\right)$ является точкой перегиба функции $y = 3x^3 + 6x^2 - x + 2$ (см. рис. 4.17).

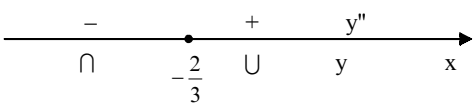


Рис. 4.17

Приведем схему, по которой удобно проводить исследование функций [2]:

1. Нахождение области определения функции, точек ее разрыва, интервалов ее непрерывности и вертикальных асимптот.
2. Проверка функции на четность, нечетность, периодичность.
3. Нахождение точек пересечения графика функции с осями координат (если это не требует больших вычислительных затрат).

4. Нахождение интервалов монотонности и точек экстремума функции.

5. Нахождение участков выпуклости, вогнутости функции и точек ее перегиба.

6. Нахождение наклонных асимптот.

7. Построение графика функции по результатам проведенного исследования.

Пример 4.26

Теперь в соответствии с приведенной схемой исследуем функцию $y = \frac{x^2}{x-2}$.

Данная функция определена на всей оси 0x за исключением точки x = 2, т. е. $x \in (-\infty; 2) \cup (2; +\infty)$. Прямая x = 2 является вертикальной асимптотой данной функции, так как

$$\lim_{x \to 2+} \frac{x^2}{x-2} = +\infty; \quad \lim_{x \to 2-} \frac{x^2}{x-2} = -\infty.$$

Функция не является ни четной, ни нечетной, ни периодической, так как $f(-x) \neq f(x)$, $f(-x) \neq -f(x)$, $f(x + T) \neq f(x)$, где T — период, а график данной функции проходит через начало координат.

Теперь найдем первую производную исходной функции и найдем участки монотонности и экстремумы.

$$y' = \frac{2x(x-2) - x^2}{(x-2)^2} = \frac{x^2 - 4x}{(x-2)^2};$$

$x^2 - 4x = 0$; $x(x - 4) = 0$; $x = 0$; $x = 4$.

Точку x = 2, где не существует первая производная исходной функции, на экстремум можно не проверять, так как в этой точке сама функция имеет бесконечный разрыв.

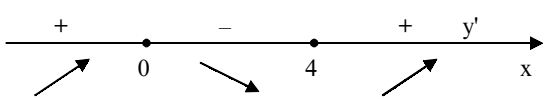


Рис. 4.18

Следовательно (см. рис. 4.18), в соответствии с достаточным признаком экстремума данная функция имеет максимум в точке с координатами x = 0, y = 0 и минимум в точке с координатами x = 4, y = 8.

Теперь найдем вторую производную и определим участки вогнутости, выпуклости и точки перегиба, используя теорему 4.5 и достаточный признак существования точки перегиба.

$$y'' = \frac{(2x-4)(x-2)^2 - 2(x-2)(x^2-4x)}{(x-2)^4} =$$

$$= \frac{(x-2)[(2x-4)(x-2) - 2x^2 + 8x]}{(x-2)^4} = \frac{2x^2 - 4x - 4x + 8 - 2x^2 + 8x}{(x-2)^3} =$$

$$= \frac{8}{(x-2)^3}.$$

Видно, что вторая производная нигде не обращается в ноль, следовательно, данная функция не имеет точек перегиба. Надо только проверить, меняет ли вторая производная исходной функции знак при переходе x через точку бесконечного разрыва x = 2 (вторая производная заданной функции также не существует в точке x = + 2).

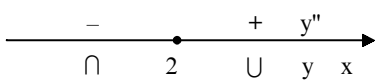


Рис. 4.19

Поэтому (см. рис. 4.19) слева от точки x = 2 исходная функция будет выпуклой , а справа от точки x = 2 — вогнутой.

Теперь проверим, имеет ли исходная функция наклонные асимптоты, для этого воспользуемся формулами (4.12) и (4.13) (так как заданная функция является дробно-рациональной, можно рассматривать произвольное стремление x к бесконечности).

$$a = \lim_{x\to\infty} \frac{x^2}{(x-2)x} = \lim_{x\to\infty} \frac{x^2}{x^2-2x} = \lim_{x\to\infty} \frac{1}{1-\frac{2}{x}} = \frac{1}{1-0} = 1.$$

$$b = \lim_{x\to\infty}\left(\frac{x^2}{x-2} - x\right) = \lim_{x\to\infty} \frac{x^2 - x^2 + 2x}{x-2} = \lim_{x\to\infty} \frac{2x}{x-2} =$$

$$= \lim_{x\to\infty} \frac{2}{1-\frac{2}{x}} = \frac{2}{1-0} = 2.$$

Поэтому прямая $y = x + 2$ является наклонной асимптотой исходной функции.

Теперь по результатам проведенного исследования построим график заданной функции (см. рис. 4.20).

Задачи для самостоятельного решения

1. Найти производные следующих функций:

1.1. $y = \log_{\cos 2x}(5\sin^2 8x - e^{4x^3})$;

1.2. $y = (\operatorname{tg} 5x)^{\cos 3x}$;

1.3. $y = e^{\operatorname{tg} 3x} \times \ln 6(2x^4 + 7x)$;

1.4. $y = \dfrac{\sqrt[5]{\cos^2 5x - x^6}}{16\operatorname{tg}^3 7x}$;

1.5. $y = 6^{\operatorname{ctg} 2x} \times x^8 + \ln^2 4x$;

1.6. $y = e^{\operatorname{tg} 3x} \times 7^{\cos^2 2x}$.

2. Найти вторые производные следующих функций:

2.1. $y = 6x^4 + 4\sin 3x$;

2.2. $y = e^{7x} \times \operatorname{tg} 5x$;

2.3. $y = \dfrac{\ln 6x}{4x^2}$;

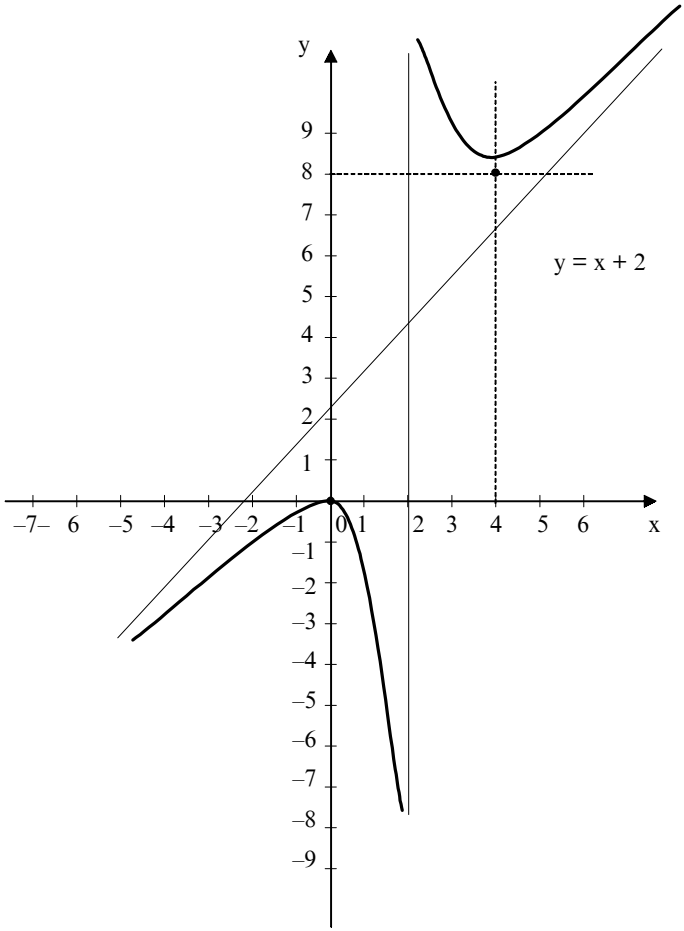


Рис. 4.20

2.4. $y = 5x^8 - 16x^5 + \sin 2x$.

3. Исследовать функции и построить их графики:

3.1. $y = 2x^4 - 8x^2 + 3$;

3.2. $y = \frac{1}{3}x^3 - 2x^2 + 3$;

3.3. $y = 2x^2 - 10$;

3.4. $y = 2x^3 - 9x^2 + 15x - 6$;

3.5. $y = 3x - x^3$;

3.6. $y = \frac{x}{x^2 + 16}$.

4. Найти $\frac{\partial z}{\partial x}$ и $\frac{\partial z}{\partial y}$, если функция Z имеет вид:

4.1. $Z = x^3 \times \sin^2 y$;

4.2. $Z = x^y$;

4.3. $Z = \sqrt{2x^2 + y^2}$;

4.4. $Z = tg^2(x^3 y) \cdot 2^{2x+3}$;

4.5. $Z = \operatorname{arctg}(x^3 + 5y^6)$;

4.7. $Z = \sqrt{tg^5 x^3 - \ln^2 y^6}$.

5. Используя правило Лопиталя, найти пределы функций:

5.1. $\lim\limits_{x \to 1} \frac{\sqrt[3]{1-2x} + 1}{\sqrt{2-x} - 1}$;

5.2. $\lim\limits_{x \to \infty} \frac{4x^3 + 7x^2 - 6x}{8x^4 - 15x^2 - 10}$;

5.3. $\lim\limits_{x \to 0} (\cos 3x)^{\frac{1}{x}}$;

5.4. $\lim\limits_{x \to 0} \frac{tg8x}{tg12x}$;

5.5. $\lim\limits_{x\to 1}\dfrac{2\ln x}{1-x^3}$;

5.6. $\lim\limits_{x\to 0}\left(\dfrac{1}{x}-\dfrac{1}{e^x-1}\right)$.

Вопросы для самопроверки

1. Дать определение производной функции $y = f(x)$.
2. Каковы геометрический и механический смыслы производной?
3. Как найти производную сложной функции?
4. Дать определение дифференциала функции $y = f(x)$.
5. Какой геометрический смысл имеет дифференциал?
6. Что называется производной второго порядка от функции $y = f(x)$?
7. В чем состоит достаточный признак экстремума?
8. Какие точки называются точками перегиба функции $y = f(x)$?
9. Что называется асимптотой функции $y = f(x)$?
10. Сформулировать правило Лопиталя и привести примеры его применения.
11. Что называется функцией двух независимых переменных?
12. Что называется графиком функции двух независимых переменных?
13. Что называется пределом функции $Z = f(x, y)$ при $x \to x_0$ и $y \to y_0$?
14. Дать определение частных производных функции двух независимых аргументов.

5. Основы интегрального исчисления

Интегральное исчисление — это раздел математического анализа, в котором изучаются свойства и способы вычисления интегралов и их применение.

Интегрирование — это действие, обратное дифференцированию. Например, с его помощью находится скорость тела по заданному ускорению.

5.1. Первообразная и неопределенный интеграл

Первообразной для функции $y = f(x)$ на некотором промежутке называется функция $F(x)$, производная которой равна исходной функции, т. е. $F'(x) = f(x)$.

Из этого определения следует, что любая функция по отношению к своей производной является первообразной [2, 16].

Рассмотрим пример $y = x^5$. Данная функция служит производной для функции $y = \frac{x^6}{6}$, так как $\left(\frac{x^6}{6}\right)' = x^5$ и $\left(\frac{x^6}{6} - 1000\right)' = x^5$, или в общем виде $\left(\frac{x^6}{6} + C\right)' = x^5$, где $C = const$.

Из данного примера видно, что любая функция $\left(\frac{x^6}{6} + C\right)$ будет первообразной для функций $y = x^5$.

Теперь приведем формулировку основной теоремы о первообразных.

ТЕОРЕМА 5.1.1. *Любая непрерывная функция имеет бесконечное множество первообразных, причем любые две из них друг от друга отличаются постоянным слагаемым* [2, 22].

Формула F (x) + C исчерпывает множество всех первообразных исходной функции. Геометрически выражение F (x) + C есть семейство кривых (см. рис. 5.1.1), каждая из которых получается путем сдвига одной из кривых вдоль оси 0y.

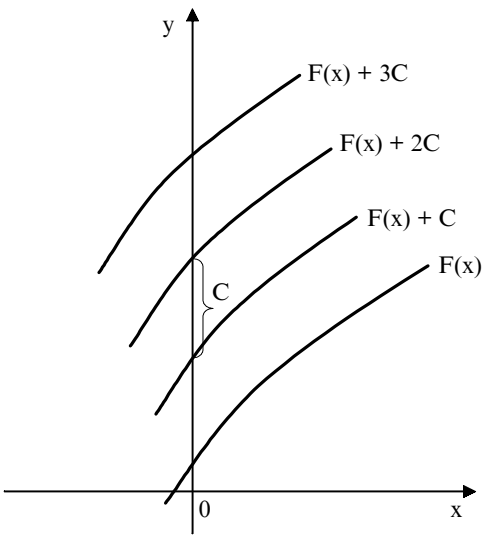


Рис. 5.1.1

Заметим, что первообразную можно находить не только по производной, но и по дифференциалу.

Теперь дадим определение неопределенного интеграла.

Отыскание первообразных называется неопределенным интегрированием, а выражение, охватывающее совокупность всех первообразных для данной функции f(x), называется неопределенным интегралом и обозначается так:

$\int f(x)dx$,

где $f(x)$ — подынтегральная функция;

$f(x)dx$ — подынтегральное выражение;

x — переменная интегрирования.

Заметим, что $f(x)$ на участке интегрирования должна быть непрерывна.

$\int$ — знак интеграла.

Таким образом, неопределенный интеграл есть семейство функций $F(x) + C$, т. е.

$\int f(x)dx = F(x) + C$ [2, 6].

Нахождение всех первообразных для данной функции $f(x)$ и называется неопределенным интегрированием. Термин неопределенное интегрирование появился, потому что не указывается, какая первообразная имеется в виду.

Сразу скажем, что интегрирование значительно сложнее дифференцирования. Дифференцирование любых элементарных функций производится по определенным правилам, а интегрирование требует в каждом конкретном случае индивидуального подхода. Разумеется, есть общие методы интегрирования, некоторые из которых будут рассмотрены далее. Заметим, что производная от любой элементарной функции есть функция элементарная, а про неопределенный интеграл от элементарной функции этого сказать нельзя, т. е. первообразная от элементарной функции может оказаться и не представимой с помощью конечного числа элементарных функций. Про такие функции говорят, что они не интегрируемы в элементарных функциях. Примерами так называемых неберущихся интегралов являются:

$$\int \frac{dx}{\sqrt{1+x^3}};\ \int \frac{dx}{\ln x};\ \int \frac{\sin x dx}{x};\ \int \frac{\cos x dx}{x};\ \int \frac{e^x dx}{x};\ \int e^{-x^2} dx \text{ и др.}$$

Из определения неопределенного интеграла следует, что

$$\left.\begin{aligned} &\left(\int f(x)dx\right)' = f(x) \\ &d\left(\int f(x)dx\right) = f(x)dx \\ &\int f'(x)dx = \int df(x) = f(x) + C \end{aligned}\right\} \qquad (5.1.1)$$

Найдем неопределенные интегралы от основных элементарных функций, использовав для этого таблицу производных от основных элементарных функций (см. главу “Основы дифференциального исчисления”).

Например, $(\sin x)' = \cos x$. Перепишем это равенство в виде

$$\frac{d\sin x}{dx} = \cos x \Rightarrow d\sin x = \cos x dx.$$

Проинтегрируем обе части последнего равенства и с учетом третьей формулы (5.1.1) получим

$$\int d\sin x = \int \cos x dx \Rightarrow \sin x + C = \int \cos x dx.$$

Это и есть табличный интеграл.

Точно так же получают и другие табличные интегралы от основных элементарных функций.

Приведем таблицу 5.1.1 интегралов от основных элементарных функций. Справедливость приведенных формул легко проверить дифференцированием.

Таблица 5.1.1

1) $\int x^n dx = \frac{x^{n+1}}{n+1} + C,\ n \neq -1;$

2) $\int \frac{dx}{x} = \ln|x| + C;$

3) $\int a^x dx = \frac{a^x}{\ln a} + C;$

4) $\int e^x dx = e^x + C;$

5) $\int \cos x dx = \sin x + C;$

6) $\int \sin x dx = -\cos x + C;$

7) $\int \frac{dx}{\cos^2 x} = \operatorname{tg} x + C;$

8) $\int \frac{dx}{\sin^2 x} = -\operatorname{ctg} x + C;$

9) $\int \frac{dx}{x^2 + 1} = \operatorname{arctg} x + C;$

10) $\int \frac{dx}{\sqrt{1 - x^2}} = \arcsin x + C;$

11) $\int \frac{dx}{\sqrt{x^2 \pm 1}} = \ln\left|x + \sqrt{x^2 \pm 1}\right| + C.$

К таблице 5.1.1 добавим формулы интегрирования гиперболических и обратных гиперболических функций:

12) $\int \operatorname{ch} x dx = \operatorname{sh} x + C;$

13) $\int \operatorname{sh} x dx = \operatorname{ch} x + C;$

14) $\int \frac{dx}{\operatorname{ch}^2 x} = \operatorname{th} x + C;$

15) $\int \frac{dx}{\operatorname{sh}^2 x} = \operatorname{cth} x + C;$

16) $\int \frac{dx}{1 - x^2} = \operatorname{Arth} x + C = \frac{1}{2} \ln\left|\frac{1 + x}{1 - x}\right| + C;$

17) $\int \frac{dx}{\sqrt{x^2 + 1}} = \operatorname{Arsh} x + C = \ln\left|x + \sqrt{x^2 + 1}\right| + C;$

18) $\int \frac{dx}{\sqrt{x^2 - 1}} = \operatorname{Arch} x + C = \ln\left|x + \sqrt{x^2 - 1}\right| + C.$

Объединение формул 17 и 18 и дает формулу 11 основной таблицы 5.1.1. Заметим, что кроме таблицы 5.1.1 существуют таблицы интегралов от элементарных и специальных функций (*Брычков Ю. А., Маричев О. И., Прудников А. П.* Таблицы неопределен-

ных интегралов. — М.: Наука, 1986; Интегралы и ряды. В 3-х томах. — М.: Наука, 1986.)

Задача "взятия" неопределенного интеграла и состоит в том, чтобы преобразовать его к табличному.

Свойства неопределенного интеграла

1. Неопределенный интеграл от алгебраической суммы конечного числа непрерывных функций равен алгебраической сумме неопределенных интегралов от этих функций, т. е.

$$\int(f_1(x) \pm f_2(x) \pm ... \pm f_n(x))dx = \int f_1(x)dx \pm \int f_2(x)dx \pm ... \pm \int f_n(x)dx.$$

2. Постоянный множитель подынтегральной функции можно выносить за знак неопределенного интеграла, т. е.

$\int kf(x)dx = k\int f(x)dx$, где $k \in R$.

3. Любая формула интегрирования сохраняет свой вид при подстановке вместо независимой переменной любой дифференцируемой функции от нее, т. е. если $\int f(x)dx = F(x) + C$, то и $\int f(u)du = F(u) + C$, где $u = u(x)$ — любая дифференцируемая функция от x.

В силу правила (3) таблица 5.1.1 будет справедливой независимо от того, является ли переменная интегрирования независимой переменной или любой дифференцируемой функцией от нее, т. е. таблица 5.1.1 сразу значительно расширяется [2, 22].

Методы интегрирования

1. Непосредственное интегрирование. Используется таблица 5.1.1, свойства неопределенных интегралов и различные преобразования подынтегрального выражения.

Пример 5.1.1

$$\int\left(2\sin x + 3x^3 - \frac{5}{\cos^2 x}\right)dx = 2\int \sin x dx + 3\int x^3 dx - 5\int \frac{dx}{\cos^2 x} =$$

$$= -2\cos x + \frac{3}{4}x^4 - 5tgx + C.$$

Пример 5.1.2

$$\int \frac{3x^3 + 4x^2 + 7x}{2x}dx =$$

$$= \frac{3}{2}\int x^2 dx + 2\int x dx + \frac{7}{2}\int dx = \frac{1}{2}x^3 + x^2 + \frac{7}{2}x + C.$$

Пример 5.1.3

$$\int \frac{\sin 2x}{\cos x} dx = \int \frac{2\sin x \cos x}{\cos x} dx = 2\int \sin x dx = -2\cos x + C.$$

Пример 5.1.4

$\int \sin^2 x dx =$ [Воспользуемся формулой $\cos 2x = \cos^2 x - \sin^2 x$, $\cos 2x =$

$= 1 - 2\sin^2 x$, $\sin^2 x = \frac{1-\cos 2x}{2}$. Последнее выражение подставляем вместо подынтегральной функции] $= \int \frac{1-\cos 2x}{2} dx =$

$= \frac{1}{2}(\int dx - \int \cos 2x dx) = \frac{1}{2}x - \frac{1}{2} \times \frac{1}{2}\int \cos 2x d2x.$ [Заметим, что $d2x = 2dx$,

заменяем $2x$ на y, т. е. $2x = y$, тогда получим] $= \frac{1}{2}x - \frac{1}{4}\int \cos y dy =$

$= \frac{1}{2}x - \frac{1}{4}\sin y + C$ = [Возвращаемся к прежнему аргументу] =

$= \frac{1}{2}x - \frac{1}{4}\sin 2x + C.$

Таким образом, в примере 5.1.4 мы использовали еще один метод интегрирования (замена переменной), который более подробно рассмотрим ниже.

2. Интегрирование по частям. Этот метод следует из формулы дифференцирования произведения двух функций.

Пусть $u(x)$ и $v(x)$ — дифференцируемые функции аргумента x, тогда имеем:

$(uv)' = u'v + v'u$ или

$d(uv) = vdu + udv$

$udv = d(uv) - vdu.$

Интегрируем обе части последнего равенства и получаем:

$$\int udv = uv - \int vdu. \qquad (5.1.2)$$

Это и есть формула интегрирования по частям.

Данный способ состоит в том, что подынтегральное выражение представляется в виде произведения двух множителей u и dv и заменяется двумя интегрированиями: 1) отыскание v из выражения для dv; 2) отыскание интеграла от vdu.

Смысл способа состоит в том, что эти два интегрирования выполнить легче, чем "взять" исходный интеграл [2, 6, 22].

Рассмотрим конкретные примеры и применения данного метода.

Пример 5.1.5

$$\int \ln x dx = \left[u = \ln x, dv = dx \Rightarrow v = x, du = \frac{dx}{x} \right] = x \ln x - \int x \frac{dx}{x} =$$

$$= x \ln x - x + C.$$

В данном примере выбор u и dv производится однозначно, но так бывает не всегда.

Пример 5.1.6

$$\int xe^x dx = [u = x; du = dx; dv = e^x dx; v = \int e^x dx = e^x] =$$

$$= xe^x - \int e^x dx = xe^x - e^x + C,$$

но если принять [$u = e^x$; $du = e^x dx$; $dv = xdx$; $v = \frac{x^2}{2}$], то $\int xe^x dx = \frac{x^2}{2} e^x - \frac{1}{2} \int x^2 e^x dx$, т. е. мы получим более сложный интеграл, чем исходный.

Бывают случаи, когда формулу (5.1.2) надо применять несколько раз.

Пример 5.1.7

$\int e^x \sin x dx = [u = \sin x; du = \cos x dx; dv = e^x dx; v = \int e^x dx = e^x] =$
$= e^x \sin x - \int e^x \cos x dx =$ [К интегралу $\int e^x \cos x dx$ опять применим формулу

(5.1.2) и получим: u = cosx; du = –sinxdx; dv = $e^x dx$; v = e^x] = e^xsinx – – (e^xcosx + ∫e^xsinxdx) = e^xsinx – e^xcosx – ∫e^xsinxdx.

Перенесем ∫e^xsinxdx в левую часть равенства и получим:

2∫e^xsinxdx = e^x(sinx – cosx) + 2C (постоянная может быть любой, возьмем ее равной 2C);

$$\int e^x \sin x dx = \frac{e^x}{2}(\sin x - \cos x) + C.$$

Пример 5.1.8

$$\int \sqrt{1-x^2}\,dx =$$

$$= \left[u = \sqrt{1-x^2}; dv = dx \Rightarrow v = x; du = \frac{1}{2} \times \frac{(-2x)dx}{\sqrt{1-x^2}} = \frac{(-x)dx}{\sqrt{1-x^2}} \right] =$$

$$= x\sqrt{1-x^2} - \int \frac{(-x^2)dx}{\sqrt{1-x^2}} = x\sqrt{1-x^2} - \int \frac{1-x^2-1}{\sqrt{1-x^2}} dx =$$

$$= x\sqrt{1-x^2} - \int \frac{1-x^2}{\sqrt{1-x^2}} dx + \int \frac{dx}{\sqrt{1-x^2}} =$$

$$= x\sqrt{1-x^2} - \int \sqrt{1-x^2}\,dx + \arcsin x.$$

Переносим $\int \sqrt{1-x^2}\,dx$ в левую часть и получаем:

$$2\int \sqrt{1-x^2}\,dx = x\sqrt{1-x^2} + \arcsin x + 2C$$

$$\int \sqrt{1-x^2}\,dx = \frac{x}{2}\sqrt{1-x^2} + \frac{1}{2}\arcsin x + C.$$

3. Метод замены переменной. Его применяют в том случае, если исходный интеграл сложно или невозможно с помощью алгебраических и иных преобразований свести к одному или нескольким табличным интегралам [2, 16].

Способ заключается в следующем: заменяется новой переменной такая часть подынтегральной функции, при дифференцировании которой получается оставшаяся часть подынтегрального выражения (не считая постоянного множителя, на который всегда можно умножить или разделить подынтегральное выражение).

Метод замены переменной основан на следующей Теореме 5.1.2: *пусть некоторая функция* $\varphi(t) = x$ *определена и дифференцируема на некотором промежутке* $[a, b]$, *пусть* X — *множество значений этой функции, на котором определена функция* $f(x)$. *Тогда, если на множестве* X *функция* $f(x)$ *имеет первообразную, то на отрезке* $[a, b]$ *справедлива формула*

$$\int f(x)dx \,|_{x=\varphi(t)} = \int f(\varphi(t))\varphi'(t)dt. \qquad (5.1.3)$$

В некоторых случаях лучше использовать замену переменной не в виде $x = \varphi(t)$, а $t = \psi(x)$[2, 16, 22].

Приведем конкретные примеры.

Пример 5.1.9

Найти $\int(5x - 6)^{90}dx$.

В этом интеграле можно разложить подынтегральную функцию, используя бином Ньютона, но это будет слишком длинно, поэтому делаем замену переменной: $t = 5x - 6 \Rightarrow x = \frac{t+6}{5}$; $dx = \frac{dt}{5}$, получаем

$$\int (5x-6)^{90}\,dx = \frac{1}{5}\int t^{90}dt = \frac{1}{5}\cdot\frac{t^{91}}{91} + C,$$

или, возвращаясь к первоначальной переменной x, имеем:

$$\int (5x-6)^{90}\,dx = \frac{(5x-6)^{91}}{455} + C.$$

Пример 5.1.10

$$\int \frac{dx}{x^2+16} = \int \frac{dx}{\frac{16(x^2+16)}{16}} = \frac{1}{16}\int \frac{dx}{(\frac{x}{4})^2+1} =$$ [Делаем замену переменной

$$y=\frac{x}{4}\Rightarrow x=4y\Rightarrow dx=4dy\ ]=\frac{1}{16}4\int\frac{dy}{y^2+1}=\frac{1}{4}arctgy+C=$$

[Возвращаем переменную x и получаем] $=\frac{1}{4}arctg\left(\frac{x}{4}\right)+C.$

Пример 5.1.11

$$\int\frac{dx}{x^2+2x+5}=\int\frac{dx}{(x+1)^2+4}=\frac{1}{4}\int\frac{dx}{\frac{(x+1)^2+4}{4}}=\frac{1}{4}\int\frac{dx}{\left(\frac{x+1}{2}\right)^2+1}=$$

$=$ [Теперь делаем замену переменной $t=\frac{x+1}{2}\Rightarrow 2t-1=x\Rightarrow dx=$

$= 2dt] = \frac{1}{4}\int\frac{2dt}{t^2+1}=\frac{1}{2}arctgt+C=$ [Возвращаем переменную x и получаем] $=$

$$=\frac{1}{2}arctg\left(\frac{x+1}{2}\right)+C.$$

Пример 5.1.12

$$\int\frac{2x^2dx}{x^3+7}=\frac{2}{3}\int\frac{3x^2dx}{x^3+7}=$$ [Заметим, что $d(x^3 + 7) = 3x^2dx$] $=$

$$=\frac{2}{3}\int\frac{d(x^3+7)}{x^3+7}=$$ [Делаем замену переменной $y = x^3 + 7$] $=$

$$=\frac{2}{3}\int\frac{dy}{y}=\frac{2}{3}\ln|y|+C=$$ [Возвращаем переменную x и получаем] $=$

$$=\frac{2}{3}\ln\left|x^3+7\right|+C.$$

Пример 5.1.13

$$\int \frac{2^{\sqrt{x}}\,dx}{\sqrt{x}} = \text{[Заметим, что } d\left(\sqrt{x}\right) = \frac{1}{2\sqrt{x}}dx\text{]} = 2\int 2^{\sqrt{x}}\,d\sqrt{x} = \text{[Теперь}$$

делаем замену переменной $\sqrt{x} = y$] $= 2\int 2^{y}\,dy = 2\cdot\frac{2^{y}}{\ln 2} + C =$ [Возвращаем

переменную x] $= \frac{2^{\sqrt{x}+1}}{\ln 2} + C.$

Пример 5.1.14

$\int e^{\sin 2x}\sin 2x\,dx = 2\int e^{\sin 2x}\sin x\cos x\,dx =$ [Заметим, что $d\sin^2 x = 2\sin x\cos x\,dx$] $= \int e^{\sin^2 x}\,d\sin^2 x =$ [Теперь делаем замену переменной $t = \sin^2 x$] $= \int e^{t}dt = e^{t} + C =$ [Возвращаем переменную x] $= e^{\sin 2x} + C$.

Интегрирование рациональных дробей

Любая рациональная функция R(x) может быть представлена в виде дроби, т. е.

$$R(x) = \frac{P(x)}{Q(x)},$$

где P(x) и Q(x) — многочлены.

Если степень числителя (m) больше или равна степени знаменателя (n), то разделив P(x) на Q(x), мы получим многочлен $P_1(x)$ и в остатке многочлен $P_2(x)$ не выше (n — 1) степени, т. е.

$$\frac{P(x)}{Q(x)} = P_1(x) + \frac{P_2(x)}{Q(x)}.$$

Интегрирование $P_1(x)$ проходит без проблем.

Надо проинтегрировать правильную рациональную дробь, степень числителя которой меньше степени знаменателя $\left(\frac{P_2(x)}{Q(x)}\right)$.

$\frac{P_2(x)}{Q(x)}$ можно представить в виде суммы простейших дробей двух видов:

$$\frac{A_i}{(x-a)^i}; \frac{B_i x + C_i}{(x^2 + px + q)^i},$$

где A_i, B_i, C_i — постоянные[2, 20, 22].

Каждому множителю $(x - a)^k$ в представлении знаменателя $Q(x)$ соответствует в разложении дроби $\frac{P_2(x)}{Q(x)}$ на слагаемые сумма k простейших дробей вида:

$$\frac{A_k}{(x-a)^k} + \frac{A_{k-1}}{(x-a)^{k-1}} + \ldots + \frac{A_1}{(x-a)}.$$

Каждому множителю $(x^2 + px + q)^t$ соответствует сумма t простейших дробей вида:

$$\frac{B_t x + C_t}{(x^2 + px + q)^t} + \frac{B_{t-1} x + C_{t-1}}{(x^2 + px + q)^{t-1}} + \ldots + \frac{B_1 x + C_1}{(x^2 + px + q)}.$$

Имеет место следующее разложение дроби $\frac{P_2(x)}{Q(x)}$ на слагаемые [2, 22]:

$$\frac{P_2(x)}{Q(x)} = \frac{A_k}{(x-a)^k} + \frac{A_{k-1}}{(x-a)^{k-1}} + \ldots + \frac{A_1}{(x-a)} + \ldots + \frac{B_t x + C_t}{(x^2 + px + q)^t} +$$

$$+ \frac{B_{t-1} x + C_{t-1}}{(x^2 + px + q)^{t-1}} + \ldots + \frac{B_1 x + C_1}{(x^2 + px + q)}. \qquad (5.1.4)$$

Пример 5.1.15

$\int \frac{dx}{\sqrt{e^x+1}}$ = [Делаем замену переменной, обозначим $\sqrt{e^x+1} = y$, тогда получим $e^x + 1 = y^2 \Rightarrow e^x = y^2 - 1 \Rightarrow \ln e^x = \ln(y^2 - 1)$, $x = \ln(y^2 - 1)$, $dx = \frac{2ydy}{y^2-1}$] = $\int \frac{2ydy}{y(y^2-1)} = 2\int \frac{dy}{y^2-1} = 2\int \frac{dy}{(y-1)(y+1)}$.

Дробь $\frac{1}{(y-1)(y+1)}$ — правильная рациональная дробь. Разложим ее на простейшие дроби (см. 5.1.4):

$$\frac{1}{(y-1)(y+1)} = \frac{A}{y-1} + \frac{B}{y+1},$$

где A и B — неизвестные коэффициенты, которые необходимо найти.

Освобождаясь от знаменателя, имеем:

$1 = A(y + 1) + B(y - 1)$;

$1 = Ay + A + By - B$.

Приравнивая коэффициенты при y и y^0, получим систему уравнений для определения A и B.

$$\left.\begin{array}{l} 0 = A + B \\ 1 = A - B \end{array}\right\} \Rightarrow \begin{array}{l} A = -B \\ 1 = -2B \Rightarrow B = -\frac{1}{2} \text{ и } A = \frac{1}{2}. \end{array}$$

Тогда получим:

$\frac{1}{(y-1)(y+1)} = \frac{0{,}5}{y-1} - \frac{0{,}5}{y+1}$ и искомый интеграл примет вид:

$$2\int \frac{dy}{(y-1)(y+1)} = 2\left(\frac{1}{2}\int \frac{dy}{y-1} - \frac{1}{2}\int \frac{dy}{y+1}\right) = \int \frac{d(y-1)}{y-1} - \int \frac{d(y+1)}{y+1} =$$

[Заметим, что $d(y + 1) = dy$ и $d(y - 1) = dy$] = $\ln|y - 1| - \ln|y + 1| + C$ =

$$\ln\left|\frac{y-1}{y+1}\right| + C = [\text{Вернемся к исходной переменной } e^x] = \ln\left|\frac{\sqrt{e^x+1}-1}{\sqrt{e^x+1}+1}\right| + C.$$

Интегрирование тригонометрических функций

Интеграл вида $\int f(\sin x, \cos x)dx$ с помощью подстановки $u = tg\frac{x}{2}$ можно преобразовать в интеграл от рациональной функции[2, 22].

Используются следующие тригонометрические формулы:

$$\sin x = \frac{2tg\frac{x}{2}}{1+tg^2\frac{x}{2}} = \frac{2u}{1+u^2};$$

$$\cos x = \frac{1-tg^2\frac{x}{2}}{1+tg^2\frac{x}{2}} = \frac{1-u^2}{1+u^2}.$$

Из равенства $x = 2arctgu$ имеем $dx = \frac{2du}{1+u^2}$.

В результате указанной подстановки исходный интеграл преобразуется к виду:

$$\int f(\sin x, \cos x)dx = \int f\left(\frac{2u}{1+u^2}, \frac{1-u^2}{1+u^2}\right)\frac{2du}{1+u^2},$$

т. е. подынтегральная функция рациональна относительно u.

Пример 5.1.16

$\int \frac{dx}{\cos x}$. Применим подстановку $u = tg\frac{x}{2}$ и получаем:

$$\int \frac{dx}{\cos x} = \int \frac{2du}{(1+u^2)\frac{(1-u^2)}{(1+u^2)}} = 2\int \frac{du}{1-u^2} = [\text{Воспользуемся формулой (16) из таблицы 5.1.1}] = \ln\left|\frac{1+u}{1-u}\right| + C = \ln\left|\frac{1+tg\frac{x}{2}}{1-tg\frac{x}{2}}\right| + C.$$

С помощью указанной подстановки хорошо "берутся" интегралы вида $\int \frac{dx}{a\cos x + b\sin x + C}$.

Интегрирование функций ∫f(sinx, cosx)dx с помощью подстановки $u = tg\frac{x}{2}$ всегда приводит к успеху, но в силу своей общности она не всегда является оптимальной.

5.2. Определенный интеграл

К понятию определенного интеграла можно прийти, рассматривая различные задачи, например, нахождение площади плоской фигуры, вычисление работы переменной силы, определение пути по заданной переменной скорости.

Найдем площадь криволинейной трапеции, т. е. фигуры, которая ограничена осью Ox, графиком непрерывной функции y = f(x) и двумя прямыми x = a и x = b (см. рис. 5.2.1). Пока будем считать, что криволинейная трапеция расположена над осью Ox, т. е. f(x) > 0

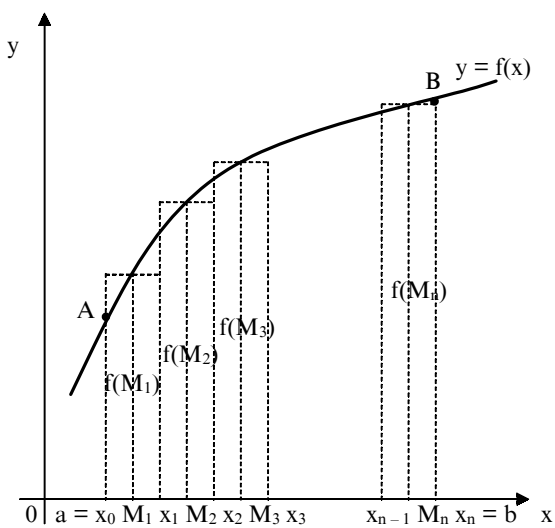


Рис. 5.2.1

Разделим отрезок [a, b] на n частичных отрезков: $[x_0, x_1]$, $[x_1, x_2]$, ..., $[x_{n-1}, x_n]$.

В точках деления отрезка [a, b] проведем прямые, параллельные оси Oy, и разобьем криволинейную трапецию aABb на n частичных трапеций. В каждом из частичных отрезков возьмем по произвольной точке M_1, M_2, ..., M_n (некоторые из этих точек могут совпадать с точками деления отрезка [a, b]).

Через точки M_1, M_2, ..., M_n проведем прямые, параллельные оси Oy до пересечения с функцией y = f(x). Отрезки этих прямых $f(M_1)$, $f(M_2)$, ..., $f(M_n)$ есть ординаты графика функции y = f(x). Взяв частичные отрезки за основания, построим на них n прямоугольников с высотами, равными $f(M_1)$, $f(M_2)$, ..., $f(M_n)$. В результате мы получим ступенчатую фигуру, состоящую их n прямоугольников. Так как площадь любого из прямоугольников будет равна $f(M_i)(x_i - x_{i-1})$, $i = \overline{1,n}$, то площадь ступенчатой фигуры можно найти по формуле

$$S_{\text{ступ}} = f(M_1)(x_1 - x_0) + f(M_2)(x_2 - x_1) + \ldots + f(M_n)(x_n - x_{n-1}) =$$

$$= \sum_{i=1}^{n} f(M_i)(x_i - x_{i-1}). \qquad (5.2.1)$$

При неограниченном увеличении количества частичных отрезков ($n \to \infty$) и при стремлении длины наибольшего из них к нулю ступенчатая фигура будет неограниченно приближаться к криволинейной трапеции aABb, т. е. получим:

$$S_{\text{кр. трап.}} = \lim_{\max(x_i - x_{i-1}) \to 0} \sum_{i=1}^{n} f(M_i)(x_i - x_{i-1}). \qquad (5.2.2)$$

Зная площадь криволинейной трапеции, мы можем находить площади любых плоских фигур (этот вопрос мы подробнее рассмотрим ниже). К выражению вида (5.2.2) приводят и другие задачи (нахождение работы переменной силы, вычисление пути по заданной переменной скорости).

Теперь приведем строгое определение определенного интеграла.

Впервые для непрерывной функции оно было дано в 1823 г. французским математиком Коши, а позднее немецкий математик Риман показал, что определение Коши применимо к более широкому классу функций [2, 22]. Это позволило ему впервые дать в общей форме определение интеграла и установить условие его существования.

Рассмотрим непрерывную на отрезке [a, b] функцию $y = f(x)$ ($f(x)$ не обязательно положительна на [a, b]). Отрезок [a, b] разбивается на n частичных отрезков точками $a = x_0, x_1, x_2, ..., x_n = b$, причем $x_0 < x_1 < x_2 < ... < x_n$.

Во всех частичных отрезках $[x_0, x_1], [x_1, x_2], ..., [x_{n-1}, x_n]$ берутся произвольно точки $M_1, M_2, ..., M_n$, находятся значения функции $y = f(x)$ в этих точках $f(M_1), f(M_2), ..., f(M_n)$.

Составляем сумму вида

$$I_n = \sum_{i=1}^{n} f(M_i)(x_i - x_{i-1}) = \sum_{i=1}^{n} f(M_i)\Delta x_i, \tag{5.2.3}$$

где $\Delta x_i = x_i - x_{i-1}$.

Затем находим предел интегральной суммы (5.2.3) при стремлении к нулю длины наибольшего частичного отрезка, т. е. при $\max \Delta x_i \to 0$.

$$I = \lim_{\max \Delta x_i \to 0} I_n = \lim_{\max \Delta x_i \to 0} \sum_{i=1}^{n} f(M_i)\Delta x_i. \tag{5.2.4}$$

В рассмотренной нами задаче о криволинейной трапеции предел (5.2.4) определяет ее площадь. В общем случае он называется определенным интегралом от функции $f(x)$ в пределах от a до b и читается: интеграл от a до b $f(x)$ по dx, т. е. согласно определению мы получаем:

$$\lim_{\max \Delta x_i \to 0} \sum_{i=1}^{n} f(M_i)\Delta x_i = \int_a^b f(x)dx. \tag{5.2.5}$$

Сумма в выражении (5.2.3) называется n-й интегральной суммой [2, 20, 22].

Как и в неопределенном интеграле $f(x)$ — есть подынтегральная функция, $f(x)dx$ — подынтегральное выражение, переменная x —

переменная интегрирования, отрезок [a, b] называется интервалом интегрирования, а числа a и b нижним и верхним пределами соответственно.

Определенный интеграл есть некоторое число, а величина его зависит только от вида функции f(x) и от чисел a и b. Заметим, что площадь криволинейной трапеции — это геометрический смысл определенного интеграла. Вычисление определенного интеграла с помощью составления интегральных сумм вида (5.2.3) вызывает серьезные проблемы даже в самых простых случаях, поэтому для их нахождения используют другой способ, который мы рассмотрим ниже.

Теперь приведем без доказательства теорему существования определенного интеграла.

ТЕОРЕМА 5.2.1. *Если функция $f(x)$ непрерывна в отрезке $[a, b]$, то ее n-я интегральная сумма стремится к пределу при стремлении к нулю наибольшего частичного интервала*. Этот предел, т. е. определенный интеграл $\int_a^b f(x)dx$, не зависит ни от способа разбиения [a, b] на частичные интервалы, ни от выбора в этих интервалах промежуточных точек [2].

Свойства определенного интеграла

1. Интеграл от алгебраической суммы конечного числа функций равен алгебраической сумме интегралов от этих функций, т. е.

$$\int_a^b (f_1(x) \pm f_2(x) \pm \ldots \pm f_n(x))dx = \int_a^b f_1(x)dx \pm \int_a^b f_2(x)dx \pm \ldots \pm \int_a^b f_n(x)dx.$$

2. Постоянный множитель подынтегральной функции можно выносить за знак интеграла, т. е.

$$\int_a^b kf(x)dx = k\int_a^b f(x)dx, k \in R.$$

3. Если переставить местами пределы интегрирования, то интеграл изменит только знак, т. е.

$$\int_a^b f(x)dx = -\int_b^a f(x)dx.$$

4. Если интервал интегрирования [a, b] разбит на две части [a, c] и [c, b], то

$$\int_a^b f(x)dx = \int_a^c f(x)dx + \int_c^b f(x)dx.$$

5. Если функция f(x) в интервале интегрирования не меняет знака, то интеграл представляет собой число того же знака, что и функция.

6. Значение определенного интеграла заключено между произведениями наименьшего и наибольшего значений функции f(x) на длину интервала интегрирования, т. е.

$$m(b - a) \le \int_a^b f(x)dx \le M(b - a),\ a < b,$$

где M и m — наибольшее и наименьшее соответственно значения функции f(x) на отрезке [a, b] (см. рис. 5.2.2) [22].

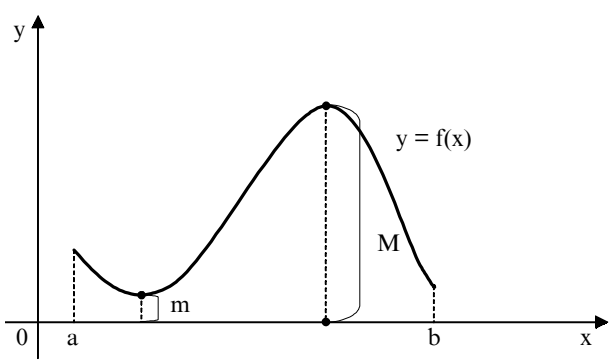


Рис. 5.2.2

7. Если в каждой точке x отрезка [a, b] $\psi(x) \le f(x) \le \varphi(x)$, то

$$\int_a^b \psi(x)dx \le \int_a^b f(x)dx \le \int_a^b \varphi(x)dx, \ a < b.$$

8. Внутри интервала интегрирования [a, b] есть хотя бы одно значение x = A, для которого выполняется следующее равенство:

$$\frac{\int_a^b f(x)dx}{b-a} = f(A).$$

Формула Ньютона-Лейбница

Приведем без доказательства формулировку теоремы.

ТЕОРЕМА 5.2.2. *Значение определенного интеграла равно разности значений любой первообразной от подынтегральной функции, взятых при верхнем и нижнем пределах интегрирования, т. е.* [2, 22]

$$\int_a^b f(x)dx = F(x)\Big|_a^b = F(b) - F(a). \qquad (5.2.6)$$

Или иначе: значение определенного интеграла равно приращению любой первообразной от подынтегральной функции в интервале интегрирования.

Формула (5.2.6) дает удобный способ вычисления определенных интервалов, если известна первообразная подынтегральной функции, т. е. необходимо найти любую первообразную подынтегральной функции и подставить в нее пределы интегрирования.

Приведем конкретные примеры.

Пример 5.2.1

$$\int_0^3 5e^x dx = 5\int_0^3 e^x dx = 5e^x\Big|_0^3 = 5(e^3 - e^0) = 5(e^3 - 1).$$

Пример 5.2.2

$$\int_{\frac{\pi}{6}}^{\frac{\pi}{4}} \sin x dx = -\cos x \Big|_{\frac{\pi}{6}}^{\frac{\pi}{4}} = -\left(\cos\frac{\pi}{4} - \cos\frac{\pi}{6}\right) = \frac{\sqrt{3}-\sqrt{2}}{2}.$$

Пример 5.2.3

$$\int_3^7 \frac{dx}{x+5} = [\text{Заметим, что } d(x+5) = dx] =$$

$$= \int_3^7 \frac{d(x+5)}{x+5} = \ln|x+5|\Big|_3^7 = \ln(7+5) - \ln(3+5) =$$

$$= \ln 12 - \ln 8 = \ln(\frac{12}{8}) = \ln(\frac{3}{2}).$$

Метод интегрирования по частям в определенном интеграле

Формула интегрирования по частям в этом случае будет иметь вид:

$$\int_a^b udv = uv\Big|_a^b - \int_a^b vdu. \qquad (5.2.7)$$

Рассмотрим конкретные примеры.

Пример 5.2.4

$$\int_0^{\frac{\pi}{2}} \sin^2 x dx = [u = \sin x;\ du = \cos x dx;\ dv = \sin x dx;\ v = \int \sin x dx =$$

$$= -\cos x] = -\sin x \cos x\Big|_0^{\frac{\pi}{2}} + \int_0^{\frac{\pi}{2}} \cos^2 x dx = -(\sin\frac{\pi}{2}\cos\frac{\pi}{2} - \sin 0 \cos 0) +$$

$$+\int_0^{\frac{\pi}{2}}(1-\sin^2 x)dx = 0+\int_0^{\frac{\pi}{2}}dx-\int_0^{\frac{\pi}{2}}\sin^2 x dx.$$

Переносим $\int_0^{\frac{\pi}{2}}\sin^2 x dx$ в левую часть равенства и окончательно получаем:

$$2\int_0^{\frac{\pi}{2}}\sin^2 x dx = x\Big|_0^{\frac{\pi}{2}} \Rightarrow \int_0^{\frac{\pi}{2}}\sin^2 x dx = \frac{\pi}{4}.$$

Пример 5.2.5

$$\int_1^2 x\ln x dx = \left[u=\ln x; du=\frac{dx}{x}; dv=xdx; v=\int xdx=\frac{x^2}{2}\right] = \frac{x^2}{2}\ln x\Big|_1^2 -$$

$$-\frac{1}{2}\int_1^2\frac{x^2dx}{x} = \left(\frac{4}{2}\ln 2-\frac{1}{2}\ln 1\right)-\frac{1}{2}\int_1^2 xdx = 2\ln 2-0-\frac{1}{2}\cdot\frac{x^2}{2}\Big|_1^2 =$$

$$=\ln 4-\frac{1}{4}(2^2-1^2)=\ln 4-\frac{3}{4}.$$

Метод замены переменной в определенном интеграле

Вычисление определенного интеграла методом замены переменной проводится так же, как и при нахождении неопределенного интеграла, за исключением того, что в данном случае нет необходимости возвращаться к первоначальной переменной. Но надо помнить, что, заменяя переменную под знаком интеграла, надо менять и пределы интегрирования.

Решим конкретные примеры.

Пример 5.2.6

$$\int_2^4 \frac{x\,dx}{x^2+8} = \frac{1}{2}\int_2^4 \frac{2x\,dx}{x^2+8} = \text{ [Заметим, что } d(x^2 + 8) = 2x\,dx] =$$

$$\frac{1}{2}\int_2^4 \frac{d(x^2+8)}{x^2+8}.$$

Делаем замену переменной, обозначим $y = x^2 + 8$. Теперь необходимо поменять пределы интегрирования: $y\big|_{x=2}^{=2^2+8=12}$; $y\big|_{x=4}^{=4^2+8=24}$ и окончательно получаем:

$$\frac{1}{2}\int_2^4 \frac{d(x^2+8)}{x^2+8} = \frac{1}{2}\int_{12}^{24}\frac{dy}{y} = \frac{1}{2}\ln|y|\Big|_{12}^{24} = \frac{1}{2}(\ln 24 - \ln 12) =$$

$$= \frac{1}{2}\ln\frac{24}{12} = \frac{1}{2}\ln 2.$$

Пример 5.2.7

$$\int_1^2 x\sqrt{1+x^2}\,dx = \frac{1}{2}\int_1^2 2x\sqrt{1+x^2}\,dx = \text{ [Заметим, что } d(1 + x^2) = 2x\,dx] =$$

$$= \frac{1}{2}\int_1^2 \sqrt{1+x^2}\,d(1+x^2).$$

Теперь заменяем переменную и пределы интегрирования $t = 1 + x^2$;

$t\big|_{x=1}^{=1+1^2=2}$; $t\big|_{x=2}^{=1+2^2=5}$ и окончательно получаем:

$$\frac{1}{2}\int_1^2 \sqrt{1+x^2}\,dx(1+x^2) = \frac{1}{2}\int_2^5 y^{\frac{1}{2}}dy = \frac{1}{2}\cdot\frac{2}{3}y^{\frac{3}{2}}\Big|_2^5 =$$

$$= \frac{1}{3}(\sqrt{5^3} - \sqrt{2^3}) = \frac{1}{3}(5\sqrt{5} - 2\sqrt{2}).$$

Пример 5.2.8

$$\int_0^{\frac{\pi}{4}} \frac{2+\sqrt{tgx}}{\cos^2 x}dx = \left[\text{Заметим, что } dtgx = \frac{dx}{\cos^2 x}\right] = \int_0^{\frac{\pi}{4}} (2+\sqrt{tgx})dtgx.$$

Теперь делаем замену переменной и меняем пределы интегрирования tgx = t, $tg\frac{\pi}{4} = 1$; tg0 = 0 в результате получаем:

$$\int_0^{\frac{\pi}{4}} (2+\sqrt{tgx})dtgx = \int_0^1 (2+\sqrt{t})dt = 2\int_0^1 dt + \int_0^1 t^{\frac{1}{2}}dt =$$

$$= 2t\Big|_0^1 + \frac{2}{3}t^{\frac{3}{2}}\Big|_0^1 = 2+\frac{2}{3} = \frac{8}{3}.$$

5.3. Некоторые сведения о несобственных интегралах

Распространим понятие определенного интеграла на случай бесконечного интервала интегрирования.

Предположим, что функция y = f(x) непрерывна на интервале [a; + ∞). Тогда можно найти интеграл от функции f(x), который взят по любому интервалу [a, b], где b > a.

Интеграл $\int_a^b f(x)dx$ тем лучше выражает значение, которое надо принять в качестве интеграла от функции f(x) в интервале [a, + ∞), чем больше b.

Пусть b неограниченно возрастает, тогда есть две возможности: или $\int_a^b f(x)dx$ при $b \to +\infty$ имеет предел, или данный интеграл предела не имеет, а это означает, что он или стремится к бесконечности, или колеблется, т. е. не стремится ни к какому пределу.

Теперь дадим определение несобственного интеграла.

Несобственным интегралом от функции f(x) в интервале [a, + ∞) называется предел интеграла $\int_a^b f(x)dx$ при $b \to \infty$. Это записывается следующим образом:

$$\int_a^{\infty} f(x)dx = \lim_{b\to\infty} \int_a^b f(x)dx. \qquad (5.3.1)$$

Если предел (5.3.1) существует, то несобственный интеграл называется *сходящимся*, а если не существует, то *расходящимся* [2, 22].

Если первообразная функция F(x) для подынтегральной функции f(x) известна, то можно определить, сходится несобственный интеграл или нет. Используем формулу Ньютона-Лейбница и получим:

$$\int_a^{+\infty} f(x)dx = \lim_{b\to+\infty} (F(b) - F(a)) = F(+\infty) - F(a).$$

Поэтому, если предел первообразной F(x) при $b \to +\infty$ существует, то несобственный интеграл сходится, а если предел не существует, то интеграл расходится.

Аналогично определяется несобственный интеграл в интервале (–∞; b):

$$\int_{-\infty}^b f(x)dx = \lim_{a\to-\infty} \int_a^b f(x)dx = F(b) - F(-\infty).$$

Если функция f(x) определена и непрерывна в интервале (–∞; +∞), то получим:

$$\int_{-\infty}^{+\infty} f(x)dx = \int_{-\infty}^{c} f(x)dx + \int_c^{+\infty} f(x)dx.$$

Если оба интеграла в правой части последнего выражения сходятся, то интеграл $\int_{-\infty}^{+\infty} f(x)dx$ сходится, а если хотя бы один из них расходится, то и $\int_{-\infty}^{+\infty} f(x)dx$ расходится [2, 22].

Если известна первообразная F(x), то

$$\int_{-\infty}^{+\infty} f(x)dx = F(+\infty) - F(-\infty).$$

Сходящиеся несобственные интегралы имеют определенный геометрический смысл. Например, график функции y = f(x) ограничивает криволинейную трапецию с бесконечным основанием (см. рис. 5.3.1).

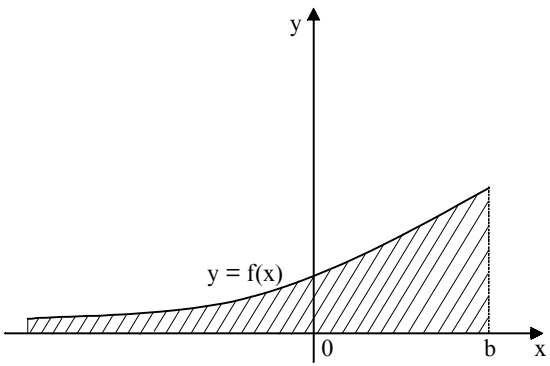


Рис. 5.3.1

Если несобственный интеграл $\int_{-\infty}^{+b} f(x)dx$ сходится, то заштрихованная фигура имеет площадь, которая равна этому интегралу. А если интеграл расходится, то говорить о площади фигуры нельзя.

Теперь приведем конкретные примеры решения несобственных интегралов.

Пример 5.3.1

Вычислим $\int_{0}^{\infty} e^{-2x}dx$ = [Делаем замену переменной –2x = y = >x = $-\frac{y}{2}$ = >dx = $-\frac{dy}{2}$. Затем меняем пределы интегрирования y(0) = 0; y(∞) = –∞. Тогда получим] =

$$\int_0^{\infty} e^{-2x}dx = -\frac{1}{2}\int_0^{-\infty} e^{y}dy = \frac{1}{2}\int_{-\infty}^{0} e^{y}dy = \frac{1}{2}e^{y}\Big|_{-\infty}^{0} = \frac{1}{2}\left(e^{0} - \frac{1}{e^{\infty}}\right) = \frac{1}{2},$$

т. е. несобственный интеграл $\int_0^{\infty} e^{-2x}dx$ сходится и равен $\frac{1}{2}$.

Пример 5.3.2

$\int_5^{\infty} \frac{3dx}{x} = 3\int_5^{\infty} \frac{dx}{x} = \ln x\Big|_5^{\infty} = \ln(\infty) - \ln(5) = \infty$, т. е. данный интеграл расходится.

Пример 5.3.3

$$\int_{\frac{\pi}{4}}^{\infty} \sin x dx = \lim_{b\to\infty}\int_{\frac{\pi}{4}}^{b} \sin x dx = \lim_{b\to\infty}(-\cos x)\Big|_{\frac{\pi}{4}}^{b} = -\lim_{b\to\infty}(\cos b - \frac{\sqrt{2}}{2}).$$

Величина $\left[-\lim_{b\to\infty}\left(\cos b - \frac{\sqrt{2}}{2}\right)\right]$ не стремится к определенному пределу при $b \to \infty$ (колеблется), значит, несобственный интеграл расходится.

Пример 5.3.4

$\int_{-\infty}^{+\infty} 3e^{x}dx = 3e^{x}\Big|_{-\infty}^{+\infty} = 3\left(e^{\infty} - \frac{1}{e^{\infty}}\right) = \infty$, т. е. данный интеграл расходится.

Часто важно знать не конкретное значение несобственного интеграла, а сходится он или расходится. Для этого используются *признаки сравнения*, которые мы и приводим.

1. Если для $\forall x(x \ge a)$ выполняется неравенство $0 \le f(x) \le \varphi(x)$ и если $\int_a^{\infty} \varphi(x)dx$ сходится, то сходится и $\int_a^{\infty} f(x)dx$, при этом выполняется неравенство [2, 22]:

$$\int_a^{\infty} f(x)dx \le \int_a^{\infty} \varphi(x)dx.$$

Например, проверим, сходится ли интеграл $\int_1^{\infty} \frac{dx}{x^2(2+e^x)}$.

При $x \ge 1$, $\frac{1}{x^2(2+e^x)} < \frac{1}{x^2}$.

Теперь рассмотрим, сходится ли несобственный интеграл:

$\int_1^{\infty} \frac{dx}{x^2} = -\frac{1}{x}\Big|_1^{\infty} = -(\frac{1}{\infty} - 1) = 1$, т. е. данный интеграл сходится.

Поэтому по признаку 1 сходится $\int_1^{\infty} \frac{dx}{x^2(2+e^x)}$ и его значения меньше 1.

2. Если для $\forall x(x \ge a)$ выполняется неравенство $0 \le \varphi(x) \le f(x)$, причем $\int_a^{\infty} \varphi(x)dx$ расходится, то расходится и $\int_a^{\infty} f(x)dx$ [2, 22].

Например, проверим сходимость интеграла $\int_1^{\infty} \frac{x+5}{\sqrt{x^3}}dx$.

Очевидно, что $\frac{x+5}{\sqrt{x^3}} > \frac{x}{\sqrt{x^3}} = \frac{1}{\sqrt{x}}$.

Теперь рассмотрим, сходится ли несобственный интеграл

$\int_1^{\infty} \frac{dx}{\sqrt{x}} = \lim_{b\to\infty} \int_1^b x^{-\frac{1}{2}}dx = \lim_{b\to\infty} 2\sqrt{x}\Big|_1^b = 2\lim_{b\to\infty}(\sqrt{b} - 1) = \infty$, т. е. данный интеграл расходится. Поэтому по признаку 2 расходится $\int_1^{\infty} \frac{x+5}{\sqrt{x^3}}dx$.

3. Если несобственный интеграл $\int_a^{\infty} |f(x)|dx$ сходится, то сходится и интеграл $\int_a^{\infty} f(x)dx$. Последний интеграл в этом случае называется абсолютно сходящимся.

В качестве примера проверим сходимость интеграла $\int_1^{\infty} \frac{\sin x}{x^3} dx$.

На интервале $[1; \infty)$ подынтегральная функция $\frac{\sin x}{x^3}$ знакопеременная.

Видно, что $\left|\frac{\sin x}{x^3}\right| \le \left|\frac{1}{x^3}\right|$. Теперь рассмотрим, сходится ли несобственный интеграл

$$\int_1^{\infty} \frac{dx}{x^3} = -\frac{1}{2x^2}\Big|_1^{\infty} = -\frac{1}{2}\left(\frac{1}{\infty} - 1\right) = \frac{1}{2},$$ т. е. данный интеграл сходится.

Поэтому по признаку 1 сходится $\int_1^{\infty} \left|\frac{\sin x}{x^3}\right| dx$, а следовательно, по признаку 3 сходится и интеграл $\int_1^{\infty} \frac{\sin x}{x^3} dx$.

При использовании признаков сравнения надо иметь запас функций, несобственные интегралы от которых или сходятся, или расходятся, и результат этот нам известен заранее. Эти функции мы будем использовать в качестве $\varphi(x)$.

5.4. Некоторые приложения определенного интеграла

5.4.1. Вычисление площадей плоских фигур

Так как определенный интеграл от непрерывной неотрицательной функции равен площади соответствующей криволинейной трапеции, а площадь любой плоской фигуры можно представить как сумму и (или) разность площадей криволинейных трапеций, то, следовательно, определенный интеграл можно использовать для вычисления площадей плоских фигур [2, 16].

Если функция y = f(x) или плоская фигура ABCD находятся выше оси Ox (см. рис 5.4.1 и рис. 5.4.2), то мы имеем $S_1 = \int_a^b f(x)dx$ и

$$S_2 = \int_a^b (f_1(x) - f_2(x))dx.$$

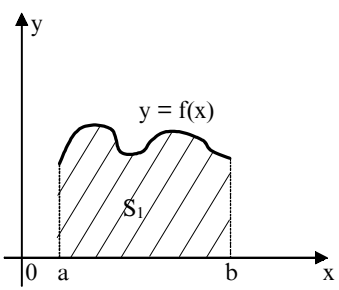


Рис. 5.4.1

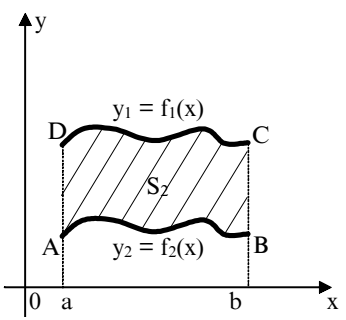


Рис. 5.4.2

Если функция y = f(x) находится полностью или частично под осью 0x (см. рис.5.4.3 и рис. 5.4.4), то мы получаем:

$$S_3 = \left|\int_a^b f(x)dx\right|; \quad S_4 = \int_a^b f(x)dx + \left|\int_b^c f(x)dx\right|.$$

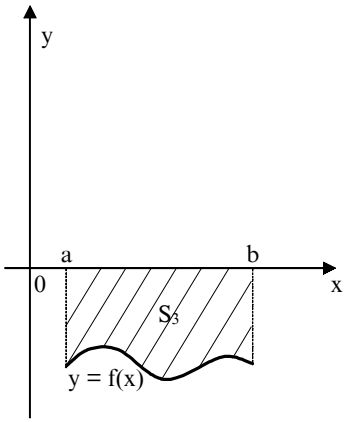


Рис. 5.4.3

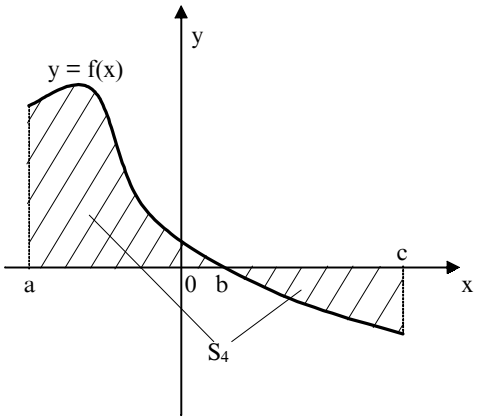


Рис. 5.4.4

Если функция x = φ(y) или плоская фигура ABCD прилегают к оси Oy, то мы имеем (см. рис. 5.4.5 и рис. 5.4.6):

$$S_5 = \int_a^b \varphi(y)dy; \quad S_6 = \int_a^b (\varphi_1(y) - \varphi_2(y))dy.$$

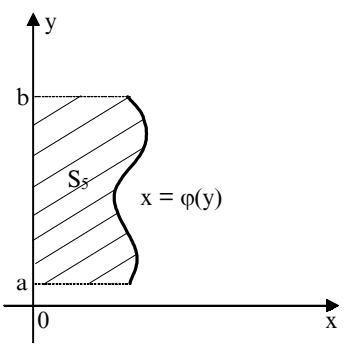


Рис. 5.4.5

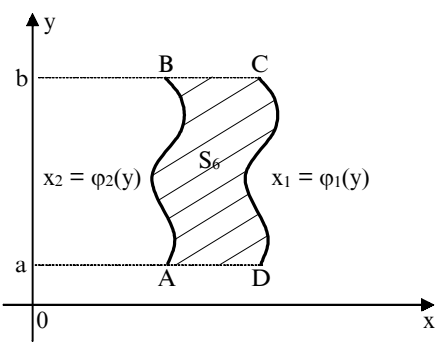


Рис. 5.4.6

Задачи на вычисление площадей плоских фигур можно решать по следующей схеме:

1. В соответствии с условиями задачи делают схематический чертеж.

2. Искомую площадь представляют как сумму и (или) разность площадей криволинейных трапеций.

3. Находят пределы интегрирования.

4. Вычисляют площади каждой криволинейной трапеции и искомую площадь фигуры.

Теперь рассмотрим конкретные примеры вычисления площадей плоских фигур.

Пример 5.4.1

Вычислите площадь фигуры, ограниченной линиями: $x = 4 - y^2$, $x = 0$.

Сначала по условиям задачи строим схематический чертеж (см. рис. 5.4.7).

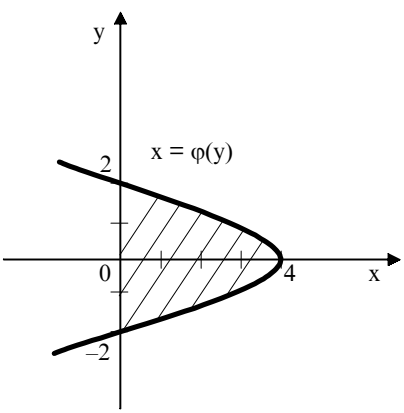


Рис. 5.4.7

$x' = 4 - y^2$ — парабола. Найдем ее вершину: $x' = -2y$, $y = 0$, $x = 4$ (max) и точки пересечения с осью Oy. $4 - y^2 = 0$, $y = 2$, $y = -2$.

$y_1 = -2$ и $y_2 = 2$ являются пределами интегрирования.

Теперь найдем искомую площадь. Так как парабола симметрична относительно оси абцисс, то можно записать.

$$S = \int_{-2}^{2}(4 - y^2)dy = 2\int_{0}^{2}(4 - y^2)dy =$$

$$= 2\left(4y\Big|_0^2 - \frac{y^3}{3}\Big|_0^2\right) = 2\left(8 - \frac{8}{3}\right) = \frac{32}{3}\text{ кв. ед.}$$

Пример 5.4.2

Вычислим площадь фигуры, ограниченной линиями: $y = x^2 - 6x + 8; y = 0$

Построим схематический чертеж искомой фигуры (см. рис. 5.4.8).

Кривая $y = x^2 - 6x + 8$ есть парабола, ветви которой направлены вверх. Найдем ее характерные точки.

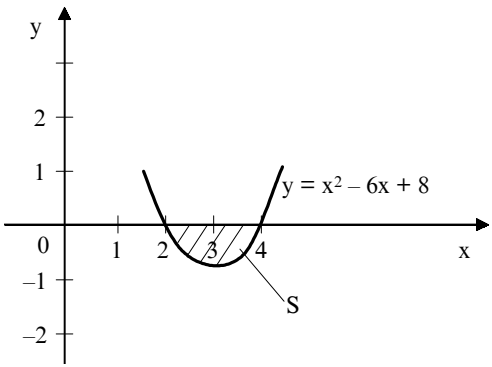


Рис. 5.4.8

$y' = 2x - 6;$

$y' = 0;$

$2x - 6 = 0;$

$x = 3, y = -1(\min).$

Найдем точки пересечения параболы с осью 0x, которые являются пределами интегрирования.

$x^2 - 6x + 8 = 0;$

$D = 36 - 4 \times 1 \times 8 = 4;$

$$x_1 = \frac{6+2}{2} = 4; \quad x_2 = \frac{6-2}{2} = 2.$$

Теперь находим искомую площадь (знак модуля ставится, так как фигура находится под осью Ox).

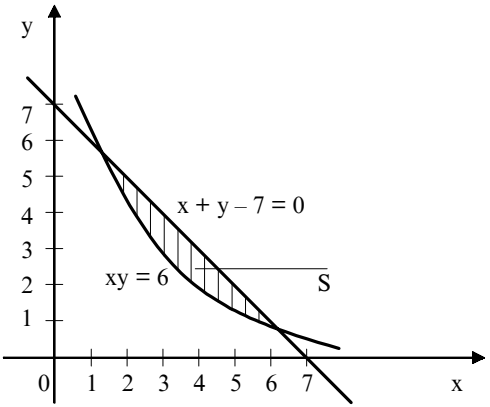
$$S = \left| \int_2^4 (x^2 - 6x + 8)dx \right| = \left| \frac{x^3}{3}\Big|_2^4 - 6\frac{x^2}{2}\Big|_2^4 + 8x\Big|_2^4 \right| =$$

$$= \left| \left(\frac{64}{3} - \frac{8}{3} \right) - 3(16-4) + 8(4-2) \right| = \left| \frac{56}{3} - 36 + 16 \right| = \left| \frac{56}{3} - 20 \right| = \frac{4}{3} \text{ кв. ед.}$$

Пример 5.4.3

Вычислите площадь фигуры, ограниченной линиями: $xy = 6$, $x + y - 7 = 0$.

Построим схематический чертеж (см. 5.4.9) и найдем пределы интегрирования.



$$\frac{6}{x} = 7 - x$$

$6 - 7x + x^2 = 0,\ x \neq 0$

$D = 25;$

$$\begin{cases} x_1 = 1 \\ x_2 = 6 \end{cases}$$

(пределы интегрирования).

Рис. 5.4.9

Теперь находим искомую площадь.

$$S = \int_1^6 \left(7 - x - \frac{6}{x}\right) dx = 7x\Big|_1^6 - \frac{x^2}{2}\Big|_1^6 - 6\ln x\Big|_1^6 =$$

$$= (42 - 7) - \left(18 - \frac{1}{2}\right) - 6(\ln 6 - \ln 1) = 35 - 17{,}5 - 6\ln 6 =$$

$$= (17{,}5 - 6\ln 6) \text{ кв. ед.}$$

5.4.2. Вычисление длины дуги

Пусть в плоскости xOy уравнением y = f(x) задана кривая линия. Вычислим длину дуги AB этой кривой, заключенной между прямыми x = a и x = b (см. рис. 5.4.10).

На дуге AB возьмем точки A, A_1, A_2, ..., A_i, ..., B с абсциссами $x_0 = a$, x_1, x_2, ..., x_i, ..., $b = x_n$. Проведем хорды AA_1, A_1A_2, ..., $A_{n-1}B$, длины которых соответственно обозначим Δl_1, Δl_2, ..., Δl_n. В результате получим ломаную A, A_1A_2, ..., A_{n-1} B, которая вписана в дугу AB. Длина этой ломаной будет равна $l_n = \sum_{i=1}^{n} \Delta l_i$. А длиной (l)

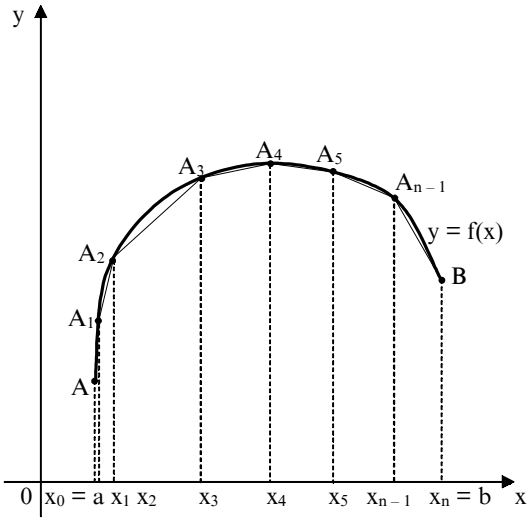


Рис. 5.4.10

дуги AB называется предел, к которому стремится длина вписанной ломаной, когда длина ее наибольшего звена стремится к нулю, т. е.

$$l = \lim_{\max \Delta l_i \to 0} \sum_{i=1}^{n} \Delta l_i .$$

В курсах математического анализа доказывается (см. например, [2, 20]), что если на отрезке [a, b] функция f(x) и ее производная f'(x) непрерывны, то этот предел существует и длина дуги AB находится по формуле

$$l = \int_a^b \sqrt{1 + [f'(x)]^2}\, dx = \int_a^b \sqrt{1 + \left(\frac{dy}{dx}\right)^2}\, dx. \qquad (5.4.1)$$

Рассмотрим конкретный пример.

Пример 5.4.4

Найдите длину дуги кривой $y = \frac{x^2}{2} - \frac{3}{2}$, отсеченной осью Ox.

Сначала построим график исходной функции и найдем a и b (см. рис. 5.4.11)

$y' = x$.

$$\min x = 0; y = -\frac{3}{2}.$$

Находим a и b.

$$\frac{x^2}{2} - \frac{3}{2} = 0$$

$$x^2 = 3$$

$$x = \pm\sqrt{3}$$

Теперь по формуле (5.4.1) находим искомую длину дуги.

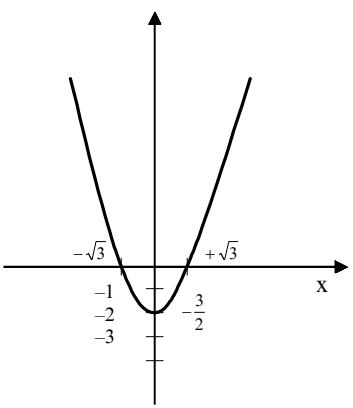


Рис. 5.4.11

$$l = \int_{-\sqrt{3}}^{\sqrt{3}} \sqrt{1+x^2}\,dx =$$ [Так как исходная парабола симметрична относительно оси Oy, то получаем] $= 2\int_{0}^{\sqrt{3}} \sqrt{1+x^2}\,dx.$

Найдем $l_1 = \int_{0}^{\sqrt{3}} \sqrt{1+x^2}\,dx = \frac{l}{2}.$

Получившийся определенный интеграл можно брать несколькими способами, например, подстановкой $x = \operatorname{sh} y$, где $\operatorname{sh} y = \frac{e^y + e^{-y}}{2}$ — гиперболический синус, или методом интегрирования по частям, которым мы и воспользуемся. Напомним, что формула интегрирования по частям имеет вид:

$$\int_a^b u\,dv = u\,v\Big|_a^b - \int_a^b v\,du.$$

В нашем случае имеем

$$u = \sqrt{1+x^2};\; dv = dx \Rightarrow v = x;$$

$$du = \frac{x}{\sqrt{1+x^2}}\,dx.$$

Тогда получим:

$$l_1 = \int_0^{\sqrt{3}} \sqrt{1+x^2}\,dx = \left(x\sqrt{1+x^2}\Big|_0^{\sqrt{3}} - \int_0^{\sqrt{3}} \frac{x^2\,dx}{\sqrt{1+x^2}} \right) =$$

$$= \left(2\sqrt{3} - \int_0^{\sqrt{3}} \frac{x^2+1-1}{\sqrt{1+x^2}}\,dx \right) = \left(2\sqrt{3} - \int_0^{\sqrt{3}} \frac{x^2+1}{\sqrt{1+x^2}} + \int_0^{\sqrt{3}} \frac{dx}{\sqrt{1+x^2}} \right) =$$

$$= 2\sqrt{3} - \int_0^{\sqrt{3}} \sqrt{1+x^2}\,dx + \int_0^{\sqrt{3}} \frac{dx}{\sqrt{1+x^2}}.$$

(Интеграл $\int \frac{dx}{\sqrt{1+x^2}}$ является табличным (см. формулу 17 раздела 5.1). Он равен:

$$\int \frac{dx}{\sqrt{x^2+1}} = \operatorname{Arsh} x + C = \ln\left| x + \sqrt{x^2+1} \right| + C.$$

В нашем случае получаем:

$$\int_0^{\sqrt{3}} \frac{dx}{\sqrt{1+x^2}} = \ln\left| x + \sqrt{1+x^2} \right|\Big|_0^{\sqrt{3}} = \ln(\sqrt{3}+2).$$

Поэтому l_1 примет вид:

$$l_1 = 2\sqrt{3} - \int_0^{\sqrt{3}} \sqrt{1+x^2}\,dx + \ln(\sqrt{3}+2).$$

Переносим $\left(-\int_0^{\sqrt{3}} \sqrt{1+x^2}\,dx \right)$ налево и окончательно получаем:

$$l = 2l_1 = 2\int_0^{\sqrt{3}} \sqrt{1+x^2}\,dx = 2\sqrt{3} + \ln(\sqrt{3}+2).$$

5.5. Приближенное вычисление определенных интегралов

В тех случаях, когда подынтегральная функция имеет сложный вид и неясно, как ее преобразовать к табличной, или же первообразная подынтегральной функции не выражается через элементарные функции, применяют приближенные методы вычисления определенных интегралов.

Приведем несколько способов приближенного интегрирования, исходя из определения интеграла как предела суммы.

1. Формула прямоугольников

Пусть на [a, b] задана непрерывная функция y = f(x). Надо вычислить $\int_a^b f(x)dx$. Отрезок [a, b] разделим точками $a = x_0, x_1, x_2, ..., x_{n-1}, x_n = b$ на n одинаковых частей длины Δx, где $\Delta x = \frac{b-a}{n}$ (см. рис. 5.5.1).

Значения функции y = f(x) в точках $x_0, x_1, x_2, ..., x_{n-1}, x_n$ обозначим через $y_0, y_1, y_2, ..., y_{n-1}, y_n$.

Теперь составим две суммы:

$$S_1 = y_0 \Delta x + y_1 \Delta x + y_2 \Delta x + ... + y_{n-1} \Delta x. \qquad (5.5.1)$$

S_1 есть суммарная площадь прямоугольников, лежащих ниже y = f(x).

$$S_2 = y_1 \Delta x + y_2 \Delta x + ... + y_n \Delta x. \qquad (5.5.2)$$

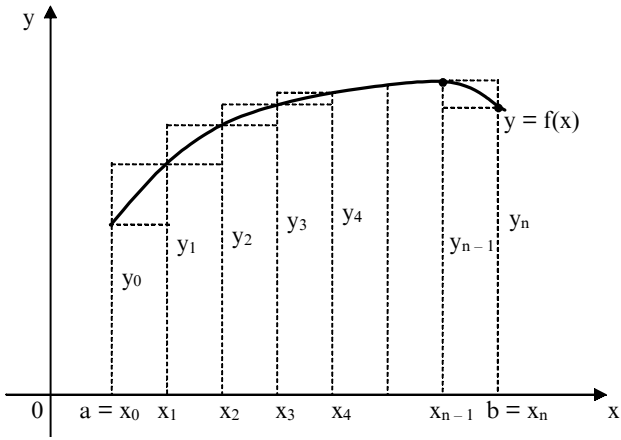


Рис. 5.5.1

S_2 есть суммарная площадь прямоугольников, лежащих выше $y = f(x)$.

Истинная площадь фигуры, ограниченная $y = f(x)$, удовлетворяет условию $S_1 < S_{ист} < S_2$.

Поэтому можно записать приближенные равенства [1, 16]:

$$\int_a^b f(x)dx \approx \frac{b-a}{n}(y_0 + y_1 + y_2 + ... + y_{n-1}). \qquad (5.5.3)$$

$$\int_a^b f(x)dx \approx \frac{b-a}{n}(y_1 + y_2 + y_3 + ... + y_n). \qquad (5.5.4.)$$

Приближенные равенства (5.5.3) и (5.5.4) и есть формулы прямоугольников. Ошибка, которую мы совершаем при вычислении интегралов по формулам (5.5.3) и (5.5.4), будет тем меньше, чем больше n. Для того чтобы определить, сколько точек деления надо взять, чтобы вычислить интеграл с заданной точностью, надо использовать формулу оценки погрешности, которая получается при приближенном вычислении интеграла. Для метода прямоугольников она имеет вид [9]:

$$|\alpha_n| \le \frac{(b-a)^2 M}{2n},$$

где $M = \max\limits_{a \le x \le b} |f'(x)|$.

Приведем конкретный пример.

Пример 5.5.1

Используя метод прямоугольников, вычислим приближенно интеграл

$$\int_2^6 \frac{dx}{\ln x},$$ взяв n = 10.

Заметим, что этот интеграл относится к числу неберущихся, т. е. он не выражается в элементарных функциях.

$$y = f(x) = \frac{1}{\ln x}$$

$$\frac{b-a}{n} = \frac{6-2}{10} = 0{,}4.$$

Используем для расчета формулу (5.5.3).

$$y_0 = \frac{1}{\ln 2} = 1{,}443;\ y_1 = \frac{1}{\ln 2{,}4} = 1{,}142;\ y_2 = \frac{1}{\ln 2{,}8} = 0{,}971;$$

$$y_3 = \frac{1}{\ln 3{,}2} = 0{,}860;\ y_4 = \frac{1}{\ln 3{,}6} = 0{,}781;\ y_5 = \frac{1}{\ln 4} = 0{,}721;$$

$$y_6 = \frac{1}{\ln`4{,}4} = 0{,}675;\ y_7 = \frac{1}{\ln 4{,}8} = 0{,}638;\ y_8 = \frac{1}{\ln 5{,}2} = 0{,}607;$$

$$y_9 = \frac{1}{\ln 5{,}6} = 0{,}581.$$

Теперь по формуле (5.5.3) имеем:

$$S_1 = \int_2^6 \frac{dx}{\ln x} \approx 0{,}4(1{,}443 + 1{,}142 + 0{,}971 + 0{,}860 + 0{,}781 + 0{,}721 +$$

$$+ 0{,}675 + 0{,}638 + 0{,}607 + 0{,}581) \approx 3{,}369.$$

2. Формула трапеций

Более точное значение определенного интеграла, чем по (5.5.3) и (5.5.4), мы получим, заменив исходную функцию y = f(x) ломаной линией (см. рис. 5.5.2).

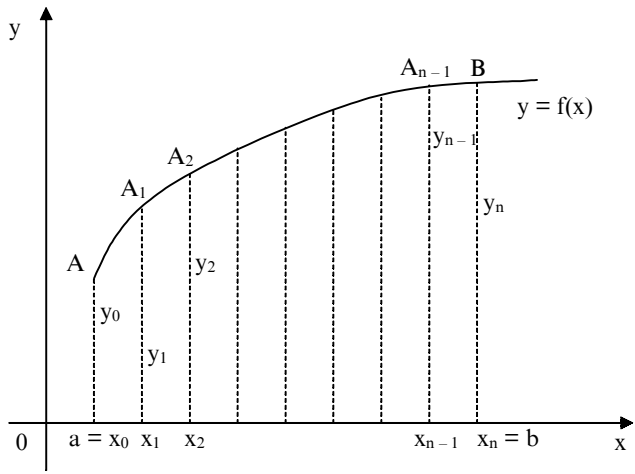


Рис. 5.5.2

Таким образом, площадь криволинейной трапеции aABb мы заменим площадью фигуры, состоящей из прямоугольных трапеций: aAA_1x_1, $x_1A_1A_2x_2$, ..., $x_{n-1}A_{n-1}Bb$. Их площади будут равны:

$$\frac{y_0+y_1}{2}\Delta x;\ \frac{y_1+y_2}{2}\Delta x;\ldots;\ \frac{y_{n-1}+y_n}{2}\Delta x.$$

Поэтому определенный интеграл приближенно будет равен [1, 16]:

$$\int_a^b f(x)dx \approx \left(\frac{y_0+y_1}{2}\Delta x+\frac{y_1+y_2}{2}\Delta x+\cdots+\frac{y_{n-1}+y_n}{2}\Delta x\right)$$

или

$$\int_a^b f(x)dx \approx \frac{b-a}{n}\left(\frac{y_0+y_n}{2}\Delta x+y_1+y_2+\cdots+y_{n-1}\right). \qquad (5.5.5)$$

Формула (5.5.5) носит название формулы трапеций.

Формула оценки погрешности, получающейся при приближенном вычислении интеграла, в этом случае имеет вид [9]:

$$|\alpha_n| \le \frac{(b-a)^3 M_1}{12n^2},$$

где $M = \max\limits_{a \le x \le b} |f''(x)|$.

Приведем конкретный пример вычисления определенного интеграла по формуле (5.5.5.)

Пример 5.5.2

Используя метод трапеции, приближенно вычислим интеграл $\int\limits_1^4 \frac{e^x}{x} dx$, приняв n = 10. Этот интеграл, как и интеграл предыдущего примера, является неберущимся. В данном случае $y = f(x) = \frac{e^x}{x}$; $\frac{b-a}{n} = \frac{4-1}{10} = 0{,}3$.

$$y_0 = \frac{e^1}{1} = e \approx 2{,}718;\ y_1 = \frac{e^{1{,}3}}{1{,}3} \approx 2{,}823;\ y_2 = \frac{e^{1{,}6}}{1{,}6} \approx 3{,}096;$$

$$y_3 = \frac{e^{1{,}9}}{1{,}9} \approx 3{,}519;\ y_4 = \frac{e^{2{,}2}}{2{,}2} \approx 4{,}102;\ y_5 = \frac{e^{2{,}5}}{2{,}5} \approx 4{,}873;$$

$$y_6 = \frac{e^{2{,}8}}{2{,}8} \approx 5{,}873;\ y_7 = \frac{e^{3{,}1}}{3{,}1} \approx 7{,}161;\ y_8 = \frac{e^{3{,}4}}{3{,}4} \approx 8{,}813;$$

$$y_9 = \frac{e^{3{,}7}}{3{,}7} \approx 10{,}932;\ y_{10} = \frac{e^4}{4} \approx 13{,}650.$$

Теперь по формуле (5.5.5) получаем:

$$\int\limits_1^4 \frac{e^x}{x} dx \approx 0{,}3\left(\frac{2{,}718 + 13{,}650}{2} + 2{,}823 + \cdots + 10{,}932\right) \approx 17{,}813.$$

3. Формула парабол (формула Симпсона)

Пусть на отрезке [a, b] задана непрерывная функция y = f(x). Разделим [a, b] на четное число частей n = 2m. Сущность способа заключается в том, что отрезки прямых, ограничивающих элементарные трапеции сверху, заменяют дугами парабол, оси которых параллельны оси 0y (см. рис. 5.5.3).

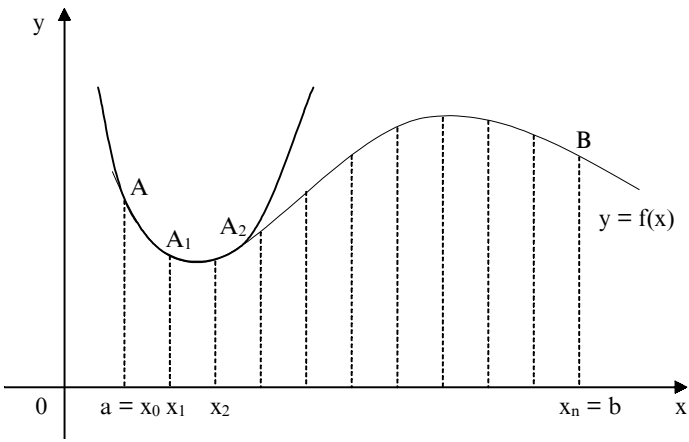


Рис. 5.5.3

Уравнения таких парабол имеют вид $y = cx^2 + dx + p$. Коэффициенты c, d, p можно однозначно найти по трем точкам, если абсциссы их различны. Дуги парабол проводят через каждую тройку точек. Криволинейную трапецию aABb заменяют суммой площадей криволинейных трапеций, ограниченных дугами парабол. Площадь первой из таких параболических трапеций равна: $S_1 = \frac{\Delta x}{3}(y_0 + 4y_1 + y_2)$, площадь второй равна: $S_2 = \frac{\Delta x}{3}$ $(y_1 + 4y_2 + y_3)$ и т. д.

Искомая формула Симпсона имеет вид [1, 16]:

$$\int_a^b f(x)dx \approx \frac{b-a}{3n}\,[y_0 + y_n + 2(y_2 + y_4 + \ldots y_{n-2}) + 4(y_1 + y_3 + \ldots + y_{n-1})]. \qquad (5.5.6)$$

Формула оценки погрешности, получающейся при приближенном вычислении интеграла, в этом случае имеет вид[9]:

$$|\alpha_n| \le \frac{(b-a)^5 M_2}{180(2n)^4},$$

где $M_2 = \max\limits_{a \le x \le b} \left| f^{(4)}(x) \right|$.

Применение формулы (5.5.6) значительно повышает точность вычисления определенного интеграла.

Метод прямоугольников является наиболее простым и наименее точным способом.

Выбор способа приближенного интегрирования зависит от подынтегральной функции и требуемой точности расчета.

Приведем конкретный пример вычисления определенного интеграла по формуле (5.5.6).

Пример 5.5.3

Используя формулу Симпсона, приближенно вычислим интеграл $\int\limits_{\frac{\pi}{4}}^{\frac{\pi}{2}} \frac{\cos x}{x} dx$, приняв n = 10. Заметим, что этот интеграл, как и интегралы из двух предшествующих примеров, является неберущимся. В данном случае $y = f(x) = \frac{\cos x}{x}$; $\frac{b-a}{3n} = \frac{\frac{\pi}{2} - \frac{\pi}{4}}{30} = \frac{\pi}{120}$; $\frac{b-a}{n} = \frac{\frac{\pi}{2} - \frac{\pi}{4}}{10} = \frac{\pi}{40}$

$$y_0 = \frac{\cos\frac{\pi}{4}}{\frac{\pi}{4}} \approx 0{,}9003; \quad y_1 = \frac{\cos\frac{11\pi}{40}}{\frac{11\pi}{40}} \approx 0{,}752; \quad y_2 = \frac{\cos\frac{12\pi}{40}}{\frac{12\pi}{40}} \approx 0{,}624;$$

$$y_3 = \frac{\cos\frac{13\pi}{40}}{\frac{13\pi}{40}} \approx 0{,}512;\ y_4 = \frac{\cos\frac{14\pi}{40}}{\frac{14\pi}{40}} \approx 0{,}413;\ y_5 = \frac{\cos\frac{15\pi}{40}}{\frac{15\pi}{40}} \approx 0{,}325;$$

$$y_6 = \frac{\cos\frac{16\pi}{40}}{\frac{16\pi}{40}} \approx 0{,}246;\ y_7 = \frac{\cos\frac{17\pi}{40}}{\frac{17\pi}{40}} \approx 0{,}175;\ y_8 = \frac{\cos\frac{18\pi}{40}}{\frac{18\pi}{40}} \approx 0{,}111;$$

$$y_9 = \frac{\cos\frac{19\pi}{40}}{\frac{19\pi}{40}} \approx 0{,}053;\ y_{10} = \frac{\cos\frac{\pi}{2}}{\frac{\pi}{2}} = 0.$$

По формуле (5.5.6) получим:

$$\int_{\frac{\pi}{4}}^{\frac{\pi}{2}} \frac{\cos x}{x}dx \approx \frac{\pi}{120}\ [0{,}9003 + 0 + 2(0{,}624 + 0{,}413 + 0{,}246 + 0{,}111) +$$

$$+ 4(0{,}752 + 0{,}512 + 0{,}325 + 0{,}175 + 0{,}053)] \approx 0{,}287.$$

Задачи для самостоятельного решения

1. Методом непосредственного интегрирования найти интегралы:

1.1. $\int \frac{x^4 + 2x}{x}dx$; 1.2. $\int\left(3x^3 + \frac{7}{x}\right)dx$; 1.3. $\int(8x^3 + 3\sin x)dx$;

1.4. $\int \frac{\sqrt{1-x^2}}{(1-x)(1+x)}dx$.

2. Найти интегралы, использовав метод замены переменной:

2.1. $\int 3x(6x^2 - 7)^5dx$; 2.2. $\int 5\operatorname{tg}x dx$; 2.3. $\int \frac{3xdx}{\sqrt{x+5}}$; 2.4. $\int \frac{7dx}{x + \sqrt{x}}$.

3. Найти интегралы, использовав метод интегрирования по частям:

3.1. $\int \operatorname{arctg} x\,dx$; 3.2. $\int 5\arcsin x\,dx$; 3.3. $\int x^2 \sin x\,dx$; 3.4. $\int x^3 e^{-x}\,dx$.

4. Вычислить определенные интегралы:

4.1. $\int_0^{\frac{\pi}{3}} 2\operatorname{tg} x\,dx$; 4.2. $\int_1^2 \frac{dx}{2x-5}$; 4.3. $\int_0^{\frac{\pi}{2}} \cos^2 x\,dx$; 4.4. $\int_0^{\frac{\pi}{2}} 3\sin x \cos^2 x\,dx$;

4.5. $\int_1^5 \frac{\sqrt{x-1}}{3x}dx$; 4.6. $\int_2^8 \frac{\ln x}{x^2}dx$; 4.7. $\int_0^{\frac{\pi}{4}} e^x \sin x\,dx$; 4.8. $\int_{\frac{\pi}{6}}^{\frac{\pi}{2}} \frac{6x\,dx}{\sin^2 x}$.

5. Исследовать на сходимость несобственные интегралы:

5.1. $\int_0^{+\infty} (5x+1)dx$; 5.2. $\int_1^{+\infty} \frac{2dx}{x^5}$; 5.3. $\int_{-\infty}^{+\infty} \frac{dx}{1+x^2}$; 5.4. $\int_{-\infty}^{+\infty} \frac{2x\,dx}{x^2+1}$.

6. Найти площадь фигуры, ограниченной кривой $y = x^3$, прямой $y = 10$ и осью 0y.

7. Вычислить площади фигур, ограниченных линиями:

7.1. $y = x^2$, $y^2 = x$; 7.2 $y = \ln x$, $x = e$, $y = 0$; 7.3. $y^3 = x^2$, $y = 1$; 7.4. $y = x^2$, $y = 2x^2 - 1$.

8. Вычислить длину дуги полукубической параболы $y^2 = x^3$, отсекаемой прямой $x = 5$.

9. Найти длину дуги кривой $y = 2\sqrt{x}$ от $x = 0$ до $x = 1$.

10. Найти длину дуги кривой $x = \frac{2}{3}y^{\frac{3}{2}}$ от $y = 0$ до $y = 3$.

11. Определить длину окружности $x^2 + y^2 = 25$.

12. Используя формулу прямоугольников, вычислить интеграл $\int_1^7 \frac{dx}{\sqrt{1+x^3}}$, приняв $n = 10$.

13. Используя формулу трапеции, вычислить интеграл $\int_0^6 e^{-x^2} dx$, приняв n = 10.

14. Используя формулу Симпсона, вычислить интеграл $\int_0^{\frac{\pi}{2}} \frac{\sin x}{x} dx$, приняв n = 10.

Вопросы для самопроверки

1. Какая функция называется первообразной?
2. В чем состоит суть метода интегрирования по частям?
3. В чем состоит суть метода замены переменной?
4. Каков геометрический смысл определенного интеграла?
5. В чем состоит суть метода замены переменной в определенном интеграле?
6. В каких случаях применяют приближенные методы интегрирования?
7. В чем заключается суть признаков сходимости несобственных интегралов с бесконечными пределами?

6. Дифференциальные уравнения

6.1. Основные понятия и определения

Дифференциальными называются уравнения, которые содержат искомую функцию, ее производные и (или) дифференциалы различных порядков, независимые *переменные* [22].

Теория дифференциальных уравнений появилась в конце XVIII века в результате решения некоторых задач механики и физики. Термин “дифференциальные уравнения” ввел Г. Лейбниц.

Дифференциальные уравнения делятся на дифференциальные уравнения в частных производных, неизвестная функция в которых зависит от двух и большего количества неизвестных, и на обыкновенные дифференциальные уравнения, неизвестная функция в которых зависит от одного аргумента.

В данном пособии мы кратно рассмотрим обыкновенные дифференциальные уравнения.

Общий вид обыкновенного дифференциального уравнения следующий [2, 22]:

$$F(x, y, y', y'', ..., y^{(n)}) = 0 \text{ или } F(x, y, \frac{dy}{dx}, \frac{d^2y}{dx^2}, ..., \frac{d^ny}{dx^n}) = 0.$$

Наивысший порядок производных, входящих в дифференциальное уравнение, называется его порядком.

Например, $2xy'' + 5xy' - 7y = 0$ это дифференциальное уравнение второго порядка.

Решить дифференциальное уравнение — это значит найти такую функцию, подстановка которой в это дифференциальное уравнение превращает его в тождество [16].

Любое дифференциальное уравнение имеет бесконечное множество решений. Геометрически они изображаются семейством

интегральных кривых. И эту совокупность решений называют общим решением дифференциального уравнения и записывают так [2, 22]:

$y = \varphi(x, C_1, C_2, ..., C_n)$.

А решения, содержащие конкретные значения постоянных, называются частными решениями дифференциальных уравнений.

6.2. Дифференциальные уравнения первого порядка

Дифференциальное уравнение первого порядка — это уравнение, связывающее независимую переменную, неизвестную функцию, ее производную и (или) дифференциал [2].

Его общий вид следующий:

$$F(x, y, y') = 0 \text{ или } F(x, y, \frac{dy}{dx}) = 0.$$

Если это уравнение можно разделить относительно производной (y'), то оно примет вид:

$y' = f(x, y)$.

Общее решение этого дифференциального уравнения имеет следующий вид:

$y = \varphi(x, C)$.

Для того чтобы получить конкретные частные решения, надо задать начальные условия, т. е. указать пару соответствующих друг другу значений аргумента (x_0) и функции (y_0). Обычно это записывается так: $y|_{x = x_0} = y_0$ [2, 22].

Задавая начальные условия, мы из семейства интегральных кривых выделяем какую-то конкретную кривую.

Вопрос о том, в каком случае можно утверждать, что частное решение дифференциального уравнения, удовлетворяющее заданному начальному условию, существует, а также является единственным, становится ясным из ТЕОРЕМЫ 6.1: *если в дифференциальном уравнении $y' = f(x, y)$ функция $f(x, y)$ и ее частная производная*

по $y\left(\frac{\partial f(x,y)}{\partial y}\right)$ *непрерывны в некоторой области D на плоскости xOy, содержащей точку* (x_0, y_0)*, то существует единственное решение этого дифференциального уравнения* $y = \varphi(x)$*, удовлетворяющее условию при* $x = x_0$ *и* $y = y_0$ [22].

Приведенная теорема была впервые сформулирована и доказана Коши. Поэтому задачу нахождения частного решения по заданным начальным условиям называют задачей Коши.

6.2.1. Дифференциальные уравнения первого порядка с разделяющимися переменными

В общем случае они имеют вид: $f_1(x)f_2(y)dx + f_3(x)f_4(y)dy = 0$.

Разделим обе части этого дифференциального уравнения на произведение $f_2(y)f_3(x)$, предполагая, что оно не равно нулю.

$$\frac{f_1(x)f_2(y)dx}{f_2(y)f_3(x)} + \frac{f_3(x)f_4(y)dy}{f_2(y)f_3(x)} = 0.$$

Далее получаем:

$$\frac{f_1(x)}{f_3(x)}dx = -\frac{f_4(y)}{f_2(y)}dy.$$

В полученном дифференциальном уравнении при dx стоит только функция от x, а при dy стоит только функция от y, т. е. переменные разделены. Интегрируем левую и правую части последнего равенства и получаем:

$$\int \frac{f_1(x)}{f_3(x)}dx = -\int \frac{f_4(y)}{f_2(y)}dy + C.$$

Это и есть общий интеграл исходного дифференциального уравнения [2, 6].

Рассмотрим несколько конкретных задач.

Пример 6.1. Найти частное решение дифференциального уравнения $xdx + ydy = 0$, если начальное условие таково: $y\big|_{x=2.}^{=10}$

$ydy = -xdx$,

$$\int ydy = -\int xdx \Rightarrow \frac{y^2}{2} = -\frac{x^2}{2} + C_1,$$

$y^2 + x^2 = 2C_1$.

Так как постоянная может быть любой, то примем $2C_1 = C^2$. Тогда получим общее решение исходного дифференциального уравнения $y^2 + x^2 = C^2$.

С геометрической точки зрения это решение представляет собой семейство концентрических окружностей с центром в начале координат и радиусом C (см. рис. 6.1).

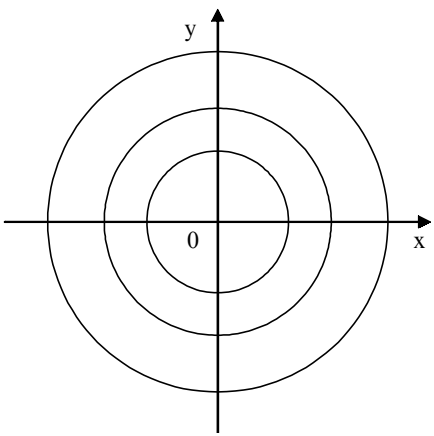


Рис. 6.1

Найдем теперь частное решение для заданных начальных условий, т. е. выделим из семейства окружностей одну. Получим $10^2 + 2^2 = C^2 \Rightarrow >C^2 = 104$, $C = \sqrt{104}$.

Поэтому частное решение имеет вид: $y^2 + x^2 = 104$.

Пример 6.2. Найти общее решение дифференциального уравнения:

$1 + y' + y + xy' = 0$.

Перепишем его в виде:

$$1 + \frac{dy}{dx} + y + x\frac{dy}{dx} = 0;$$

$$(1 + y) + \frac{dy}{dx}(1 + x) = 0.$$

Правую и левую части домножаем на dx:

$(1 + y)dx + dy(1 + x) = 0$.

Правую и левую части делим на $(1 + x) \neq 0$:

$$\frac{(1+y)dx}{1+x}+dy=0.$$

Правую и левую части делим на $(1 + y) \neq 0$:

$$\frac{dx}{1+x}+\frac{dy}{1+y}=0;$$

$$\frac{dy}{1+y}=-\frac{dx}{1+x}.$$

Теперь интегрируем правую и левую части:

$$\int\frac{dy}{1+y}=-\int\frac{dx}{1+x};$$

$$\int\frac{d(1+y)}{1+y}=-\int\frac{d(1+x)}{1+x};$$

$$\ln|1+y|=-\ln|1+x|+\ln C;$$

$$\ln|1+y|=\ln\left|\frac{C}{1+x}\right|;$$

$$1+y=\frac{C}{1+x};$$

$$y=\frac{C}{1+x}-1.$$

Полученное выражение и есть общее решение исходного дифференциального уравнения.

Пример 6.3. Найти общее решение дифференциального уравнения:

$2xyy' = y^2 - 1$.

Перепишем его в виде:

$$2xy\frac{dy}{dx}=y^2-1.$$

Домножим правую и левую части на dx:

$2xydy = (y^2 - 1)dx$.

Разделим правую и левую части на $x \neq 0$:

$$2ydy = \frac{(y^2 - 1)dx}{x}.$$

Разделим правую и левую части на $y^2 - 1 \neq 0$:

$$\frac{2ydy}{y^2 - 1} = \frac{dx}{x}.$$

Теперь проинтегрируем правую и левую части полученного выражения:

$$\int \frac{2ydy}{y^2 - 1} = \int \frac{dx}{x};$$

$$\int \frac{d(y^2 - 1)}{y^2 - 1} = \ln|x| + \ln C;$$

$\ln|y^2 - 1| = \ln|xC|;$
$y^2 - 1 = xC;$
$y^2 = xC + 1;$

$$y = \sqrt{xC + 1}.$$

Полученное уравнение и есть общее решение исходного дифференциального уравнения.

Пример 6.4. Найти частое решение дифференциального уравнения $(x^2 + 4)y' - 2xy = 0$, если задано следующее начальное условие: $y\big|_{x=1}^{=5}$.

Перепишем исходное дифференциальное уравнение так:

$$(x^2 + 4)\frac{dy}{dx} - 2xy = 0.$$

Домножим правую и левую части на dx:
$(x^2 + 4)dy - 2xydx = 0.$
Разделим правую и левую части на $x^2 + 4$:

$$dy - \frac{2xydx}{x^2 + 4} = 0.$$

Разделим правую и левую части на $y \neq 0$:

$$\frac{dy}{y} - \frac{2xdx}{x^2+4} = 0;$$

$$\frac{dy}{y} = \frac{2xdx}{x^2+4}.$$

Теперь интегрируем обе части полученного выражения:

$$\int \frac{dy}{y} = \int \frac{2xdx}{x^2+4};$$

$$\ln|y| = \int \frac{d(x^2+4)}{x^2+4};$$

$\ln|y| = \ln|x^2 + 4| + \ln C;$
$\ln|y| = \ln|C(x^2 + 4)|;$
$y = C(x^2 + 4).$

Полученное уравнение и есть общее решение исходного дифференциального уравнения. Геометрически оно представляет собой семейство парабол.

По заданным начальным условиям найдем частное решение, т. е. выделим конкретную параболу из полученного семейства.

$5 = C(1^2 + 4) \Rightarrow > 5 = 5C \Rightarrow > C = 1.$

Поэтому частное решение имеет вид: $y = x^2 + 4$.

Рассмотрим некоторые классы дифференциальных уравнений, которые сводятся к дифференциальным уравнениям с разделяющимися переменными.

6.2.2. Однородные дифференциальные уравнения

Функция $f(x, y)$ называется однородной функцией n-го измерения относительно аргументов x и y, если при любом k справедливо *равенство* [22]

$f(kx, ky) = k^n f(x, y).$

Например, функция $f(x, y) = 2xy - 3y^2$ является однородной функцией второго измерения, так как

$2 (kx)(ky) - 3(ky)^2 = k^2(2xy - y^2).$

А функция $f(x,y)=\dfrac{x^2-y^2}{xy}$ есть однородная функция нулевого измерения, так как $\dfrac{(kx)^2-(ky)^2}{(kx)(ky)}=\dfrac{x^2-y^2}{xy}$, т. е. f(kx, ky) = f(x, y).

Дифференциальное уравнение первого порядка

$$\frac{dy}{dx}=f(x,y) \tag{6.1}$$

называется однородным относительно x и y, если функция f(x, y) является однородной функцией нулевого измерения относительно x и y.

По условию имеем f(kx, ky) = f(x, y), положим $k=\dfrac{1}{x}$, тогда получим $f(x,y)=f\left(1,\dfrac{y}{x}\right)$, т. е. однородная функция нулевого измерения зависит только от отношения аргументов. Тогда дифференциальное уравнение (6.1) примет вид:

$$\frac{dy}{dx}=f\left(1,\frac{y}{x}\right). \tag{6.2}$$

Сделаем замену переменных, обозначив $u=\dfrac{y}{x}$, т. е. y = ux, тогда $y'=u'x+u=\dfrac{du}{dx}x+u$.

После подстановки дифференциальное уравнение (6.2.) примет вид:

$$\frac{du}{dx}x=f(1,u)-u, \tag{6.3}$$

т. е. мы пришли к дифференциальному уравнению с разделяющимися переменными. Преобразуя (6.3), получим:

$$\frac{du}{f(1,u)-u}=\frac{dx}{x}. \tag{6.4}$$

Интегрируя обе части (6.4), получаем:

$$\int\frac{du}{f(1,u)-u}=\ln|x|+C. \tag{6.5}$$

Так как постоянная C может быть любой, можно записать (6.5) в виде

$$\int\frac{du}{f(1,u)-u}=\ln|x|+\ln|C|=\ln|Cx|.$$

Интегрируя (6.5), получаем u, затем делаем обратную замену $u=\frac{y}{x}$, получаем искомое общее решение однородного дифференциального уравнения. При наличии начальных условий можно найти и частное решение.

Пример 6.5. Найти общее решение дифференциального уравнения:

$xy' - y = y(\ln y - \ln x)$;

$$\frac{dy}{dx}x-y=y\ln\left|\frac{y}{x}\right|.$$

Обе части последнего равенства разделим на x, тогда получим

$$\frac{dy}{dx}-\frac{y}{x}=\frac{y}{x}\ln\left|\frac{y}{x}\right|;$$

$$\frac{dy}{dx}=\frac{y}{x}\left(\ln\left|\frac{y}{x}\right|+1\right),$$

т. е. исходное дифференциальное уравнение является однородным.

Для его решения используем замену $u=\frac{y}{x}\Rightarrow y=ux\Rightarrow y'=u'x+u.$

После замены дифференциальное уравнение примет вид:

$u'x + u = u(\ln u + 1)$;

$$\frac{du}{dx}x = u(\ln u + 1) - u;$$

$$\frac{du}{dx}x = u\ln u;$$

$$\frac{du}{u\ln u} = \frac{dx}{x}.$$

Интегрируем обе части последнего равенства:

$$\int\frac{du}{u\ln u} = \int\frac{dx}{x};$$

$$\int\frac{d\ln u}{\ln u} = \ln|x| + \ln C;$$

$\ln(\ln u) = \ln|Cx|;$

$\ln u = Cx \Rightarrow u = e^{Cx}.$

Делаем обратную замену $u = \frac{y}{x}$ и получаем: $\frac{y}{x} = e^{Cx}$ или $y = xe^{Cx}$ — это и есть общее решение исходного дифференциального уравнения.

Предположим, что заданы начальные условия: $y\big|_{x=2}^{=1}$. Тогда находим частное решение заданного дифференциального уравнения:

$$1 = 2e^{2C} \Rightarrow \frac{1}{2} = e^{2C} \Rightarrow \ln\frac{1}{2} = \ln e^{2C} \Rightarrow \ln\frac{1}{2} = 2C\ln e \Rightarrow C = \frac{\ln\frac{1}{2}}{2},$$

т. е. частное решение имеет вид:

$$y = xe^{\frac{\ln\frac{1}{2}}{2}x}.$$

6.2.3. Линейные дифференциальные уравнения первого порядка

К линейным дифференциальным уравнениям относятся дифференциальные уравнения вида:

$$y' + p(x)y = q(x), \tag{6.6}$$

т. е. линейное относительно неизвестной функции и ее производной.

В (6.6) p(x) и q(x) — известные функции аргумента x [2, 22].

Дифференциальное уравнение (6.6) сводится к двум дифференциальным уравнениям с разделяющимися переменными с помощью следующего приема.

Представим функцию y в виде произведения двух функций y = uv. Одной из этих функций можно распорядиться произвольно, а вторая при этом должна быть определена в зависимости от первой так, чтобы их произведение удовлетворяло исходному дифференциальному уравнению. Свободой выбора одной из функций u и v надо воспользоваться для упрощения дифференциального уравнения, получающегося после замены.

Из равенства y = uv получим y' = u'v + v'u. Это выражение подставим в (6.6) и получим:

$$u'v + v'u + p(x)uv = q(x);$$

$$u'v + u(v' + p(x)v) = q(x).$$

В качестве v выберем какое-нибудь частное решение дифференциального уравнения:

$$v' + p(x)v = 0. \tag{6.7}$$

Тогда для нахождения u мы получим дифференциальное уравнение:

$$u'v = q(x). \tag{6.8}$$

Из дифференциального уравнения (6.7) находим v:

$$\frac{dv}{dx} = -p(x)v \Rightarrow \frac{dv}{v} = -p(x)dx.$$

Интегрируем обе части последнего выражения:

$$\int \frac{dv}{v} = -\int p(x)dx;$$

$$\ln|v| = -\int p(x)dx; \qquad (6.9)$$

$$v = e^{-\int p(x)dx}.$$

Под неопределенным интегралом в (6.9) понимается какая-то одна первообразная от функции p(x), т. е. v есть вполне определенная функция от x.

Теперь, используя найденное значение функции v из (6.8), находим функцию u:

$$\frac{du}{dx} = \frac{q(x)}{v} \Rightarrow \frac{du}{dx} = q(x)e^{\int p(x)dx} \Rightarrow du = q(x)e^{\int p(x)dx}dx.$$

Интегрируем обе части последнего выражения и получаем:

$$u = \int q(x)e^{\int p(x)dx}dx + C. \qquad (6.10)$$

В (6.10) для функции u берутся все первообразные.

Зная функции u и v, находим искомую функцию y:

$$y = uv = e^{-\int p(x)dx}\left(\int q(x)e^{\int p(x)dx}dx + C\right). \qquad (6.11)$$

Выражение (6.11) и является общим решением линейного дифференциального уравнения первого порядка.

Пример 6.6. Найти общее решение линейного дифференциального уравнения $y' - 3\frac{y}{x} = x^3$.

Используем подстановку $y = uv \Rightarrow y' = u'v + v'u$ и получим:

$$u'v + v'u - \frac{3}{x}uv = x^3;$$

$$u'v + u\left(v' - \frac{3}{x}v\right) = x^3.$$

В качестве v выберем какое-то частное решение дифференциального уравнения $v' - \frac{3}{x}v = 0$, тогда u можно найти из дифференциального уравнения $u'v = x^3$.

Находим функцию v:

$$\frac{dv}{dx} - \frac{3}{x}v = 0 \Rightarrow \frac{dv}{dx} =$$

$$= \frac{3}{x}v \Rightarrow \frac{dv}{v} = 3\frac{dx}{x} \Rightarrow \int\frac{dv}{v} = 3\int\frac{dx}{x} \Rightarrow \ln|v| = 3\ln|x| \Rightarrow v = x^3$$

Зная v, находим функцию u:

$$\frac{du}{dx}v = x^3 \Rightarrow \frac{du}{dx}x^3 = x^3;$$

$$\frac{du}{dx} = 1 \Rightarrow du = dx;$$

$$u = x + C.$$

Зная функции u и v, находим исходную функцию y:

$$y = uv = x^3(x + C). \qquad (6.12)$$

Это (6.12) и есть общее решение исходного дифференциального уравнения.

Линейные дифференциальные уравнения первого порядка с постоянными коэффициентами

Общий вид таких дифференциальных уравнений следующий [16]:

$$y' + ay = b, \qquad (6.13)$$

где $a, b \in R$.

Дифференциальное уравнение вида (6.13) решается разделением переменных, т. е.

$$\frac{dy}{dx} = b - ay \Rightarrow \frac{dy}{b - ay} = dx.$$

Интегрируем левую и правую части последнего выражения и получаем:

$$\int\frac{dy}{b - ay} = \int dx \Rightarrow -\frac{1}{a}\int\frac{d(b - ay)}{b - ay} = x + C_1 \Rightarrow -\frac{1}{a}\ln|b - ay| = x + C_1;$$

$$\ln|b - ay| = -ax - aC_1;$$

$b - ay = e^{-ax}e^{-aC_1}$;

$-ay = e^{-ax}e^{-aC_1} - b$;

$$y = -\frac{1}{a}e^{-ax}e^{-aC_1} + \frac{b}{a}.$$

Так как постоянная может быть любая, обозначим $C = -\frac{1}{a}e^{-aC_1}$ и получаем общее решение дифференциального уравнения (6.13):

$$y = Ce^{-ax} + \frac{b}{a}. \qquad (6.14)$$

Пример 6.7. Найдем общее решение дифференциального уравнения: $y' + 2y + 5 = 0$.

$$\frac{dy}{dx} = -2y - 5 \Rightarrow \frac{dy}{(-2y-5)} = dx;$$

$$\int \frac{dy}{(-2y-5)} = \int dx;$$

$$-\frac{1}{2}\int \frac{d(-2y-5)}{(-2y-5)} = x + C;$$

$$-\frac{1}{2}\ln|-2y-5| = x + C;$$

$\ln|-2y - 5| = -2x - 2C$;

$-2y - 5 = e^{-2x}e^{-2C}$;

$-2y = e^{-2x}e^{-2C} + 5$;

$$y = -\frac{1}{2}e^{-2x}e^{-2C} - \frac{5}{2}.$$

Задачи для самостоятельного решения

1. Найти общие решения дифференциальных уравнений:

а) $(x + 5)dy - (y + 10)dx = 0$;

б) $(3xy^2 + 2x)dx + (2y + x^2y)dy = 0$;

в) $y' = \frac{10}{x^2 - 4}$;

г) $2\sin x dx + \frac{3dy}{\sqrt{2y}} = 0$;

д) $2y\cos x dx - \sin^5 x dy = 0$.

2. Предположим, что темп изменения производительности труда характеризуется функцией f(t). Найти функцию производительности труда y = y(t), если

а) $f(t) = \frac{5t}{\sqrt{t^2 + 4}}$; $y\big|_{t=0}^{=0}$;

б) $f(t) = \frac{\ln t}{6t}$; $y\big|_{t=e}^{=1}$.

3. Найти общие решения однородных дифференциальных уравнений:

а) $(y - x)dx + (y + x)dy = 0$;

б) $xdy - ydx = \sqrt{x^2 + y^2}dx$;

в) $(6y + 4x)dx + (3y + 8x)dy = 0$;

г) $x\cos\frac{y}{x}(ydx + xdy) = y\sin\frac{y}{x}(xdy - ydx)$.

4. Найти общие решения линейных дифференциальных уравнений и частные решения там, где заданы начальные условия:

а) $y' + \frac{2x}{1 - x^2}y = 1$;

б) $y' - 7ytgx = \frac{5x}{\cos x}$;

в) $2xy' - 3x^2y = \frac{e^{x^2}}{x + \frac{1}{x}}$, $y\big|_{x=0}^{=1}$;

г) $2xy'+\frac{1}{\ln x}y=\frac{x^2+7}{\ln x}$, $y\big|_{x=e}^{=e^2}$.

5. Найти общие решения дифференциальных уравнений:

а) $y' - 2y + 7 = 0$;

б) $3y' - 6y + 9 = 0$.

Вопросы для самопроверки

1. Какое дифференциальное уравнение называется дифференциальным уравнением первого порядка?

2. Что такое общее решение дифференциального уравнения первого порядка?

3. Что такое частное решение и в чем суть начальных условий для дифференциального уравнения первого порядка?

4. Дать формулировку теоремы существования и единственности решения дифференциального уравнения первого порядка.

5. Что является геометрической иллюстрацией общего и частного решений дифференциального уравнения первого порядка?

6. Что такое дифференциальное уравнения первого порядка с разделяющимися переменными и каким методом его можно решить?

7. Какие дифференциальные уравнения первого порядка называются однородными, каков их метод решения?

8. Какие дифференциальные уравнения первого порядка называются линейными, каков их метод решения?

9. Какие дифференциальные уравнения первого порядка называются обыкновенными? Каков их общий вид?

10. Какие функции называются однородными функциями n-го измерения?

11. Как найти общее решение линейного дифференциального уравнения первого порядка с постоянными коэффициентами?

7. Ряды

7.1. Числовые ряды

Выражение (7.1) $w_1 + w_2 + ... + w_n + ... = \sum_{n=1}^{\infty} w_n$, где w_1, w_2, ..., w_n, ... — члены ряда, являющиеся некоторыми числами, называют *числовым рядом*.

Для любого числового ряда $\sum_{n=1}^{\infty} w_n$ можно построить последовательность его частичных сумм S_n:

$S_1 = w_1$
$S_1 = w_1 + w_2$
$S_3 = w_1 + w_2 + w_3$
... ...
$S_n = w_1 + w_2 + w_3 + ... + w_n$, n = 1, 2, 3 ...

Если существует конечный предел $S = \lim_{n\to\infty} S_n$, то его называют *суммой ряда* (7.1) и говорят, что этот ряд сходится. Если этот предел не существует, то говорят, что ряд (7.1) расходится и суммы не имеет [3, 11, 22].

Приведем конкретные примеры.

Пример 7.1

Гармонический ряд $1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+...+\frac{1}{n}+...$ расходится.

Пример 7.2

Геометрическая прогрессия $w + wq + wq^2 + ... + wq^{n-1} + ...$ ($w \neq 0$) сходится при $|q| < 1$и расходится при $|q| \geq 1$.

Если $|q| < 1$, то $w + wq + wq^2 + ... + wq^{n-1} + ... = \frac{w}{1-q}$.

Пример 7.3

Обобщенный гармонический ряд $\frac{1}{1^a} + \frac{1}{2^a} + \ldots + \frac{1}{n^a} + \ldots$ сходится при $a > 1$ и расходится при $a \le 1$.

Пример 7.4

$$\frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \ldots + \frac{1}{n!} + \ldots = e - 1,$$

т. е. данный ряд сходится и его сумма равна $(e - 1)$.

При исследовании рядов одним из важнейших вопросов является вопрос о том, сходится изучаемый ряд или расходится. Далее мы рассмотрим достаточные признаки, на основании которых можно решить этот вопрос. Сейчас же приведем необходимый признак сходимости рядов, т. е. условие, при невыполнении которого ряды расходятся.

ТЕОРЕМА 7.1. *Если ряд сходится, то его n-й член стремится к нулю при неограниченном возрастании n* [2, 22].

СЛЕДСТВИЕ: если n-й член ряда не стремится к нулю при $n \to \infty$, то ряд расходится. Например, гармонический ряд из примера 7.1 расходится, несмотря на то, что $\lim\limits_{n\to\infty} 1/n = 0_1$.

Основные свойства сходящихся числовых рядов [11]:

1. Сходимость числового ряда не нарушится, если приписать или отбросить конечное число его членов.

2. Если члены сходящегося ряда умножить на одно и то же число k, то его сходимость не нарушится.

3. Два сходящихся ряда $u_1 + u_2 + ... + u_n + ... = S_1$; $v_1 + v_2 + ... + v_n + ... = S_2$
можно почленно складывать (или вычитать), так что ряд $(u_1 \pm v_1) + (u_2 \pm v_2) + ... + (u_n \pm v_n) + ...$ будет сходиться, а его сумма будет равна $S_1 + S_2$.

Признаки сходимости положительных числовых рядов [2, 11, 22]

Признаки сравнения:

1. Если все члены рядов

$$u_1 + u_2 + ... + u_n + ... \qquad (7.2)$$

$$v_1 + v_2 + ... + v_n + ... \qquad (7.3)$$

неотрицательны и $u_n \le v_n$, n = 1, 2, 3, ..., то из сходимости ряда (7.3) следует сходимость ряда (7.2). Из расходимости ряда (7.2) следует расходимость ряда (7.3).

2. Если все члены рядов (7.2) и (7.3) положительны и существует $\lim\limits_{n\to\infty} \frac{u_n}{v_n} = C$, $0 < C < +\infty$, то эти ряды сходятся или расходятся одновременно.

Пример 7.5

Ряд $\frac{1}{2} + \frac{1}{4} + \ldots + \frac{1}{2n} + \ldots$ расходится, так как гармонический ряд $1 + \frac{1}{2} + ... + \frac{1}{n} + ...$ расходится и $\lim\limits_{n\to\infty}\left(\frac{1}{2n} : \frac{1}{n}\right) = \lim\limits_{n\to\infty} \frac{n}{2n} = \frac{1}{2}$.

Пример 7.6

Ряд $1 + \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{3}} + \ldots + \frac{1}{\sqrt{n}} + \ldots$ расходится, так как его члены (начиная со второго) больше соответствующих членов гармонического ряда $1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n} + \ldots$, который расходится.

Признак Коши

Если все члены ряда $w_1 + w_2 + ... + w_n + ...$ неотрицательны и существует $\lim\limits_{n\to\infty} \sqrt[n]{w_n} = b$, то при $b < 1$ этот ряд сходится, а при $b > 1$ расходится (при $b = 1$ данный признак не дает возможности судить о поведении ряда).

Пример 7.7

Исследуем сходимость ряда:

$$\frac{1}{3}+\left(\frac{2}{5}\right)^2+\left(\frac{3}{7}\right)^3+\ldots+\left(\frac{n}{2n+1}\right)^n+\ldots$$

Применяем к данному ряду признак Коши:

$$\lim_{n\to\infty}\sqrt[n]{w_n}=\lim_{n\to\infty}\sqrt[n]{\left(\frac{n}{2n+1}\right)^n}=\lim_{n\to\infty}\frac{n}{2n+1}=\frac{1}{2}$$ и видим, что он сходится.

Признак Даламбера

Если все члены ряда $w_1 + w_2 + ... + w_n + ...$ положительны и существует $\lim\limits_{n\to\infty}\frac{w_{n+1}}{w_n}=l$, то при $l < 1$ этот ряд сходится, а при $l > 1$ этот ряд расходится (при $l = 1$ данный признак не дает возможности судить о поведении ряда).

Пример 7.8

Исследуем сходимость ряда:

$$1+\frac{1}{1\cdot 2}+\frac{1}{1\cdot 2\cdot 3}+\ldots+\frac{1}{1\cdot 2\cdot 3\cdot\ldots\cdot n}+\ldots$$

Применяем к данному ряду признак Даламбера:

$$\lim_{n\to\infty}\frac{w_{n+1}}{w_n}=\lim_{n\to\infty}\left(\frac{1}{1\cdot 2\cdot 3\cdot\ldots\cdot n(n+1)}:\frac{1}{1\cdot 2\cdot 3\cdot\ldots\cdot n}\right)=\lim_{n\to\infty}\frac{n!}{(n+1)!}=$$

$$=\lim_{n\to\infty}\frac{1}{n+1}=0$$

и видим, что он сходится.

Интегральный признак сходимости Коши

Если $w_n = f(n)$, $n = 1, 2, 3, ...$, где $f(n)$ — значение при $x = n$ некоторой функции $f(x)$, непрерывной, положительной и не воз-

растающей при $x \geq 1$, то ряд $w_1 + w_2 + ... + w_n + ...$ сходится или расходится в зависимости от того, существует или нет конечный

$$\lim_{b\to\infty}\int_1^b f(x)dx.$$

Пример 7.9

Исследуем сходимость ряда:

$$\frac{1}{1^2}+\frac{1}{2^2}+\frac{1}{3^2}+\ldots+\frac{1}{n^2}+\ldots$$

Применяем к данному ряду интегральный признак Коши, положив $f(x)=\frac{1}{x^2}$:

$$\lim_{b\to\infty}\int_1^b f(x)dx=\lim_{b\to\infty}\int_1^b\frac{dx}{x^2}=\lim_{b\to\infty}\left(-\frac{1}{x}\right)\Bigg|_1^b=-\lim_{b\to\infty}\left(\frac{1}{b}-1\right)=-\lim_{b\to\infty}\left(\frac{1}{b}-1\right)=1$$

и видим, что он сходится.

Абсолютная и условная сходимость рядов

Числовой ряд $w_1 + w_2 + ... + w_n + ...$ (7.4) называется абсолютно сходящимся, если сходится ряд, составленный из абсолютных величин его членов:

$$|w_1| + |w_2| + ... + |w_n| + ... \qquad (7.5)$$

Абсолютно сходящийся ряд всегда сходится. Если ряд (7.4) сходится, а ряд (7.5) расходится, то говорят, что ряд (7.4) сходится условно [11, 22].

Определение: ряд, положительные и отрицательные члены которого следуют друг за другом поочередно, называется знакочередующимся.

Теперь приведем ТЕОРЕМУ 7.2 Лейбница, которая применяется для знакочередующихся рядов: *ряд* $w_1 - w_2 + w_3 - w_4 + ... + (-1)^{n-1}w_n + ...$, *где все* $w_n > 0$, *сходится, а его сумма положитель-*

на и не превосходит первого члена, если все его члены таковы, что $w_1 > w_2 > w_3 > ... > w_n > ...$ *и* $\lim\limits_{n\to\infty} w_n = 0$ [2, 11, 22].

Свойства абсолютно и условно сходящихся рядов:

1. Если ряд сходится абсолютно, то новый ряд, полученный из него перестановкой членов, также сходится и имеет ту же сумму, что и исходный ряд.

2. Если ряд сходится условно, то какое бы число S ни взять, можно так переставить члены в этом ряду, чтобы сумма преобразованного ряда была равна именно S.

3. Если ряд сходится условно, то можно так переставить члены в этом ряду, что новый ряд будет расходиться.

7.2. Функциональные ряды

Выражение вида:

$$W_1(x) + W_2(x) + ... + W_n(x) ... = \sum_{n=1}^{\infty} W_n(x), \qquad (7.6)$$

где $W_1(x)$, $W_2(x)$, ..., $W_n(x)$, ... — некоторые функции, определенные на одном и том же множестве D, называется функциональным рядом [2, 11].

Множество $E \subseteq D$ всех значений x, при которых функциональный ряд (7.6) сходится (как числовой ряд), называется областью сходимости этого ряда. Функция $S(x)$, $x \in E$ является суммой ряда (7.6), если $S(x) = \lim\limits_{n\to\infty} S_n(x)$, где $S_n(x) = W_1(x) + W_2(x) + ... + W_n(x)$.

Если функция $S(x)$, $x \in P (P \subseteq E)$ является суммой ряда (7.6), то говорят, что этот ряд сходится на множестве P к функции $S(x)$.

Функциональный ряд называется равномерно сходящимся на множестве P к функции $S(x)$, если для $\forall$ числа $\varepsilon > 0$ существует такой номер N, что при $n \geq N$ сразу для всех $x \in P$ выполняется неравенство

$|S(x) - S_n(x)| < \varepsilon$ [11].

Если функциональный ряд сходится на множестве P, то на этом множестве сходимость не обязана быть равномерной, но на

некотором подмножестве множества P сходимость может оказаться равномерной.

Приведем *признак равномерной сходимости Вейерштрасса*: если члены функционального ряда $W_1(x) + W_2(x) + ... + W_n(x) ...$ удовлетворяют на множестве P неравенствам $|W_n(x)| \le W_n$ (n = 1, 2, 3, ...), где W_n — члены сходящегося числового ряда $W_1 + W_2 + ... + W_n + ...$, то функциональный ряд сходится на множестве P равномерно [11].

Пример 7.10

Ряд — $\frac{\sin x}{1^2} + \frac{\sin 2x}{2^2} + \ldots + \frac{\sin nx}{n^2} + \ldots$

сходится на P = (–∞; +∞) равномерно, так как всегда $\left|\frac{\sin nx}{n^2}\right| \le \frac{1}{n^2}$ и ряд $\sum_{n=1}^{\infty} \frac{1}{n^2}$ сходится.

Если функции $W_n(x)$ непрерывны на [a, b], а составленный из них ряд $W_1(x) + W_2(x) + ... + W_n(x) ...$ сходится равномерно на этом отрезке к функции S(x), то:

1) функция S(x) на [a, b] непрерывна;

2) $\int_a^b S(x)dx = \int_a^b W_1(x)dx + \int_a^b W_2(x)dx + \ldots + \int_a^b W_n(x)dx + \ldots$

Пример 7.11

Ряд $1 + x + x^2 + ... + x^{n-1} + ...$ на отрезке $\left[0, \frac{1}{2}\right]$ сходится равномерно к функции $\frac{1}{1-x}$, поэтому

$$\int_0^{1/2} 1 \times dx + \int_0^{1/2} x dx + \ldots + \int_0^{1/2} x^{n-1} dx + \ldots = \int_0^{1/2} \frac{dx}{1-x} \text{ или}$$

$$\frac{1}{2}+\frac{1}{2\times 2^2}+\ldots+\frac{1}{n\times 2^n}+\ldots=\ln 2.$$

Если функции $W_n(x)$ имеют непрерывные производные на отрезке [a, b] и на этом отрезке:

1) ряд $W_1(x) + W_2(x) + ... + W_n(x) ...$ сходится к функции $S(x)$;

2) ряд $W'_1(x) + W'_2(x) + ... + W'_n(x) ...$ сходится равномерно, то $S(x)$ имеет на [a, b] непрерывную производную и

$S'(x) = W'_1(x) + W'_2(x) + ... + W'_n(x) + ...$ [11, 20, 22].

7.3. Степенные ряды

Функциональный ряд

$$\alpha_0 + \alpha_1(x_1 - x_0) + \alpha_2(x - x_0)^2 + ... + \alpha_n(x - x_0)^n +$$

$$+ ... = \sum_{n=0}^{\infty}\alpha_n(x - x_0)^n, \qquad (7.7)$$

где $\alpha_n(n = 0, 1, 2, ...)$ и x_0 — некоторые числа, называют *степенным рядом* с центром в точке x_0.

Возможны следующие три случая:

1) степенной ряд (7.7) сходится только при $x = x_0$ (везде расходящийся ряд);

2) степенной ряд (7.7) сходится (причем абсолютно) при $\forall$ значениях (всюду сходящийся ряд);

3) существует число $R > 0$ такое, что ряд (7.7) сходится абсолютно при $|x - x_0| < R$ и расходится при $|x - x_0| > R$ (R — радиус сходимости ряда). $R = 0$ для всюду расходящегося ряда и $R = \infty$ для всюду сходящегося ряда.

Интервал $(x_0 - R, x_0 + R)$ называют интервалом сходимости степенного ряда (7.7). При этом на концах интервала сходимости степенной ряд может как сходиться, так и расходиться.

Пример 7.12

Найдем интервал сходимости степенного ряда:

$$\frac{x}{1\times 2}+\frac{x^2}{2\times 2^2}+\ldots+\frac{x^n}{n\times 2^n}+\ldots$$

Положим, $W_n = \frac{|x|^n}{n \times 2^n}$; $W_{n+1} = \frac{|x|^{n+1}}{(n+1) \times 2^{n+1}}$.

Тогда по признаку Даламбера имеем:

$$\lim_{n\to\infty} \frac{W_{n+1}}{W_n} = \lim_{n\to\infty} \frac{|x|^{n+1} \times n \times 2^n}{(n+1)2^{n+1}|x|^n} = \frac{|x|}{2} \lim_{n\to\infty} \frac{n}{n+1} = \frac{|x|}{2},$$

следовательно, данный степенной ряд сходится абсолютно при $|x| < 2$ и расходится при $|x| > 2$, а радиус его сходимости равен 2 ($R = 2$).

Исследуем сходимость ряда на концах интервала сходимости: при $x = 2$ ряд $\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n} + \ldots$ расходится, а при $x = -2$ ряд $-\frac{1}{1} + \frac{1}{2} - \frac{1}{3} + \ldots + (-1)^n \frac{1}{n} + \ldots$ сходится условно. Поэтому внутри интервала $[-2, 2)$ ряд сходится абсолютно и неабсолютно сходится в конце $x = -2$ этого интервала.

Основные свойства степенных рядов [2, 11]

1. Если степенной ряд не является всюду расходящимся, то его сумма непрерывна в каждой точке области сходимости.

2. Степенной ряд внутри его области сходимости можно интегрировать почленно, так что если

$\alpha_0 + \alpha_1(x_1 - x_0) + \alpha_2(x - x_0)^2 + \ldots + \alpha_n(x - x_0)^n + \ldots = S(x)$,

$x \in E$,

то

$$\alpha_0(x - x_0) + \alpha_1 \frac{(x - x_0)^2}{2} + \alpha_2 \frac{(x - x_0)^3}{3} + \ldots +$$

$$+ \alpha_n \frac{(x - x_0)^{n+1}}{n+1} + \ldots = \int_0^x S(x)dx.$$

3. Степенной ряд внутри его интервала сходимости можно дифференцировать почленно, так что если

$\alpha_0 + \alpha_1(x_1 - x_0) + \alpha_2(x - x_0)^2 + \ldots + \alpha_n(x - x_0)^n + \ldots = S(x)$,
$x \in (x_0 - R, x_0 + R)$, $R > 0$,
то $\alpha_1 + 2\alpha_1(x - x_0) + 3\alpha_2(x - x_0)^2 + \ldots + n\alpha_n(x - x_0)^{n-1} + \ldots = S'(x)$,
$x \in (x_0 - R, x_0 + R)$.

Это свойство сохраняет силу и для конца интервала сходимости, если только последний ряд на этом конце сходится.

4. Если степенной ряд

$\alpha_0 + \alpha_1(x_1 - x_0) + \alpha_2(x - x_0)^2 + \ldots + \alpha_n(x - x_0)^n + \ldots$

не является всюду расходящимся, то его сумма S(x) имеет внутри интервала сходимости производные всех порядков. При этом

$$\alpha_0 = S(x_0);\ \alpha_1 = S'(x_0),\ \alpha_2 = \frac{S''(x_0)}{2!}, \ldots, \alpha_n = \frac{S^{(n)}(x_0)}{n!}, \ldots$$

Разложение функций в степенные ряды

Если функция f(x) имеет производные всех порядков при $x = x_0$, то степенной ряд

$$f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \ldots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n + \ldots \qquad (7.8)$$

называют рядом Тейлора для функции f(x). При $x_0 = 0$ получают частный случай ряда Тейлора

$$f(0) + \frac{f'(0)}{1!}x + \frac{f''(0)}{2!}x^2 + \ldots + \frac{f^{(n)}(0)}{n!}x^n + \ldots, \qquad (7.9)$$

который часто называют рядом Маклорена [п. 4.3.1, 2, 11, 22].

Для того чтобы ряд (7.8) сходился к функции f(x), необходимо и достаточно, чтобы $\lim\limits_{n\to\infty} R_n(x) = 0$, где $R_n(x)$ — остаточный член ряда Тейлора.

Приведем теорему, которая позволяет устанавливать, стремится ли $R_n(x)$ к нулю при неограниченном возрастании n или нет, т. е. разлагается ли функция f(x) в ряд Тейлора или нет.

ТЕОРЕМА 7.3. *Если функция* $f(x)$ *во всех точках некоторого интервала, содержащего точку* x_0, *имеет* $(n+1)$*-ю производную* $f^{(n+1)}(x)$, *то остаточный член* $R_n(x)$ *для любой точки этого интервала имеет вид*

$$R_n(x) = f^{(n+1)}(\eta)\frac{(x-x_0)^{n+1}}{(n+1)!},$$ где η заключено между x_0 и x [2]

(см. также главу 4).

Приведем разложения в степенной ряд некоторых функций:

$$1+\frac{x}{1!}+\frac{x^2}{2!}+\ldots+\frac{x^n}{n!}+\ldots=e^x,\ x\in(-\infty;+\infty);$$

$$\frac{x}{1!}-\frac{x^3}{3!}+\frac{x^5}{5!}-\ldots+(-1)^{n+1}\frac{x^{2n-1}}{(2n-1)!}+\ldots=\sin x,\ x\in(-\infty;+\infty);$$

$$1-\frac{x^2}{2!}+\frac{x^4}{4!}-\ldots+(-1)^n\frac{x^{2n}}{(2n)!}+\ldots=\cos x,\ x\in(-\infty;+\infty);$$

$$x-\frac{x^3}{3}+\frac{x^5}{5}-\ldots+(-1)^n\frac{x^{2n+1}}{2n+1}+\ldots=\operatorname{arctg}x,\ |x|\le 1;$$

$$x-\frac{x^2}{2}+\frac{x^3}{3}-\frac{x^4}{4}+\ldots+(-1)^{n-1}\frac{x^n}{n}+\ldots=\ln(x+1),\ x\in(-1;1]$$

Разложим в ряд Маклорена функцию sinx. Для этого находим последовательно значения ее производных различных порядков в точке x = 0:

$f(0) = (\sin x)_{x=0} = 0$; $f'(0) = (\cos x)_{x=0} = 1$.

$f''(0) = (-\sin x)_{x=0} = 0$; $f'''(0) = (-\cos x)_{x=0} = -1$.

$f^{(4)}(0) = (\sin x)_{x=0} = 0$; $f^{(5)}(0) = (\cos x)_{x=0} = 1$.

$f^{(6)}(0) = (-\sin x)_{x=0} = 0$; $f^{(7)}(0) = (-\cos x)_{x=0} = -1$ и т. д.

Видно, что значения производных повторяются и образуют периодическую последовательность 0, 1, 0, –1, 0, 1, 0, –1, ... Любая производная функции sinx по абсолютной величине не превосходит 1.

Поэтому ряд Маклорена для формулы sinx сходится к ней на всей числовой оси [2].

Задачи для самостоятельного решения

1. С помощью признаков сравнения исследовать сходимость рядов:

1.1. $\sum_{n=1}^{\infty}\frac{5^n}{5^{2n}+2}$; 1.2. $\sum_{n=1}^{\infty}\frac{1}{n^4+1}$;

1.3. $\sum_{n=1}^{\infty}\cos\frac{1}{n}$; 1.4. $\sum_{n=1}^{\infty}\frac{1}{7^n-1}$.

2. Исследовать сходимость рядов с помощью признака Даламбера:

2.1. $1+\frac{2!}{5}+\frac{3!}{5^2}+\frac{4!}{5^3}+\ldots$

2.2. $\frac{1}{(\ln 2)^3}+\frac{1}{2!(\ln 3)^3}+\frac{1}{3!(\ln 4)^3}+\ldots$

3. Исследовать сходимость рядов с помощью признака Коши:

3.1. $\sum_{n=0}^{\infty}\frac{n+1}{4^n}$; 3.2. $\sum_{n=0}^{\infty}2^n\left(\frac{n-3}{n+5}\right)^{n^2}$.

4. Исследовать сходимость рядов с помощью интегрального признака Коши:

4.1. $\sum_{n=1}^{\infty}\frac{1}{n^2+4n+3}$; 4.2. $\sum_{n=1}^{\infty}\frac{3n^2}{5n^4+2}$.

5. Исследовать абсолютную или условную сходимость рядов:

5.1. $\sum_{n=1}^{\infty}\frac{\cos 2n}{n^3}$; 5.2. $\sum_{n=1}^{\infty}\frac{(-1)^n}{n!}$;

5.3. $\sum_{n=2}^{\infty} \frac{(-1)^{n-1}}{n \ln n}$; 5.4. $\sum_{n=1}^{\infty} \frac{(-1)^n}{(3n-1) \cdot 4^n}$.

6. Найти интервалы сходимости степенных рядов:

6.1. $\sum_{n=1}^{\infty} \frac{(x+5)^{2n}}{3n}$; 6.2. $\sum_{n=1}^{\infty} \frac{(x-2)^{2n-1}}{n^3}$.

6.3. $\sum_{n=1}^{\infty} (-1)^{n-1} \frac{x^n}{n}$; 6.4. $x + 2x^2 + 4x^3 + 8x^4 + \ldots$

Вопросы для самопроверки

1. Что называется числовым рядом?
2. Что такое сумма ряда? Дать определение сходящегося и расходящегося рядов.
3. В чем состоит необходимый признак сходимости ряда?
4. В чем суть признаков Даламбера и Коши?
5. В чем суть интегрального признака Коши?
6. Какой ряд называется знакочередующимся?
7. В чем сущность признака Лейбница?
8. Что называется абсолютной и условной сходимостью ряда?
9. Какой ряд называется функциональным?
10. Что называется областью сходимости функционального ряда?
11. Какой ряд называется степенным?
12. Каковы основные свойства степенных рядов?

Раздел II
ОСНОВЫ ИНФОРМАТИКИ

8. Методологические основы информатики

8.1. Объект и предмет информатики

8.1.1. Основные понятия и определения

Будучи достаточно сложным процессом, автоматизация любой деятельности человека при решении практических задач должна иметь научное — прежде всего методологическое — обеспечение. Как уже было отмечено во введении, наукой, изучающей наиболее общие закономерности внедрения средств автоматизации (компьютеризации) во все сферы жизни общества и последствия этого, является информатика. В рамках данной научной дисциплины автоматизация профессиональной деятельности определяется как процесс создания, внедрения и использования технических, программных средств и математических методов, освобождающих человека от непосредственного участия в получении, преобразовании и передаче энергии, материалов и (или) информации в профессиональной деятельности. Основные виды автоматизируемой профессиональной деятельности: производственные процессы, проектирование, обучение, научные исследования, управление. Основу автоматизации профессиональной деятельности в современных условиях составляют средства электронно-вычислительной техники (ЭВТ) и связи.

Весьма важными и особенно интересными для широкого круга специалистов в области организационного управления представляются особенности *автоматизации управленческой деятельности* как процесса создания, внедрения и использования технических, программных средств и математических методов, предназначенных для автоматизированного сбора, хранения, поиска, переработки и передачи информации, используемой при управлении эргатическими системами, в ходе реализации новых информационных технологий управления. Целью автоматизации управленческой деятельности [2] является повышение эффективности управления (качества управленческих решений, оперативности, повышения производительности управленческого труда и т. д.).

Информатика изучает цели, способы и средства автоматизации деятельности должностных лиц на базе ЭВТ при управлении персоналом, разработке новых систем, совершенствовании видов, форм и способов управления, обучении персонала.

Как и всякая другая научная дисциплина, информатика имеет свой объект и предмет.

В качестве *объекта информатики* выступает автоматизированная информационная система (АИС), представляющая собой совокупность технических, программных средств и организационных мероприятий, предназначенных для автоматизации информационных процессов в профессиональной деятельности. Основным техническим средством АИС является ЭВМ.

Используя термин *информация*, мы, как правило, не задумываемся о том, что она собой представляет. Надо отметить, что вопрос этот является достаточно сложным (он будет подробнее рассмотрен в п. 9.1). До настоящего времени в науке не выработано строгого определения понятия информации. Говоря об информационных процессах в АИС, мы пока будем понимать под информацией некоторую совокупность данных (текстовых, числовых, графических) и связей между ними.

Под переработкой информации понимаются все возможные информационные процессы, сопровождающие профессиональную деятельность: сбор информации, хранение информации, поиск информации, представление информации на определенном носи-

теле в определенном виде (визуальном, графическом, текстовом, звуковом), получение новой информации (например, в результате проведения расчетов), передача информации по каналам связи различным адресатам и др.

АИС должна рассматриваться как инструмент в руках должностных лиц, реализующих переработку информации в процессе профессиональной деятельности. Можно сказать, что наличие этого инструмента фактически определяет новую технологию осуществления профессиональной деятельности.

Понятие *технология* означает комплекс знаний о способах, приемах труда, наборах материально-технических факторов, способах их соединения для создания какого-либо продукта или оказания услуги. Применительно к промышленному производству используется понятие *производственная индустриальная технология*.

Применение понятия *технология* к информационным процессам привело к возникновению понятия *информационная технология* — совокупность знаний о способах автоматизированной переработки информации с использованием ЭВМ для автоматизации управленческой деятельности. Важно отметить, что результатом реализации любой информационной технологии является создание информационного продукта (например, программы на ЭВМ) или оказание информационной услуги (например, формирование справки в ответ на запрос пользователя банка данных).

Создание новых информационных технологий и внедрение их в профессиональную деятельность является одной из основных задач информатики. Именно поэтому в качестве *предмета информатики* целесообразно рассматривать *информационные технологии*, определяющие рациональные способы разработки и применения АИС.

Каждая АИС обеспечивает реализацию некоторой информационной технологии переработки информации в процессе профессиональной деятельности. Таким образом, в качестве задач информатики можно рассматривать создание новых информационных технологий и реализующих их АИС или перенесение известных информационных технологий из одной области человеческой деятельности в другую.

8.1.2. Классификация автоматизированных информационных систем

В качестве основного классификационного признака АИС целесообразно рассматривать особенности автоматизируемой профессиональной деятельности — процесса переработки входной информации для получения требуемой выходной информации, в котором АИС выступает в качестве инструмента должностного лица или группы должностных лиц, участвующих в управлении организационной системой [54].

В соответствии с предложенным классификационным признаком можно выделить следующие классы АИС (рис. 8.1.1):

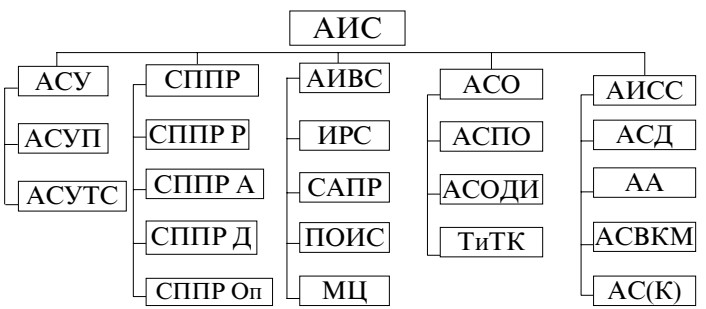


Рис. 8.1.1. Классификация АИС

- автоматизированные системы управления (АСУ);
- системы поддержки принятия решений (СППР);
- автоматизированные информационно-вычислительные системы (АИВС);
- автоматизированные системы обучения (АСО);
- автоматизированные информационно-справочные системы (АИСС).

Рассмотрим особенности каждого класса АИС и характеристики возможных видов АИС в их составе каждого класса.

Автоматизированные системы управления

Автоматизированная система управления предназначена для автоматизации всех или большинства задач управления, решаемых коллективным органом управления (министерством, дирекцией, правлением, службой, группой управления и т. д.). В зависимости от объекта управления различают АСУ персоналом и техническими средствами (АСУП и АСУТС). АСУ является организационной и технической основой реализации рациональной технологии коллективного решения задач управления в различных условиях. В этой связи разработка рациональной технологии организационного управления является определяющим этапом создания любой АСУ.

АСУП обеспечивает автоматизированную переработку информации, необходимой для управления организацией в повседневной деятельности, а также при подготовке и реализации программ развития.

АСУТС предназначены для реализации соответствующих технологических процессов. Они являются по сути передаточным звеном между должностными лицами, осуществляющими управление техническими системами, и самими техническими системами.

В настоящее время АСУТС нашли широкое распространение во всех развитых государствах. Объясняется это тем, что управление существующими новейшими технологическими процессами без применения АСУТС становится практически невозможным. Что касается АСУП, то в настоящее время такие системы широко используются в странах Запада и непрерывно ведутся работы по созданию новых систем, в том числе на базе достижений в области искусственного интеллекта.

Системы поддержки принятия решений

Системы поддержки принятия решений (СППР) являются достаточно новым классом АИС, теория создания которых в настоящее время интенсивно развивается.

СППР предназначена для автоматизации деятельности конкретных должностных лиц при выполнении ими своих должност-

ных (функциональных) обязанностей в процессе управления персоналом и (или) техническими средствами.

Выделяются четыре категории должностных лиц, деятельность которых отличается различной спецификой переработки информации: руководитель, должностное лицо аппарата управления, оперативный дежурный, оператор. В соответствии с четырьмя категориями должностных лиц различают и четыре вида СППР: СППР руководителя (СППР Р), СППР должностного лица аппарата управления (СППР А), СППР оперативного дежурного (СППР Д) и СППР оператора (СППР Оп).

Рассмотрим специфику деятельности должностных лиц, относящихся к каждой выделенной категории.

К категории "руководитель" относятся должностные лица, на которых возложено управление подчиненными должностными лицами (подразделениями организации) и принятие решений в процессе руководства. Основная форма деятельности руководителя — деловое общение.

Деятельность должностных лиц, относящихся к категории "руководитель", характеризуется следующими *особенностями*:

- при централизации принятия решений резко возрастают объемы информации, уменьшается время на обдумывание и анализ, растут сложности комплексного учета всех факторов;
- велика доля текущих задач, не позволяющих сосредоточиться на стратегических целях;
- в процессе деятельности преобладают приемы, обусловленные привычками, опытом, традициями и другими неформализуемыми обстоятельствами;
- при принятии решения руководитель не всегда в состоянии описать и даже представить достаточно полную умозрительную модель ситуации, а вынужден использовать лишь некоторое представление о ней;
- деятельность руководителя в значительной мере зависит от темперамента и стиля деятельности, от степени знаний причин и следствий, ясности представления взаимосвязей, объема имеющейся информации.

Перечисленные особенности деятельности должностных лиц категории "руководитель" обусловливают крайнюю сложность автоматизации их деятельности, которая содержит большое количество неформальных элементов, прежде всего таких, как оперативное и стратегическое управление, а также принятие решений. Исходя из особенностей деятельности руководителя, можно сформулировать следующие основные *требования*, предъявляемые к СППР Р:

1) наличие широкой информационной базы с возможностью оперативного поиска требуемой информации;

2) наглядность представления информации в форме, адаптированной к запросам конкретного должностного лица (текста, таблиц, графиков, диаграмм и т. д.);

3) обеспечение оперативной связи с другими источниками информации в системе управления, и особенно с непосредственными помощниками;

4) наличие диалоговых программных средств обеспечения принятия решений на основе формальных (математических) методов;

5) простота работы при повышенной надежности технических и программных средств;

6) обеспечение возможности накопления в памяти ЭВМ опыта и знаний (в рамках интеллектуальных СППР).

Необходимо отметить, что требования 2, 3 и 5 являются универсальными и относятся ко всем видам СППР.

В настоящее время требования 1, 2, 3 и 5 могут быть полностью удовлетворены с использованием известных информационных технологий. Что касается требований 4 и 6 (наличия программных средств обеспечения решений и накопления в памяти ЭВМ опыта и знаний), то их удовлетворение составляет основную теоретическую проблему, возникающую при создании СППР Р.

К категории "должностное лицо аппарата управления" относятся специалисты, занимающиеся аналитической работой по подготовке решений руководителя и их документальным оформлением. Основу деятельности этой категории лиц составляет оценка

различных вариантов решения (проведение оценочных расчетов) и разработка проектов различных документов.

Эффективность функционирования аппарата управления во многом определяется продуктивностью деятельности специалистов, особенно в вопросах создания новой информации. Доля творческого труда в их работе достаточно высока. Именно эти специалисты обеспечивают практически всю информационную подготовку для принятия решения руководителем. Они являются основными исполнителями документов, определяя их качество.

СППР А должна прежде всего создать должностным лицам условия для плодотворного ведения аналитической работы и сведения к минимуму доли рутинных работ (поиск информации, оформление документов, проведение оперативных расчетов и т. д.).

Особенности деятельности должностных лиц аппарата управления определяют следующие основные *требования* к СППР А:

1) обеспечение оперативного поиска и отображения всей информации, необходимой для подготовки решений и формирования проектов документов в пределах его компетентности;

2) обеспечение возможности ведения оперативных расчетов и моделирования для оценки ситуации и подготовки вариантов решений;

3) обеспечение возможности автоматизированной подготовки проектов документов (текстов, графиков, диаграмм и т. п.).

К основным элементам СППР А следует отнести средства ведения оперативных расчетов и моделирования, поскольку именно эти средства в наибольшей степени обеспечивают повышение эффективности и качества управления.

К категории “оперативный дежурный” относятся должностные лица, выполняющие обязанности по оперативному руководству организационной системой во время дежурства на соответствующих пунктах управления в течение определенного времени.

Основными *особенностями* деятельности оперативных дежурных являются:

- относительно узкий круг решаемых задач;
- жесткая регламентация деятельности в большинстве вариантов складывающейся обстановки;

• жесткий лимит времени на принятие решений и выполнение различных операций.

Перечисленные особенности деятельности оперативных дежурных определяют в качестве основных *требований* к СППР Д обеспечение оперативного предоставления информации, необходимой оперативному дежурному в заранее определенных ситуациях, а также обеспечение оперативного анализа складывающейся ситуации. Последнее требование может быть обеспечено с использованием технологии экспертных систем.

К категории "оператор" могут быть отнесены должностные лица, выполняющие техническую работу по заранее определенному алгоритму. Основная особенность деятельности оператора — отсутствие необходимости принимать сложные решения в процессе своей деятельности. СППР Оп должна обеспечивать возможность работы должностного лица со справочной информацией и автоматизированной подготовки текстов документов.

Автоматизированные информационно-вычислительные системы

АИВС предназначены для решения сложных в математическом отношении задач, требующих больших объемов самой разнообразной информации. Таким образом, видом деятельности, автоматизируемым АИВС, является проведение различных (сложных и "объемных") расчетов. Эти системы используются для обеспечения научных исследований и разработок, а также как подсистемы АСУ и СППР в тех случаях, когда выработка управленческих решений должна опираться на сложные вычисления.

В зависимости от специфики области деятельности, в которой используются АИВС, различают следующие виды этих систем.

Информационно-расчетные системы (ИРС)

ИРС — это автоматизированная информационная система, предназначенная для обеспечения оперативных расчетов и автоматизации обмена информацией между рабочими местами в пределах некоторой организации или системы организаций. ИРС обычно сопрягается с автоматизированной системой управления и в рамках последней может рассматриваться как ее подсистема.

Технической базой ИРС являются, как правило, сети больших, малых и микро-ЭВМ. ИРС имеют сетевую структуру и могут охватывать несколько десятков и даже сотен рабочих мест различных уровней иерархии. Основной сложностью при создании ИРС является обеспечение высокой оперативности расчетов и обмена информации в системе при строгом разграничении доступа должностных лиц к служебной информации.

Системы автоматизации проектирования (САПР)

САПР — это автоматизированная информационная система, предназначенная для автоматизации деятельности подразделений проектной организации или коллектива специалистов в процессе разработки проектов изделий на основе применения единой информационной базы, математических и графических моделей, автоматизированных проектных и конструкторских процедур. САПР является одной из систем интегральной автоматизации производства, обеспечивающих реализацию автоматизированного цикла создания нового изделия от предпроектных научных исследований до выпуска серийного образца.

В области экономики САПР могут использоваться при проектировании экономических информационных систем и их элементов. Кроме того, технология САПР может обеспечить создание автоматизированной системы отображения обстановки на экране в процессе ведения экономических операций или в ходе деловых игр различных типов.

Проблемно-ориентированные имитационные системы (ПОИС)

ПОИС предназначены для автоматизации разработки имитационных моделей в некоторой предметной области [39]. Например, если в качестве предметной области взять развитие автомобилестроения, то любая модель, создаваемая в этой предметной области, может включать стандартные блоки, моделирующие деятельность предприятий, поставляющих комплектующие; собственно сборочные производства; сбыт, обслуживание и ремонт автомобилей; рекламу и др. Эти стандартные блоки могут строиться с различной детализацией моделируемых процессов и различной оперативностью расчетов. Пользователь, работая с ПОИС,

сообщает ей, какая модель ему нужна (т. е. что необходимо учесть при моделировании и с какой степенью точности), а ПОИС автоматически формирует имитационную модель, необходимую пользователю.

В состав программного обеспечения ПОИС входят банки типовых моделей (БТМ) предметных областей, планировщик моделей, базы данных предметных областей, а также средства диалогового общения пользователя с ПОИС.

ПОИС является достаточно сложной АИС, реализуемой, как правило, с использованием технологии искусственного интеллекта на высокопроизводительных ЭВМ.

Моделирующие центры (МЦ)

МЦ — автоматизированная информационная система, представляющая собой комплекс готовых к использованию моделей, объединенных единой предметной областью, информационной базой и языком общения с пользователями [39].

МЦ, так же как и ПОИС, предназначены для обеспечения проведения исследований на различных моделях. Но в отличие от ПОИС МЦ не обеспечивают автоматизацию создания имитационных моделей, а предоставляют пользователю возможность комфортной работы с готовыми моделями.

МЦ могут являться системами как коллективного, так и индивидуального использования и в принципе не требуют для своей реализации мощных ЭВМ.

Автоматизированные системы обучения (АСО)

Традиционные методы обучения специалистов в различных областях профессиональной деятельности складывались многими десятилетиями, в течение которых накоплен большой опыт.

Однако, как свидетельствуют многочисленные исследования, традиционные методы обучения обладают рядом недостатков: пассивный характер устного изложения, трудность организации активной работы студентов, невозможность учета в полной мере индивидуальных особенностей отдельных обучаемых и т. д.

Одним из возможных путей преодоления этих трудностей является создание АСО — автоматизированных информационных систем, предназначенных для автоматизации подготовки специалистов с участием или без участия преподавателя и обеспечивающих обучение, подготовку учебных курсов, управление процессом обучения и оценку его результатов [39]. Основными видами АСО являются автоматизированные системы программированного обучения (АСПО), системы обеспечения деловых игр (АСОДИ), тренажеры и тренажерные комплексы (ТиТК).

АСПО ориентированы на обучение в основном по теоретическим разделам курсов и дисциплин. В рамках АСПО реализуются заранее подготовленные квалифицированными преподавателями "компьютерные курсы". При этом учебный материал разделяется на порции (дозы) и для каждой порции материала указывается возможная реакция обучаемого. В зависимости от действий обучаемого и его ответов на поставленные вопросы АСПО формирует очередную дозу представляемой информации.

Наибольшую сложность при создании АСПО составляет разработка "компьютерного курса" для конкретной дисциплины. Именно поэтому в настоящее время наибольшее распространение получили "компьютерные курсы" по традиционным, отработанным в методическом плане дисциплинам (физике, элементарной математике, химии и т. д.).

АСОДИ предназначена для подготовки и проведения деловых игр, сущность которых заключается в имитации принятия должностными лицами индивидуальных и групповых решений в различных проблемных ситуациях путем игры по заданным правилам.

В ходе деловой игры на АСОДИ возлагаются следующие задачи:

- хранение и предоставление обучаемым и руководителям игры текущей информации о проблемной среде в процессе деловой игры в соответствии с их компетенцией;
- формирование по заданным правилам реакции проблемной среды на действия обучаемых;

• обмен информацией между участниками игры (обучаемыми и руководителями игры);

• контроль и обобщение действий обучаемых в процессе деловой игры;

• предоставление руководителям игры возможности вмешательства в ход игры, например, для смены обстановки.

Технической базой АСОДИ являются высокопроизводительные ЭВМ или локальные вычислительные сети. Методологической базой АСОДИ, как правило, является имитационное моделирование на ЭВМ.

ТиТК предназначены для обучения практическим навыкам работы на конкретных рабочих местах (боевых постах). Они являются средствами индивидуального (тренажеры) и группового (тренажерные комплексы) обучения.

ТиТК являются достаточно дорогостоящими средствами обучения, а их создание требует больших затрат времени. Однако их чрезвычайно высокая эффективность при обучении таких специалистов, как летчики, космонавты, водители, операторы систем управления и т. д., позволяет считать их достаточно перспективными видами АСО.

Автоматизированные информационно-справочные системы (АИСС)

АИСС — это автоматизированная информационная система, предназначенная для сбора, хранения, поиска и выдачи в требуемом виде потребителям информации справочного характера.

В зависимости от характера работы с информацией различают следующие виды АИСС:

• автоматизированные архивы (АА);

• автоматизированные системы делопроизводства (АСД);

• автоматизированные справочники (АС) и картотеки (АК);

• автоматизированные системы ведения электронных карт местности (АСВЭКМ) и др.

В настоящее время разработано большое количество разновидностей АИСС и оно продолжает увеличиваться. АИСС создаются с использованием технологии баз данных, достаточно хоро-

шо разработанной и получившей широкое распространение. Для создания АИСС, как правило, не требуется высокопроизводительная вычислительная техника.

Простота создания АИСС и высокий положительный эффект от их применения определили их активное использование во всех сферах профессиональной (в том числе и управленческой) деятельности.

8.1.3. Место информационных и расчетных задач в составе программного обеспечения ЭВМ

Согласно определению, данному в п. 8.1.1, АИС представляет собой совокупность трех взаимосвязанных компонентов: технических средств, программных средств и организационных мероприятий. Под техническими средствами понимаются ЭВМ, устройства ввода и вывода информации (дисплеи, печатающие устройства, графопостроители, сканеры, плоттеры, мониторы и т. д.), устройства долговременного хранения информации (накопители информации различных типов), сетевое оборудование и каналы связи. Технические средства АИС сами по себе не в состоянии решить какую-либо задачу. Для того чтобы АИС начала функционировать, в ЭВМ необходимо ввести программу, описывающую алгоритм работы технических средств по переработке информации в интересах решения конкретной практической задачи.

Совокупность математических методов, алгоритмических языков и алгоритмов, характеризующих логические и математические возможности ЭВМ, называется математическим обеспечением ЭВМ. Алгоритмы, входящие в математическое обеспечение, реализуются в ЭВМ или аппаратно, или программно. Аппаратная реализация алгоритмов предполагает наличие в составе ЭВМ технических устройств, преобразующих входные сигналы в выходные по жесткому, неизменяемому алгоритму.

Комплекс программ, описаний и инструкций, обеспечивающих создание и отладку программ и решение задач на ЭВМ, называется программным обеспечением ЭВМ. По существу, программное обеспечение — это записанное на входном языке ЭВМ мате-

матическое обеспечение ЭВМ. Одно и то же математическое обеспечение может быть реализовано для различных типов ЭВМ различным программным обеспечением.

Поскольку математические методы и алгоритмы неразрывно связаны с программами, их реализующими, на практике вместо терминов "математическое обеспечение" и "программное обеспечение" часто используется термин "математическое и программное обеспечение" (МПО).

При анализе состава МПО ЭВМ будем вести речь о программном обеспечении (ПО), так как аналогичный состав имеет и соответствующее математическое обеспечение.

Вариант типовой структуры программного обеспечения ЭВМ представлен на рис. 8.1.2 [53, 54].

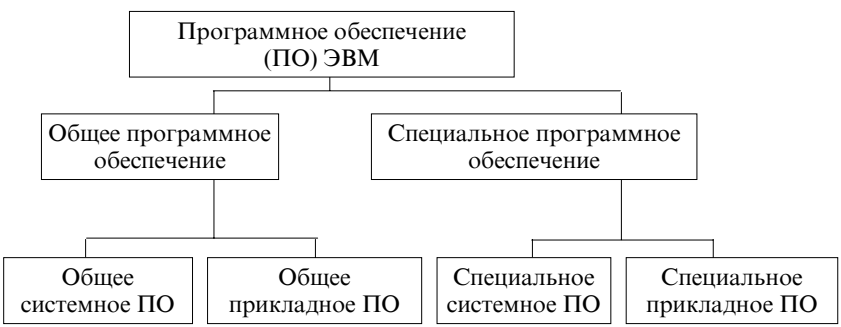


Рис. 8.1.2. Структура программного обеспечения ЭВМ (АИС)

Программное обеспечение ЭВМ состоит из двух частей: общего программного обеспечения и специального программного обеспечения.

Общее программное обеспечение (ОПО) представляет собой комплекс программ, предназначенных для обеспечения работы ЭВМ в различных режимах и снижения трудоемкости создания и отладки программ пользователей.

Основные функции ОПО сводятся к следующим:

• автоматическое управление вычислительным процессом в различных режимах работы ЭВМ при минимальном вмешатель-

стве оператора, программиста, конечного пользователя в этот процесс;

• обеспечение возможности подготовки программ к решению на ЭВМ с помощью средств автоматизации программирования;

• рациональное распределение ресурсов ЭВМ при одновременном решении нескольких задач, что значительно повышает эффективность использования ЭВМ;

• разграничение доступа различных пользователей к данным, хранимым и обрабатываемым в ЭВМ, и обеспечение защиты данных;

• контроль, диагностика и локализация неисправностей ЭВМ и т. д.

По назначению и функциональным особенностям ОПО делится на две взаимосвязанные части: общее системное программное обеспечение (ОСПО) и общее прикладное программное обеспечение (ОППО).

В состав ОСПО входят операционная система (ОС), системы программирования (СП) и программы контроля и диагностики состояния ЭВМ.

Операционной системой называется комплекс программ, осуществляющих управление вычислительным процессом, обеспечивающих связь пользователя с ЭВМ на этапах запуска задач и реализующих наиболее общие алгоритмы обработки информации на данной ЭВМ. Главная функция ОС — обеспечение эффективной работы ЭВМ и всех внешних устройств (мониторов, устройств ввода-вывода и т. д.) в различных режимах работы.

Под режимом работы понимается способ организации выполнения ЭВМ задания или нескольких заданий одновременно. Основными режимами работы являются: монопольный, многопрограммный (мультипрограммный) и режим разделения времени.

В *монопольном* режиме все устройства ЭВМ заняты выполнением только одного задания, являющегося основной единицей работы ЭВМ. Задание может включать несколько пунктов, выполняемых ОС последовательно. Например, задание может включать:

1) трансляцию программы;

2) компоновку оттранслированной программы;

3) запуск программы.

При монопольном режиме все ресурсы ЭВМ используются по мере надобности для отработки очередного пункта задания. С точки зрения загрузки ЭВМ этот режим наименее эффективен, так как в процессе обработки одного задания различные устройства ЭВМ работают с неодинаковой нагрузкой или вообще простаивают значительную часть времени. Однако этот режим наиболее удобен для пользователя, так как время решения задачи при этом минимально. В настоящее время монопольный режим наиболее широко используется в микро-ЭВМ (прежде всего персональных ЭВМ).

Для увеличения производительности и эффективности использования ЭВМ за счет организации параллельной работы основных устройств ЭВМ применяется *мультипрограммный* режим работы.

В этом режиме ОС принимает к исполнению сразу несколько заданий. При достаточно большом количестве одновременно находящихся в памяти ЭВМ заданий данный режим обеспечивает практически полную загрузку всех устройств ЭВМ.

Мультипрограммный режим является основным в работе ЭВМ серий ЕС и СМ (такие ЭВМ по-прежнему используются весьма широко). Кроме того, он находит частичное применение и в ПЭВМ высокой производительности, что позволяет пользователю одновременно ввести в ЭВМ несколько заданий. Применение мультипрограммного режима на "больших" ЭВМ позволило обеспечить пакетную обработку задач, при которой пользователи передавали задание оператору, оператор формировал пакеты этих заданий, пропускал их через ЭВМ и затем возвращал пользователям результаты решения сразу по всем заданиям, составляющим очередной пакет.

Пакетная обработка заданий позволяет существенно повысить эффективность работы ЭВМ, но крайне неудобна для пользователя, поскольку он связан с ЭВМ не непосредственно, а через оператора. Наличие этого промежуточного звена приводит к существенному увеличению суммарного времени решения задачи на ЭВМ.

Для приближения пользователя к ЭВМ и устранения оператора как промежуточного звена между пользователем и ЭВМ были созданы ОС, реализующие особый вид мультипрограммного режима — *режима разделения времени*. Основным средством связи пользователей с ЭВМ стали дисплеи. Реализация режима разделения времени приводит к тому, что пользователи получают связь с ЭВМ поочередно на небольшой промежуток времени. Если этот промежуток времени невелик и невелико количество одновременно работающих пользователей, то каждый работающий пользователь не будет ощущать перерывов связи с ЭВМ. Таким образом, создается впечатление, что пользователь работает один на некоторой воображаемой ЭВМ.

Недостатком режима разделения времени является уменьшение скорости вычислений пропорционально числу одновременно работающих пользователей. Однако, несмотря на этот недостаток, режим разделения времени является основным режимом работы всех современных ЭВМ, обслуживающих несколько пользователей.

Основными элементами ОС являются процессор языка управления, супервизор и файловая система.

Процессор языка управления представляет собой программу, предназначенную для распознавания и преобразования команд пользователя и оператора ЭВМ в машинное представление с целью их последующей обработки.

Основными функциями *супервизора* являются следующие: контроль загруженности различных устройств ЭВМ заданиями, распределение оперативной памяти между заданиями, защита одновременно решаемых заданий (задач) друг относительно друга, запуск операций ввода-вывода и т. д. По своему месту в программном обеспечении ЭВМ супервизор занимает положение посредника между аппаратным обеспечением ЭВМ и всем другим программным обеспечением машины.

Файловая система образуется программами, которые поддерживают ведение всей совокупности файлов (наборов данных) в ЭВМ. Основными функциями этой системы являются: поиск тре-

буемых файлов, модификация информации в файлах, перемещение файлов, копирование файлов, удаление файлов.

Системы программирования предназначены для обеспечения создания и отладки программ пользователей, написанных на каком-либо языке программирования (ПАСКАЛЬ, С, С++ , ФОРТРАН и т. д.). В настоящее время для этих целей широко используются так называемые среды программирования (разработки программ) — например, продукты фирмы Borland DELPHI или Builder С++ , позволяющие быстро создавать качественные приложения.

Программы *контроля и диагностики* состояния ЭВМ предназначены для осуществления непрерывного контроля работы основных устройств ЭВМ, а также поиска неисправных блоков и узлов ЭВМ в случае обнаружения отказов или устойчивых сбоев.

Общее прикладное программное обеспечения (ОППО) включает: пакеты прикладных программ, системы управления базами данных, интеграторы и другие (подобные) прикладные программные системы. Особенностью объектов ОППО является то, что эти средства не требуют от пользователей при решении ими конкретных практических задач на ЭВМ проведения операций, связанных с программированием.

Под *пакетами прикладных программ* (ППП) понимается совокупность готовых к решению программ, объединяемых в пакет по единому содержательному признаку с помощью дополнительной управляющей программы. Данная программа автоматизирует и упрощает стандартную схему использования готовых программ на ЭВМ. Основными функциями управляющей программы являются следующие: поддержание диалоговой (дружественной) формы получения информации от пользователя (обычно это режим меню), вызов соответствующих программ из пакета с целью решения ими содержательных задач, поставленных пользователем, выдача пользователю выходных данных по решенным задачам в удобной форме.

В настоящее время ППП наряду с системами управления базами данных являются самой распространенной формой прикладного программного продукта для массового пользователя. Среди

ППП выделяются пакеты трех типов: проблемно-ориентированные, интегрированные и инструментальные.

Проблемно-ориентированные пакеты структурно являются наиболее простыми. Они состоят из программ, которые нацелены на решение фиксированного числа задач из относительно узкой предметной области. При этом каждой частной задаче соответствует вполне определенная программа ее решения. В функции управляющей программы входит распознавание в запросе пользователя имени и атрибутов той задачи, которую он выбирает для решения, и запуск соответствующей программы на исполнение.

Интегрированные пакеты программ являются расширением ППП первого типа путем их наращивания такими программами, которые автоматизируют все (или большинство) сопутствующих операций, выполняемых лицом, пользующимся пакетом. К числу указанных программ наиболее часто относятся текстовый редактор, система управления базами данных, графический редактор, реже электронная таблица и др. В отличие от самостоятельных версий этих программ данные версии названных программ носят упрощенный характер, достаточный лишь для решения задач из соответствующей предметной области.

Инструментальные пакеты программ отличаются от рассмотренных выше двух типов ППП отсутствием в них программ, строго ориентированных на решение конкретных практических задач. Данные пакеты состоят из программ, каждая из которых может рассматриваться как необходимый элемент решения задач из некоторой предметной области. Таким образом, при использовании данных ППП пользователь в своем запросе указывает структуру элементов, которая реализует прикладную задачу. Управляющая программа пакета на основе заданной структуры элементов создает соответствующую рабочую программу решения требуемой прикладной задачи и затем передает ей управление. В последнее время появились так называемые интеллектуальные ППП, которые будут рассмотрены в п. 10 настоящего учебника.

Объекты ОПО, как правило, поставляются промышленностью совместно с ЭВМ. Это носит универсальный характер в том смысле, что позволяет создать любую программу, совместимую с

аппаратными и вычислительными возможностями конкретной ЭВМ, и провести на ней расчеты. ОПО по существу является инструментом создания специального программного обеспечения (СПО) — комплекса программ, предназначенных для решения конкретных управленческих, исследовательских или производственных задач. Конкретное содержание СПО полностью определяет вид конкретной АИС и, конечно, зависит от ее типа.

СПО, так же как и ОПО, как правило, состоит из двух частей: специального системного программного обеспечения (ССПО) и специального прикладного программного обеспечения (СППО). ССПО выполняет в АИС функции, аналогичные функциям ОС в ОПО.

Необходимость ССПО в вычислительных комплексах экономического назначения обусловливается двумя причинами: обеспечением требования поддержки особых (специальных) режимов проведения вычислительных работ в этих комплексах (связанных прежде всего с дополнительной защитой обрабатываемой информации); необходимостью управления функционированием специальных (нетрадиционных) внешних устройств (например, демонстрационных табло).

СППО представляет собой комплекс программ, каждая из которых реализует тот или иной алгоритм переработки информации. Данные программы принято называть задачами и, хотя это название нельзя признать удачным, оно в настоящее время является общепринятым. Задачи являются основными элементами АИС, в том числе и экономического назначения, поскольку они определяют ее возможности как средства автоматизации деятельности должностных лиц при управлении персоналом.

В дальнейшем в учебнике будут более подробно рассмотрены вопросы создания и использования СППО как основного элемента АИС.

8.1.4. Классификация информационных и расчетных задач

Все задачи, входящие в СППО, можно классифицировать по нескольким признакам:

• характеру переработки информации;
• назначению;
• уровню применения.

Необходимость приведенной ниже классификации определяется различием требований, предъявляемых к задачам каждого класса.

Основным классификационным *признаком*, по которому все задачи, входящие в СППО, делятся на два различных класса, является *характер переработки информации*. В зависимости от характера переработки информации задачи делятся на *информационные* и *расчетные*.

Информационной задачей (ИЗ) называется элемент специального прикладного программного обеспечения ЭВМ (программа на ЭВМ), *алгоритм переработки информации которого не приводит к созданию новой информации, не содержащейся в исходной*. Примером информационных задач могут служить задачи: поиска информации, хранящейся в памяти ЭВМ, оформления (печати) бухгалтерских и управленческих документов, нанесения информации на карту и т. д. Таким образом, информационные задачи осуществляют процессы сбора, хранения, поиска информации и преобразования ее из одного вида в другой без изменения существа этой информации и без создания новой информации.

Информационные задачи являются в настоящее время одними из самых простых, имеющих хорошо развитые средства создания, и достаточно эффективными элементами СППО при автоматизации деятельности должностных лиц. Они позволяют полностью исключить или значительно упростить прежде всего рутинные процедуры в деятельности должностных лиц (хранение, поиск, сортировка информации, составление документов и их тиражирование и т. д.) и тем самым сократить необходимое количество персонала, занятого в основном технической деятельностью (машинистки, делопроизводители, работники библиотек, архивов и т. д.).

Расчетной задачей (РЗ) называется элемент специального прикладного программного обеспечения ЭВМ (программа на ЭВМ), *алгоритм переработки информации которого приводит к созданию*

новой информации, непосредственно не содержащейся в исходной. К расчетным задачам относятся следующие: анализ итогов хозяйственной деятельности, расчет показателей эффективности экономической операции, расчет заработной платы сотрудников и т. д.

В свою очередь, *расчетные задачи* подразделяются *на вычислительные задачи* и *математические модели*.

Вычислительной задачей (ВЗ) называется расчетная задача, алгоритм переработки информации которой построен без использования методов математического моделирования. Обычно алгоритмы вычислительных задач *известны до начала их разработки* и, как правило, *нормативно закреплены* в директивах, наставлениях, справочниках, ГОСТ и т. п. Примерами вычислительных задач являются задачи: расчет подоходного налога, расчет показателей финансовой отчетности, расчет нормативного расхода средств, подведение итогов работы фирмы и т. д.

Математической моделью (ММ) называется расчетная задача, алгоритм переработки информации которой основан на использовании тех или иных методов математического моделирования. Классификацию элементов СППО по назначению и уровню применения приведем для тех задач, которые используются в целях автоматизации управленческой деятельности (остальные будут рассматриваться ниже).

По назначению информационные и расчетные задачи делятся на *штатные* и *исследовательские*.

Штатной называют информационную или расчетную задачу, официально включенную в типовой цикл управления организацией и *используемую должностными лицами аппаратов управления в процессе служебной деятельности*.

Штатные ИРЗ подразделяются на *одноуровневые* (используемые в звеньях управления одного уровня, например — задачи предприятия) и *многоуровневые* (используемые в звеньях управления нескольких уровней, например — на предприятии, объединении и в министерстве).

Основными особенностями штатных ИРЗ, непосредственно следующими из их назначения, являются *высокая достоверность результатов* расчетов и *оперативность их получения*. Кроме того,

штатные задачи должны обеспечивать *простоту и удобство общения с пользователем* в процессе его работы на ЭВМ.

Исследовательской называется информационная или расчетная задача, *используемая должностными лицами при проведении научно-исследовательских работ, обосновании перспективных программ развития, прогнозирования экономических ситуаций и т. п.* Как правило, исследования проводятся с использованием математических моделей.

Исследовательские модели не имеют жестких требований по оперативности работы, поэтому они позволяют обеспечить широкий учет различных факторов при моделировании. Кроме того, исследовательские задачи должны обеспечивать легкость изменения (при необходимости) алгоритма своей работы в ходе исследований. При этом трудно обеспечить простоту и удобство работы с задачей. Исследовательские задачи в ряде случаев могут рассматриваться в качестве прототипов штатных задач, хотя это возможно далеко не всегда (подобнее об этом будет сказано ниже — в п. 8.2.2).

Вопросы для самопроверки

1. Что является объектом информатики?
2. Что является целью любой информационной технологии?
3. Какой признак лежит в основе классификации автоматизированных информационных систем? Вспомните эту классификацию.
4. Дайте определение математического обеспечения ЭВМ (АИС).
5. Дайте определение программного обеспечения ЭВМ (АИС).
6. Назовите основные составляющие общего системного программного обеспечения.
7. Назовите основные составляющие общего прикладного программного обеспечения.
8. В каких случаях и кем разрабатываются элементы специального системного программного обеспечения?

9. Поясните место информационных и расчетных задач в составе программного обеспечения ЭВМ (АИС).

10. В чем принципиальное различие алгоритмов информационных и расчетных задач?

11. Какие информационные и расчетные задачи называют штатными? исследовательскими?

8.2. Организационные основы проектирования элементов специального программного обеспечения

8.2.1. Основные требования к информационным, расчетным задачам и их комплексам

Информационные, расчетные задачи и их комплексы (ИРЗ и К) составляют основу любой АИС, определяют ее возможности по автоматизации профессиональной деятельности. Ввиду особой важности и значимости этих элементов специального программного обеспечения (СПО) их разработка организуется в соответствии с требованиями федеральных указов, законов, циркуляров, директив, ГОСТ и других руководящих документов [13, 14]. Перечислим эти требования, а затем рассмотрим каждое из них подробнее.

- достоверность результатов использования ИРЗ и К;
- оперативность получения результатов;
- соответствие ИРЗ и К уровню руководства;
- системный подход к созданию и применению СПО;
- обеспечение безопасности обрабатываемой информации.

Достоверность результатов

Под достоверностью результатов использования ИРЗ (расчета, моделирования) будем понимать соответствие значений параметров, получаемых в результате решения задачи, их требуемым (“истинным”) значениям.

Возможными *причинами недостоверности* получаемых в процессе расчетов результатов являются:

- неадекватность применяемой математической модели операции (процесса, явления);
- низкая точность вычислений;
- ошибки в алгоритме переработки информации, в соответствии с которым работает задача;
- ошибки пользователя при проведении расчетов;
- ошибки (сбои) в работе ЭВМ.

Под *адекватностью* в теории систем понимается степень соответствия используемой математической модели реальному процессу (системе, объекту). Следовательно, для оценки адекватности математической модели необходимо провести реальную операцию, осуществить математическое моделирование этой же операции в тех же условиях и сравнить реальные результаты операции с результатами моделирования, используя некоторый показатель, например показатель эффективности операции. Если результаты реальной операции будут хорошо согласовываться с результатами моделирования, то это означает, что используемая математическая модель в данных условиях проведения операции является адекватной реальному процессу (системе, объекту). Важно отметить, что в этом случае можно количественно оценить адекватность модели в рамках суждений типа "результаты моделирования расходятся с реальными не более чем на 10 процентов".

Формально оценить адекватность модели не всегда удается, поскольку не всегда возможно проведение реальной операции для сравнения с результатами моделирования (например, для экологических или крупномасштабных экономических моделей). В таких условиях под адекватностью принято понимать степень доверия должностного лица к результатам моделирования, используемым для принятия решений. При этом невозможно ввести показатель, объективно характеризующий степень адекватности модели. Модель может быть или адекватной, или неадекватной. Должностному лицу следует сделать вывод об адекватности модели на основании анализа ее существа и полноты учета в ней всех факторов, влияющих на проведение операции в конкретных условиях.

Низкая точность вычислений также может стать причиной недостоверности получаемых результатов расчета. Существуют две возможные причины возникновения ошибок вычислений: методические ошибки и ошибки округления. Методические ошибки связаны с использованием приближенных численных методов (например, при использовании метода численного интегрирования или дифференцирования функций). Ошибки округлений связаны с тем, что числа в ЭВМ представляются всегда с некоторой точностью, определяемой количеством значащих цифр в записи числа (для современных ЭВМ такие ошибки практически всегда связаны с неверными действиями программистов, в частности — при выборе форматов обрабатываемых данных).

Ошибки в алгоритме переработки информации, в соответствии с которым работает ЭВМ, являются достаточно редким источником недостоверности результатов расчетов и, как правило, бывают связаны с тем, что в алгоритме задачи не учитываются все возможные варианты исходных данных. При некоторых вариантах исходных данных могут возникнуть ситуации, когда алгоритм задачи работает с ошибками. Поэтому при создании алгоритма задачи необходимо тщательно проанализировать возможные значения исходных данных и определить их допустимые значения. Выявление ошибок в алгоритме переработки информации является одной из важнейших целей при проведении контрольных расчетов на этапе приемки И и РЗ.

Ошибки пользователя при проведении расчетов являются на первый взгляд ошибками, которые невозможно исключить за счет создания специальных алгоритмических и программных средств. Тем не менее существуют способы уменьшения возможностей для появления таких ошибок (конечно, имеются в виду непреднамеренные, “случайные” ошибки). Речь идет о программном контроле вводимой пользователем информации. Эта информация может включать значения параметров или команды. Как правило, при вводе параметров можно программно проконтролировать допустимость значения вводимого параметра, причем ограничения на значения параметра могут быть как постоянными, так и изменяться в зависимости от значений других параметров. Например, в зада-

че планирования транспортной операции по доставке потребителям какой-либо продукции допустимые значения скорости движения зависят от типов транспортных средств, участвующих в операции, и состояния дорог на маршрутах движения.

Что касается контроля команд, вводимых пользователем, то он может включать проверку допустимости данной команды на конкретном этапе работы с задачей (например, проверка наличия всех необходимых исходных данных перед выполнением команды начала расчета), а также выдачу на экран монитора запроса для подтверждения пользователем намерения выполнить какую-либо важную команду (например, при уничтожении каких-либо данных на экран монитора выводится вопрос типа. Вы действительно хотите уничтожить эти данные? и требуется утвердительный ответ пользователя для выполнения команды). Кроме того, особо ответственные команды могут предусматривать запрос на подтверждение полномочий на их проведение (например, ввод пароля).

Ошибки (сбои) в работе ЭВМ могут повлиять на достоверность результатов расчетов, если они не селектируются техническими средствами и операционной системой. Единственным средством исключения неселектируемых ошибок (сбоев) в работе ЭВМ является повторное решение задачи. Поэтому наиболее ответственные расчеты должны дублироваться на другой ЭВМ и (или) с использованием другой задачи, имеющей аналогичный алгоритм.

Оперативность результатов

Под *оперативностью* получения результатов расчетов на ИРЗ понимается возможность практического использования результатов их решения (расчетов, моделирования) либо в реальном ритме работы должностных лиц, либо за заданное время. Задача обладает требуемой оперативностью решения, если время работы пользователя с ней обеспечивает своевременное применение получаемых результатов в профессиональной деятельности. *Время работы с задачей* включает время на настройку (при необходимости) программного обеспечения (а иногда — и технических средств), подготовку исходных данных, ввод их в ЭВМ, проведение расчетов и выдачу результатов в виде, удобном для дальнейшего использования.

Таким образом, оперативность получения результатов расчетов является интегральной характеристикой, которая включает в себя не только скорость вычислений по алгоритму задачи, но и скорость ввода исходных данных, а также получение результатов в виде, не требующем какой-либо дополнительной обработки (переписывания, перепечатывания и т. д.). Поэтому при создании ИРЗ необходимо предусматривать минимально необходимый объем исходных данных, вводимый пользователем при использовании задачи, а также удобство их ввода.

Соответствие уровню руководства

Под требованием соответствия ИРЗ и К уровню руководства понимается:

- использование в них информации с детализацией и точностью, которыми располагает данное должностное лицо (должностные лица), работающее с задачей;
- представление результатов в наглядном (привычном для пользователя) виде, соответствующем форме и содержанию реальных документов;
- применение показателей, имеющих для конкретного должностного лица ясный технический, оперативный и физический смысл (так называемых *транспарентных* показателей).

Системный подход

Требование системного подхода означает, что все создаваемые ИРЗ и К должны быть составными элементами общей системы задач, т. е. они должны быть согласованы между собой по цели и назначению, составу учитываемых факторов и ограничений, содержанию и формам входных и выходных документов, показателей, критериев и нормативов, структуре и содержанию информационной базы, принципам защиты обрабатываемой информации.

Обеспечение безопасности информации

Требование обеспечения безопасности обрабатываемой информации заключается в исключении возможности уничтожения или искажения информации, обрабатываемой на ЭВМ, а также

возможности несанкционированного получения этой информации не допущенными к ней лицами. Выполнение данного требования достигается осуществлением комплекса организационных мероприятий и технических мер.

8.2.2. Принципы разработки специального программного обеспечения

Помимо основных требований к создаваемым ИРЗ и К руководящими (нормативными) документами определены и основные принципы разработки и поддержания в работоспособном состоянии элементов СПО. Руководство данными принципами является обязательным и позволяет создавать и применять ИРЗ и К. Сформулируем эти принципы применительно к средствам автоматизации наиболее сложной области профессиональной деятельности — управлению сложными человеко-машинными системами экономического назначения:

- централизованная разработка по единому плану и замыслу на общих информационных и математических основах;
- конкретность предназначения создаваемых задач и их комплексов;
- непосредственное руководство и участие в создании задач предприятий и фирм (организаций), в интересах которых они создаются;
- обеспечение возможности перестройки задач в процессе их эксплуатации применительно к конкретной обстановке;
- непрерывное сопровождение разработанных И и РЗ и их комплексов представителями заказчика и разработчика.

Централизованность разработки

Принцип *централизованной разработки* по единому плану и замыслу на общих информационных и математических основах используется при создании ИРЗ и К в рамках единой АСУ. Этот принцип должен неукоснительно соблюдаться при создании задач, результаты решения которых используются во всех или нескольких звеньях АСУ (например, задач, используемых для автоматизации управления отраслью экономики в министерстве).

Для обеспечения централизованной разработки ИРЗ в организации формируется и утверждается перспективный план создания элементов СПО. В перспективном плане указываются: название ИРЗ, ее заказчик и разработчик, а также срок создания задачи. Перспективный план, как правило, разрабатывается на пять лет. На основании перспективных планов разрабатываются годовые планы создания И и РЗ.

Принцип централизованной разработки И и РЗ *может не учитываться* подразделениями организаций, создающими одноуровневые задачи, предназначенные для применения в рамках данного подразделения и использующие автономные ЭВМ (например, персональные ЭВМ, не входящие в АСУ). При этом подразделения организации выступают в роли заказчика СМПО и осуществляют разработку (совершенствование) И и РЗ на основании своих перспективных планов. Отметим, что с насыщением аппаратов управления современной ЭВТ следовать этому принципу становится все труднее, и на первое место при его реализации выдвигаются организационные мероприятия.

Конкретность предназначения

Принцип *конкретности предназначения* создаваемых задач и их комплексов предполагает необходимость разработки элементов СПО, специально предназначенных для автоматизации решения конкретных задач управления.

На практике достаточно часто встречаются ситуации, когда создается задача для проведения *научных исследований* или в *учебных целях*, а затем предпринимаются попытки внедрения этой задачи (как правило, с некоторыми доработками) в той или иной организации.

Однако поскольку исследовательские и учебные задачи создаются в целях проведения научных исследований или обучения, они не могут эффективно использоваться, а зачастую являются просто непригодными для автоматизации управления реальными предприятиями и фирмами. Исследовательские задачи, обладая обычно высокими показателями достоверности результатов, часто имеют плохую оперативность расчетов и слабую эргономич-

ность (не отвечают требованиям удобства и простоты работы должностных лиц с задачей). Учебные задачи имеют высокую оперативность расчетов и эргономичность, но достоверность получаемых результатов, как правило, является недостаточной для использования при автоматизации управления на практике.

Таким образом, исследовательские и учебные задачи нуждаются в существенной переработке перед их внедрением в промышленность. Данная переработка является достаточно трудоемкой, причем затраты на доработку задачи соизмеримы с затратами на создание новой задачи.

Поэтому более правильным является путь, когда ИРЗ создается *специально* для автоматизации деятельности руководителя или должностных лиц предприятия или фирмы при решении конкретной задачи управления персоналом. При этом, конечно, необходимо использовать отдельные математические алгоритмы, фрагменты программ, а также опыт создания и использования исследовательских и учебных задач, являющихся прототипами разрабатываемых ИРЗ.

Непосредственное руководство заинтересованных предприятий и фирм (организаций)

Принцип *непосредственного руководства и участия* в создании ИРЗ и К предприятий и фирм (организаций), в интересах которых они создаются, является важнейшим принципом, лежащим в основе всей технологии создания СПО и обеспечивающим создание качественных задач для автоматизации управления персоналом фирм.

Разработчики задачи, как правило, плохо представляют себе специфику управления персоналом, а также роль создаваемой задачи в процессе управления и требования, предъявляемые к ней. Учет этой специфики и соответствующих требований к задаче должен проводиться в процессе разработки ее оперативной постановки, являющейся совместным документом заказчика и разработчика.

Непрерывный контроль со стороны заказчика на всех этапах создания И и РЗ позволяет избежать неправильного толкования

разработчиком положений и требований оперативной постановки задачи, своевременно устранять недостатки и тем самым ускорить создание и улучшить качество создаваемых задач.

Кроме того, участие в разработке оперативной постановки и контроля результатов отдельных этапов создания ИРЗ позволит должностным лицам, для которых создается задача, глубже понять механизмы переработки информации в задаче. Понимание должностными лицами этих механизмов обеспечит грамотное и эффективное применение задач в процессе решения задач управления.

Возможность перестройки

Принцип обеспечения *возможности перестройки* задач в процессе их эксплуатации применительно к конкретной обстановке предполагает, что при создании ИРЗ необходимо более полно учесть возможные изменения обстановки, внешних условий, а также изменения характеристик и условий применения создаваемой продукции, которые вызовут необходимость корректировки алгоритмов и программ ИРЗ.

Конечно, заранее предусмотреть и оговорить какие-либо конкретные изменения (кроме плановых — например, модернизации продукции или договорных ограничений) невозможно. Тем не менее при разработке задач необходимо учитывать возможные направления изменения тех или иных параметров и создавать такие задачи, которые позволили бы с минимумом затрат проводить их корректировку.

Непрерывное сопровождение СПО заказчиком и разработчиком

Непрерывное сопровождение разработанных ИРЗ и К представителями заказчика и разработчика является основным условием, обеспечивающим поддержание задач и их комплексов в готовности к применению. В функцию представителей заказчика при сопровождении ИРЗ и К входит обеспечение работоспособности используемых задач, а также анализ процесса их эксплуатации и выработка предложений по их совершенствованию. Представители разработчика при сопровождении ИРЗ устраняют недостатки,

выявляемые в процессе эксплуатации, и проводят совершенствование задач в плане повышения их эксплуатационных характеристик.

Перечисленные выше основные принципы и требования являются нормативной базой при разработке и применении СПО в АИС (рис. 8.2.1).

Рис. 8.2.1. Требования к СПО и принципы его разработки

Конечно, уровни обеспечения этих требований существенно зависят от класса задачи, ее назначения и особенностей применения. В зависимости от существа и особенностей применения создаваемых задач перечень требований к ним может расширяться или сужаться. Формирование конкретных требований к создаваемым задачам осуществляется совместными усилиями представителей заказчика и разработчика на этапе разработки технического задания и утверждается заказчиком.

8.2.3. Основы алгоритмизации задач

Этап алгоритмизации является важным этапом создания информационных и расчетных задач. Сущность этого этапа состоит в разработке обобщенных и детальных алгоритмов создаваемой задачи. Как уже отмечалось, при традиционном (классическом) подходе к созданию элементов программного обеспечения данный этап занимает промежуточное положение между этапами постановки задачи и программированием (кодированием). Разработанные алгоритмы задачи входят в состав документов, которые должны представляться разработчиком задачи заказчику. Алгоритмы задачи используются должностными лицами, работающими с задачей, для ее изучения и оценки качества получаемых результатов при ее практическом использовании.

Понятие алгоритма

Решение многих вопросов практики связано с выполнением некоторой последовательности отдельных действий (элементарных операций). Если общее количество таких операций не является малым, а последовательность их выполнения может быть переменной в зависимости от исходного состояния и промежуточных результатов, то варианты объединения этих элементарных операций в единые совокупности, нацеленные на решение определенных задач практики, могут представлять самостоятельный интерес. В этом случае целесообразно введение понятия алгоритма.

Под *алгоритмом* понимается совокупность предписаний и правил, которые однозначно определяют содержание и последо-

вательность действий исполнителя и выполняя которые за конечное число шагов им достигается требуемый результат [9]. Элементарные операции, которые выполняет исполнитель, реализуя алгоритм, называются *пунктами* или *шагами алгоритма*. Процесс составления алгоритма называется алгоритмизацией.

Существенным в понятии алгоритма являются следующие его особенности.

1. В понятии алгоритма не конкретизируется исполнитель, поэтому им может являться как техническое (кибернетическое) устройство, так и человек или коллектив.

2. Понятие алгоритма требует однозначного восприятия исполнителем составляющих алгоритма (при выполнении алгоритма в одних и тех же исходных условиях различными исполнителями должен получаться один и тот же результат).

3. Алгоритм перестает быть алгоритмом, если предписываемая им последовательность действий является (или может являться при определенных условиях) бесконечной. Алгоритм всегда конечен.

Понятие алгоритма не детализирует характер выполняемых в его рамках операций (действий). Эти операции могут проводиться как над материальными объектами, так и над данными (информацией). Алгоритмы последнего типа составляют основу информационных технологий и являются предметом последующего рассмотрения.

Примеры алгоритмов приведены ниже.

В качестве первого примера представим алгоритм действий пешехода при переходе улицы. Данный алгоритм ориентирован на выполнение его человеком и предусматривает выполнение как информационных операций, так и действий материальных объектов.

Итак, алгоритм перехода улицы.

П. 1. Определить, является ли переход управляемым (т. е. установлен ли на нем светофор). Если светофор имеется, перейти к п. 3, если отсутствует — к п. 2.

П. 2. Если переход является неуправляемым, следует посмотреть налево и, если отсутствуют движущиеся по данной полосе

(полосам) машины, выйти на середину дороги. Если движущиеся машины мешают переходу, пропустить их. Посмотреть направо и, если отсутствуют движущиеся по данной полосе (полосам) машины, завершить переход улицы. Если движущиеся машины мешают переходу, пропустить их и после этого завершить переход. После этого перейти к п. 4.

П. 3. Если переход является управляемым, следует убедиться, горит ли зеленый сигнал светофора. Если "да", перейти улицу. Если "нет", ждать его появления, после чего перейти улицу. После этого перейти к п. 4.

П. 4. После перехода продолжить запланированные действия (алгоритм завершен).

Заметим, что приведенный алгоритм пригоден, вообще говоря, лишь при выполнении нескольких условий. Во-первых, при использовании "в районе операции по переходу" правостороннего дорожного движения (см. п. 2). Во-вторых, при наличии такой регулировки светофора, которая позволяет пешеходу (успеть) перейти улицу до изменения цвета сигнала — в противном случае следует в п. 3 предусмотреть остановку на середине улицы ("островке безопасности"). В-третьих, наконец, при выполнении всеми участниками движения (прежде всего — водителями) правил дорожного движения. Таким образом, следует признать, что практически всегда алгоритмы представляют собой модели действий в некоторых условиях, которые могут отражать субъективные предпочтения своих авторов и которые следует четко определить. Так, например, родители вправе потребовать от детей переходить улицы только в тех местах, где установлены светофоры, что упрощает алгоритм действий.

На рис. 8.2.2 представлен алгоритм для проведения расчетов, которые реализуют следующую математическую зависимость:

$$Y = \begin{cases} A + B, & \text{если} \quad B \geq A; \\ A - B, & \text{если} \quad B < A. \end{cases}$$

На рис. 8.2.3 представлен алгоритм, который позволяет упорядочить по убыванию значений K чисел в массиве данных C(1:K).

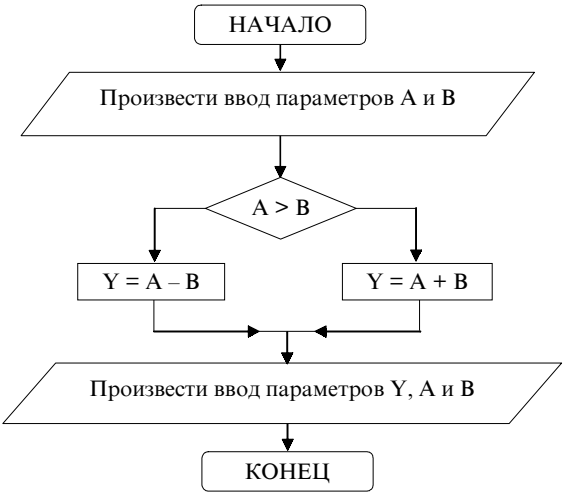


Рис. 8.2.2. Алгоритм вычисления функции

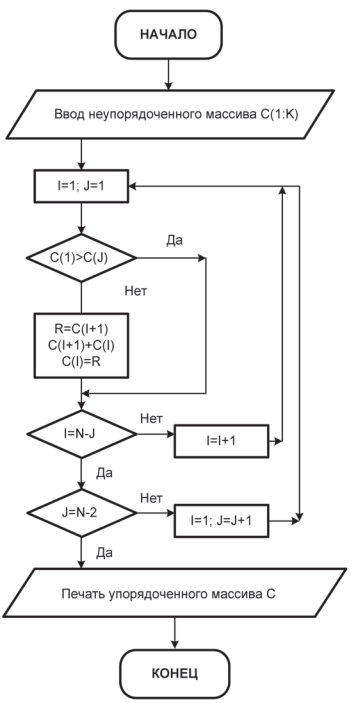


Рис. 8.2.3. Алгоритм упорядочения массива чисел

Алгоритмы, представленные на рис. 8.2.2 и 8.2.3, могут быть реализованы как человеком, так и техническим средством.

Необходимо обратить внимание на то, что алгоритм, представленный на рис. 8.2.3, является не единственно возможным. Существует много вариантов алгоритмов упорядочения чисел в массиве. Таким образом, при разработке алгоритма возникает проблема выбора лучшего по какому-либо критерию (правилу) алгоритма из множества возможных.

Формы записи алгоритма

К настоящему времени сформировались и нашли широкое распространение шесть различных форм записи алгоритмов [6, 9, 10]. Такими формами являются:

- табличная;
- словесная;
- запись алгоритма в виде блок-схемы;
- операторная;
- форма, использующая псевдокоды;
- языки программирования.

Табличная форма записи алгоритма часто используется для ведения учетной документации. При разработке программ на ЭВМ такая форма практического применения не находит. Однако табличная форма представления алгоритма часто выступает в виде исходной информации о механизме выполнения действий на практике. Кроме того, такая форма записи алгоритмов получила "новую жизнь" с появлением и широким распространением табличных процессоров (Excel; Lotus 1-2-3). Примеры табличной формы записи алгоритма представлены на рис. 8.2.4 и 8.2.5.

Учет поступления моторного масла (ММ) на склад ГСМ

Номер заказа	Количество полученных бочек с ММ, шт.	Вес ММ в бочке, т	Количество полученного ММ без тары, т	Всего ММ, т	% ММ в таре
3.7/12	10	0,2	15	15	25%

Рис. 8.2.4. Табличная форма записи алгоритма в учетной документации

N	a	b	c	a^2	$\sqrt{c}$	$A = a^2 - b$	$B = A/\sqrt{c}$	sin(B)
1	4	16	9	16	3	0	0	0

Рис. 8.2.5. Табличная форма записи алгоритма вычисления арифметического выражения

Словесная форма записи алгоритма предполагает его описание с помощью слов и предложений. Именно в словесной форме записан приведенный в начале пункта алгоритм перехода улицы.

Данная форма является наиболее простой и естественной для самого широкого круга людей. Наибольшее распространение эта форма находит на начальном этапе разработки программных систем при формировании основных элементов их замысла, а также при описании динамики взаимодействия основных блоков (модулей). Необходимо иметь в виду, что словесная форма записи допускает запись любого сколь угодно сложного алгоритма.

Недостатками словесной формы записи алгоритма являются ее громоздкость и отсутствие наглядности для сложных алгоритмов.

Блок-схемы алгоритмов получили большое распространение благодаря наглядному и компактному представлению алгоритмов в графической форме. Каждому пункту (шагу) алгоритма в блок-схеме ставится в соответствие отдельный блок. Блоки между собой соединяются стрелками, которые определяют порядок реализации действий, предусмотренных в каждом блоке. Приняты определенные стандарты графического изображения блоков при описании информационных алгоритмов. Основными из них являются такие геометрические фигуры, как прямоугольник, ромб, параллелограмм, трапеция, овал и др.

В прямоугольнике помещаются действия по безусловной переработке информации. Блоки в виде ромбов описывают проверку различных условий, связанных с текущим значением перерабатываемой информации. Овалы обозначают начало и окончание работы алгоритма. Для записи действий внутри блоков используется естественный язык или математическая символика.

Примеры записи алгоритмов с использованием блок-схем представлены на рис. 8.2.2 и 8.2.3.

Операторная форма — это запись алгоритма в виде условных буквенно-цифровых символов-операторов, графических знаков-стрелок и знаков препинания. Операторы подразделяются на основные — арифметические, логические и вспомогательные — ввода-вывода, останова и др. Для операторов определенных типов принимаются одинаковые обозначения в пределах всего алгоритма. Например, А — арифметический оператор, Р — логический оператор, П — оператор ввода-вывода, Я — оператор останова и т. д. Операторы в общем случае могут состоять из сочетания нескольких букв.

Запись алгоритма в операторной форме производится с соблюдением следующих правил:

- все операторы записываются в одну строку;
- каждый из операторов снабжается индексом, указывающим порядковый номер оператора (нумерация операторов сквозная);
- если символы операторов стоят рядом, то это значит, что оператор, стоящий справа, получает управление от оператора, стоящего слева;
- если оператор, стоящий справа, не получает управление от стоящего слева оператора, то между ними ставится точка с запятой;
- передача управления оператору, не стоящему справа, обозначается стрелкой.

Пример записи в операторной форме алгоритма определения корней квадратного уравнения представлен на рис. 8.2.6.

$$П_1 П_2 A_3 P_4 \overset{\uparrow 6}{\underset{\downarrow 7}{A_5}} ; A_6 \overset{\downarrow 4}{\underset{\uparrow 5}{П_7}} Я_8$$

$П_1$ — оператор ввода исходных данных (a, b, c);
$П_2$ — оператор печати исходных данных;
A_3 — оператор расчета величины $D = b^2 - 4ac$;
P_4 — оператор проверки условия $D >= 0$;
A_5 — оператор расчета действительных корней;
A_6 — оператор расчета мнимых корней;
$П_7$ — оператор вывода на печать значений корней;
$Я_8$ — оператор останова.

Рис. 8.2.6. Операторная форма записи алгоритма вычисления корней квадратного уравнения

Операторная форма записи алгоритмов является самой компактной. Однако, в силу малой наглядности содержания действий, выполняемых каждым оператором, при разработке программных систем рассматриваемая форма применяется редко.

Псевдокод представляет собой систему обозначений и правил, предназначенных для единообразной записи различных алгоритмов. Псевдокоды занимают промежуточное положение между естественным и формальным языком. В псевдокоде не приняты строгие синтаксические правила для записи команд в отличие от формальных языков (языков программирования). Это облегчает запись алгоритма на стадии его проектирования и дает возможность использовать более широкий набор команд, рассчитанный на абстрактного исполнителя. Однако в псевдокоде имеются некоторые конструкции, присущие формальным языкам, что облегчает переход от записи на псевдокоде к записи на том или ином языке программирования. В частности, псевдокод, так же как и язык программирования, имеет ключевые слова с четким смыслом, определяющим некоторую последовательность действий.

Пример записи с использованием псевдокода алгоритма, реализующего подсчет числа символов в кодограмме, приведен на рис. 8.2.7.

НАЧАЛО АЛГОРИТМА
записать в счетчик слов 0;
установить указатель на первую запись текста;
ПОКА символ есть “ . ” ПОВТОРЯТЬ

НАЧАЛО ПОВТОРЕНИЯ

ЕСЛИ символ есть пробел
ТО счетчик слов увеличить на 1;
перевести указатель на следующий символ;

КОНЕЦ ПОВТОРЕНИЯ
Взять число в счетчике слов в качестве результата;

КОНЕЦ АЛГОРИТМА

Рис. 8.2.7. Запись алгоритма с использованием псевдокода

В настоящее время для записи сравнительно несложных алгоритмов довольно часто используют языки программирования (как правило, высокого уровня). Очевидны по крайней мере два недостатка такой формы записи: исключительно низкая наглядность (кроме тривиально простых алгоритмов) и необходимость знания используемого языка программирования. Приведем простейший пример фрагмента программы перевода долларов в рубли, созданной в среде программирования Borland C++ Builder (язык — C++):

```
/* Программа перевода долларов в рубли */
{
int DollarsNum; RateValue ; // Исходные данные
int Result ; // Результат
DollarsNum = StrToInt (Dollars->Text) ;
RateValue = StrToInt (Rate->Text) ;
Result = DollarsNum * RateValue ;
Label1->Caption = IntToStr(Result) ;
}
```

В результате выполнения такой программы в поле надписи Label1 появится сумма в рублях — результат перевода значения поля ввода Edit1 Dollars (суммы в долларах) по курсу, заданному в поле ввода Edit2 Rate.

Структуры алгоритмов

Рассмотренные выше примеры записи алгоритмов указывают на то, что каждый из них, в какой бы форме он ни был записан, представляет собой некоторую структуру, состоящую из отдельных базовых команд. Данные команды в зависимости от своей сложности делятся на простые и составные.

Простая команда соответствует одному элементарному шагу алгоритма. Применительно к алгоритму рис. 8.2.2 такими элементарными командами являются команды Y = A + B и Y = A – B. Простые команды в блок-схемах изображаются в виде блока, имеющего один вход и один выход.

Из совокупности простых команд образуются составные команды, которые имеют сложную структуру. Из всех сложных команд целесообразно выделить три основных типа:

• команда следования;
• команда ветвления;
• команда повторения (цикла).

Команда *следования* образуется из простых команд следующих строго одна за другой. На рис. 8.2.8 представлена блок-схема команды следования.

Рис. 8.2.8. Блок-схема команды следования

С помощью команды *ветвления* осуществляется выбор одного из нескольких возможных вариантов действий в зависимости от выполнения некоторых условий. Команда ветвления может использоваться в сокращенной форме, когда при выполнении условия выполняется одно действие, а при невыполнении условия вообще никаких действий не выполняется.

На рис. 8.2.9 представлена блок-схема команд ветвления в полной (а) и сокращенной (б) форме.

Команда *ветвления (цикла)* реализует возможность многократного выполнения одной и той же совокупности определенных действий. Имеются два типа команд повторения:

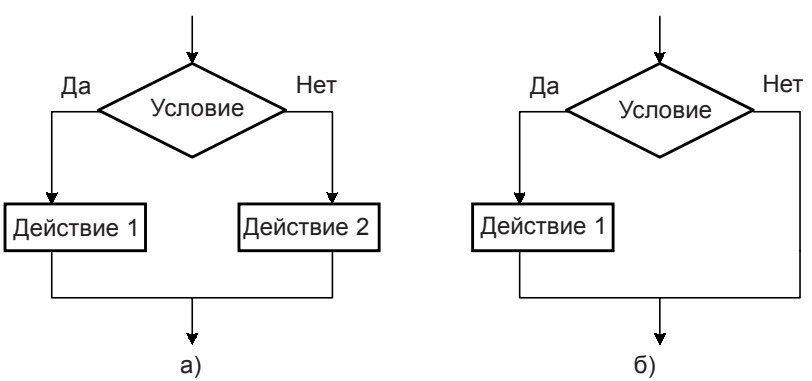


Рис. 8.2.9. Команды ветвления в полной (а) и сокращенной (б) форме.

• команда повторения с предусловием;
• команда повторения с постусловием.

Исполнение команд повторения с *предусловием* состоит в том, что сначала проверяется условие, контролирующее необходимость повторения действий, а затем выполняются соответствующие действия. Основной особенностью команды повторения с предусловием является то, что действия могут не выполниться ни разу, если при обращении к команде повторения проверка условия даст отрицательный результат.

При выполнении команды повторения с *постусловием* контроль необходимости очередного повторения проводится после выполнения действий. В отличие от предыдущей команды, команда повторения с постусловием предусматривает выполнение действий хотя бы один раз.

Блок-схемы команд повторения с предусловием (а) и постусловием (б) представлены на рис. 8.2.10.

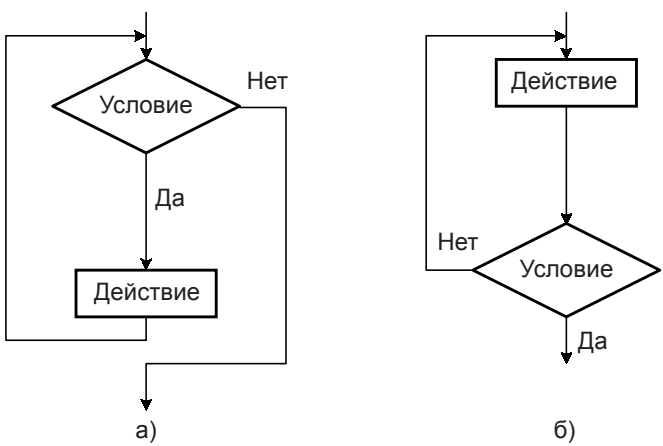


Рис. 8.2.10. Команды повторения (цикла) с предусловием (а) и постусловием (б)

Созданием алгоритмов программных систем завершается этап их технического проектирования, за которым следует этап рабо-

чего проектирования, предназначенный для написания и отладки программ на ЭВМ в соответствии с разработанными алгоритмами.

В заключение отметим три важных обстоятельства. Во-первых, при создании информационных и расчетных задач, как правило, сначала разрабатываются *обобщенные*, а затем — *детальные* алгоритмы, по которым и осуществляется собственно программирование (кодирование). При этом для записи обобщенных и детальных алгоритмов могут использоваться различные формы. Во-вторых, перечисленные выше типовые конструкции алгоритмов легли в основу одной из довольно широко применяемых технологий создания программ — *структурного программирования*, позволяющего разрабатывать весьма наглядные древовидные архитектуры программ, которые легко читаются и — при необходимости — модифицируются. Наконец, в-третьих, существуют более современные средства алгоритмизации задач (CASE-средства), в которых для записи алгоритмов используются специальные диаграммы различного вида (см. п. 8.3 настоящего учебника).

8.2.4. Содержание работ на этапах создания информационных, расчетных задач и их комплексов

Порядок создания информационных, расчетных задач и их комплексов (ИРЗ и К) определен федеральными законами Российской Федерации, а также государственными стандартами. В создании задач и их комплексов участвуют две стороны: заказчик и разработчик. Если разработчиков несколько, то среди них определяется *головной разработчик (исполнитель)* и *соисполнители*. Возможно также существование *нескольких заказчиков*. Тогда среди них выделяется *головной заказчик*, а остальные заказчики называются *созаказчиками*.

Процесс создания ИРЗ и К в принципе одинаков и включает следующие этапы [54]:

- разработки технического задания;
- эскизного проектирования;
- технического проектирования;
- рабочего проектирования.

По решению заказчика, сформированному в техническом задании (ТЗ), допускается объединение отдельных этапов разработки, изменение их содержания или введение других этапов. Продолжительность этапов, а также уровень, с которого начинается разработка задач и их комплексов, определяется в каждом конкретном случае, исходя из имеющегося научно-методического задела по данной проблеме. Каждый этап завершается в порядке, установленном ТЗ. В частности, итоги каждого этапа рассматриваются заказчиком.

Для повышения оперативности взаимодействия заказчика и разработчика, непрерывного контроля за деятельностью разработчика желательно из состава заказывающей организации выделить сотрудника, которому поручается научно-техническое и организационное сопровождение всех видов работ по созданию ИРЗ и К. Этот сотрудник называется сотрудником сопровождения.

Этап разработки технического задания

ТЗ является исходным документом, устанавливающим основное *назначение*, технические *характеристики* и *требования*, предъявляемые к создаваемым задачам и их комплексам, а также *порядок работ* на всех этапах и *сроки* их проведения. ТЗ формируется заказчиком совместно с разработчиком.

Для проведения работ на этом этапе заказчик может создавать рабочие группы из своих представителей, представителей разработчика и других специалистов (экспертов) в зависимости от характера создаваемых задач и моделей.

При разработке ТЗ осуществляется:

• проведение *информационного обследования* объекта автоматизации и уточнение функций и задач управления, подлежащих автоматизации;

• *определение* необходимого *состава* комплекса И и РЗ;

• разработка *оперативных постановок задач*;

• формирование *задания* (определение разработчиков, сроков и порядка создания задач и их комплексов, сроков, объемов и порядка финансирования работ, мер по охране авторских прав и др.) и *исходных данных*.

Проведение информационного обследования объекта автоматизации и уточнение функций и задач управления, подлежащих

автоматизации, являются необходимым элементом этапа разработки ТЗ. При информационном обследовании анализируется процесс функционирования объекта автоматизации по переработке информации и определяются те элементы этого процесса, которые могут или должны быть возложены на ЭВМ. Одним из результатов информационного обследования является состав комплекса задач, которые должны быть разработаны.

Основным документом, содержащим всю информацию о создаваемой задаче (или комплексе задач), ее назначении и требованиях к ней, является *оперативная постановка задачи (или комплекса задач)*, которая оформляется как *обязательное приложение к ТЗ*.

ТЗ в целом и оперативные постановки задач подписываются головным разработчиком, согласовываются с организациями-соисполнителями, аппаратом управления, на котором будет внедряться задача, и *утверждаются заказчиком*. Проведение разработки задач и их комплексов без утвержденного ТЗ не допускается. В процессе дальнейших работ по созданию АИС или ее элементов при невозможности выполнения требований оперативной постановки она может корректироваться *с разрешения заказчика* на любом этапе создания и внедрения ИРЗ и К.

Этапы эскизного и технического проектирования

После утверждения заказчиком ТЗ разработчик приступает к этапу эскизного проектирования (ЭП), который часто объединяют с этапом технического проектирования (ТП). На этапах эскизного и технического проектирования осуществляются следующие действия:

- определение *принципов построения*, состава и структуры технических и программных средств ИРЗ и К (этап ЭП);
- определение *обобщенного алгоритма* функционирования, назначения и порядка работы элементов задач и их комплексов (этап ЭП);
- определение *содержания и общих характеристик информационных связей* между элементами задач (комплексов задач) (этап ЭП);
- определение *состава необходимого программного обеспечения* (ПО) для создания задач и их комплексов (этап ЭП);
- выбор *используемых математических методов* и математическое описание моделей экономических операций (этап ЭП);

• оценка *возможности выполнения основных требований* оперативной постановки задачи (этап ЭП);

• разработка *детальных алгоритмов* задач и комплексов, их информационного и лингвистического обеспечения (этап ТП);

• проектирование и *разработка необходимых баз данных* (этап ТП).

Алгоритмы И и РЗ и К разрабатываются в строгом соответствии с утвержденным ТЗ и оперативными постановками задач и являются определяющими документами для последующего написания программ. Схемы алгоритмов и программ выполняются в соответствии с нормативными требованиями.

Этап рабочего проектирования

На этапе *рабочего проектирования* в соответствии с разработанными ранее алгоритмами осуществляется разработка программ, их отладка и экспериментальная проверка (испытания) на ЭВМ и оформление документации по разработанной задаче (или комплексу задач).

Перед сдачей отлаженных программ заказчику разработчик проводит их испытания с целью проверки соответствия программного продукта требованиям ТЗ. В процессе испытаний проверяются:

• *достоверность результатов* расчетов в различных вариантах исходных данных и, в частности, адекватность математических моделей операций;

• *характер влияния различных исходных данных* на результаты расчета (моделирования);

• *надежность* применяемых технических и программных средств защиты данных;

• *оперативность* полученных результатов расчетов;

• *удобство работы* с ЭВМ в процессе расчета или моделирования;

• *качество* разработанных алгоритмов и программ и т. д.

Проверка *достоверности результатов* проводится на вариантах исходных данных с реальной или учебной информацией, обеспечивающих проведение всесторонней оценки получаемых результатов путем сравнения с результатами проведенных реальных экономических операций. Для окончательной оценки достоверности результатов расчетов (моделирования) могут привлекаться компетентные эксперты.

Все работы по проверке готовности программного продукта проводятся *на технической базе разработчика*. В работе по проверке (испытанию) ИРЗ и К участвует сотрудник сопровождения. Обобщенные результаты экспериментальной проверки разработанных задач и комплексов представляются заказчику вместе с отчетными материалами по программному изделию, подготовленному к сдаче.

На каждую ИРЗ и в целом комплекс задач оформляется отчетная документация в четырех частях:

Часть 1. Оперативная постановка задачи.

Часть 2. Алгоритмы задачи.

Часть 3. Описание программы. Инструкция оператору-программисту по ее применению. Программы на магнитных носителях и их распечатки (тексты программ).

Часть 4. Инструкция должностному лицу по использованию задачи (комплекса задач).

Каждая часть документации *оформляется отдельной книгой (или несколькими книгами)*. Части 1, 2, 4 используются специалистами аппарата управления при изучении сущности задачи и порядка работы с ней. В вычислительный центр (ВЦ) соответствующей организации документация передается в полном объеме.

Помимо указанной отчетной документации после завершения каждого этапа разработки задачи (комплекса) разработчиком выпускается и представляется заказчику отчет. Этим обеспечивается объективный контроль за ходом создания задач. Порядок, сроки выпуска и содержание таких отчетов оговариваются в техническом задании. Анализ отчетов и выдача заключений по результатам каждого этапа работ производится, как правило, при активном участии сотрудника сопровождения.

Приведенные выше этапы создания задач (комплексов задач) и отчетность в процессе их создания не являются строго обязательными (кроме документации по готовым задачам и комплексам задач) и зависят от объема и сложности создаваемой задачи (комплекса задач). В любом случае обязательным документом является ТЗ, в котором оговаривается как содержание этапов создания задач, так и состав документов, разрабатываемых на каждом этапе.

8.2.5. Порядок внедрения информационных, расчетных задач и их комплексов

Прием в эксплуатацию разработанных ИРЗ и К осуществляется по приказу или директиве заказчика. Этим приказом заказчик назначает комиссию по приемке готового программного продукта, определяет ее состав и задачи, а также сроки и порядок ее работы.

В задачи комиссии входит:

• изучить документацию по задаче (комплексу задач), представленную разработчиком, определить ее качество, соответствие требованиям ТЗ и другим нормативным документам;

• установить соответствие алгоритмов и программ требованиям ТЗ и оперативной постановке задачи;

• проверить достаточность мер по обеспечению безопасности обработки информации и ее выдачи на автоматизированные рабочие места должностных лиц;

• провести контрольные расчеты на внедряемой задаче с целью проверки:

• работоспособности программ, достоверности, полноты и качества получаемых результатов в широком диапазоне исходных данных;

• практической пригодности и эффективности использования задачи в деятельности руководителя по управлению персоналом;

• технологичности, трудоемкости и временных характеристик основных этапов подготовки и ввода исходных данных, проведения расчетов и выдачи результатов;

• подготовить предложения о внесении изменений в методику работы должностных лиц с использованием результатов применения задачи;

• при необходимости определить перечень работ и продолжительность этапа опытной эксплуатации задачи (комплекса), а также составить план-задание на ее проведение.

Приведенный выше перечень задач комиссии по решению заказчика может быть расширен.

При приемке задачи (комплекса) головной разработчик обязан:

• не позднее чем за два месяца до начала работы комиссии представить заказчику и организациям, определенным заказчиком,

по одному экземпляру документации в согласованном с заказчиком объеме;

• представить эталонные программы с контрольными вариантами решений на машинных носителях в ВЦ, на котором планируется внедрение задачи (комплекса);

• выполнить контрольные расчеты по указанию заказчика и принять участие в анализе полученных результатов.

• устранить недостатки, выявленные в процессе приемки задачи (комплекса).

По указанию заказчика в приемке задачи (комплекса) могут принять участие заинтересованные организации (предприятия, фирмы), где планируется использование внедряемых задач и их комплексов. В этом случае заинтересованные организации обязаны:

• принять участие в подготовке контрольных вариантов решений;

• оценить возможности задачи, обеспечить проверку ее работоспособности в условиях, соответствующих особенностям работы данного предприятия;

• представить заказчику в установленные им сроки предложения и замечания о результатах проверки задачи (комплекса).

ВЦ, на котором планируется внедрение задачи, предоставляет технические средства, участвует в контрольных решениях и проверяет эксплуатационно-технические характеристики задачи (комплекса).

По результатам работы комиссия представляет заказчику акт по приемке задачи (комплекса). Копия акта комиссии высылается в адрес головного разработчика и является заключением заказчика на выполненное ТЗ.

ИРЗ (или их комплекс) допускается к штатной эксплуатации, если она отвечает требованиям ТЗ и является работоспособной в реальных условиях профессиональной деятельности. ИРЗ (или их комплекс) принимается в опытную эксплуатацию, если в ходе приемки выявлена необходимость в ее доработках, не оказывающих существенного влияния на достоверность результатов расчетов. Схема методики внедрения ИРЗ и их комплексов представлена на рис. 8.2.11.

Рис. 8.2.11. Схема методики приема элементов специального программного обеспечения

Для организации опытной эксплуатации заказчиком утверждается план-задание, в котором указываются:

- сроки опытной эксплуатации;
- содержание и порядок работы;
- меры по обеспечению безопасности обработки информации;
- способы оценки эффективности применения данной задачи (комплекса) в практической работе предприятия.

В ходе опытной эксплуатации организуется обучение персонала организации порядку работы с задачей (комплексом), изучение ее возможностей и проведение оценки эффективности ее применения. В практике должностных лиц в реальных условиях эксплуатации оценивается надежность средств и методов защиты информации, проводятся все необходимые доработки задачи.

Головной разработчик обязан участвовать на всех этапах опытной эксплуатации и устранять недостатки, выявленные как в процессе приемки задачи, так и в процессе опытной эксплуатации.

Опытная эксплуатация ИРЗ и их комплексов завершается актом, разрабатываемым организацией, в которой должна использоваться задача. Акт подписывается всеми участниками опытной эксплуатации и представляется заказчику.

Прием задачи (комплекса) осуществляется по директиве (приказу) заказчика. Директивой (приказом) устанавливается порядок применения задачи (комплекса) соответствующими должностными лицами.

8.2.6. Порядок использования информационных, расчетных задач и их комплексов в практике работы аппарата управления

Применение ИРЗ и К организуется и осуществляется на основании указаний руководителя предприятия, распоряжений вышестоящих организаций и других руководящих документов.

Документы, регламентирующие применение ИРЗ и К, должны включать:

- цели и сроки применения каждой задачи;
- порядок проведения расчетов (моделирования) в различных ситуациях, подготовки исходных данных, анализа промежуточных и

конечных результатов, выдачи их соответствующим должностным лицам и использования результатов решения в процессе управления;

• порядок обобщения опыта применения задач и их комплексов, разработки и реализации предложений по совершенствованию методов работы с использованием результатов расчетов;

• список сотрудников, выделенных для оперативного сопровождения задач и их комплексов;

• перечень мероприятий по исключению утечки информации в процессе производства расчетов и анализа их результатов (при необходимости);

• порядок подготовки и допуска персонала организации к работе с задачей и использованию результатов расчетов, а также порядок проведения необходимых периодических тренировок с этими сотрудниками;

• перечень мероприятий по поддержанию в работоспособном состоянии средств программного, технического и других видов обеспечения.

Ответственность за внедрение, освоение персоналом, применение, совершенствование задач и их комплексов, а также обеспечение безопасности информации в процессе ее обработки возлагается на руководителя соответствующей организации. Ответственность за поддержание задач и их комплексов в работоспособном состоянии возлагается на начальника ВЦ (если таковой имеется).

Оперативное сопровождение задач и их комплексов осуществляется выделенными для этой цели сотрудниками организации и включает:

• поддержание программ и средств их информационного обеспечения в работоспособном состоянии;

• подготовку предложений по совершенствованию оперативных постановок, алгоритмов и инструкций по использованию задач и их комплексов в связи с изменением самой экономической (политической) обстановки и взглядов на проведение экономического анализа, появлением новых технических и программных средств, новой организационной структуры организации, отрасли и т. д.;

• подготовку рекомендаций по совершенствованию методики работы персонала организации с использованием разработанных ИРЗ и К.

В процессе эксплуатации задач и их комплексов разработчик осуществляет научно-техническое сопровождение (авторский надзор), которое включает совершенствование математических методов, алгоритмов, программ и информационного обеспечения в целях повышения оперативно-технических характеристик задач и их комплексов.

Вопросы для самопроверки

1. Назовите и поясните основные требования к информационным и расчетным задачам и их комплексам.
2. Назовите и поясните основные принципы создания информационных и расчетных задач и их комплексов.
3. Поясните диалектическую связь между названными требованиями и принципами.
4. Дайте определение алгоритма.
6. Вспомните формы записи алгоритмов. Какая из них на ваш взгляд, является наиболее простой (понятной, наглядной)?
7. В чем существо операторной формы записи алгоритмов?
8. Почему сегодня можно говорить о достаточно широком распространении табличной формы?
9. Какие работы проводятся на этапах создания элементов СПО АИС? этапах эскизного и технического проектирования?
10. Что указывается в акте о приеме информационной (расчетной) задачи?
11. Кто несет ответственность за правильное использование специального программного обеспечения?

8.3. Понятие о CASE-технологиях

За последние десятилетия сформировалось новое направление в программотехнике — CASE (Computer-Aided Software/System Engineering) — в дословном переводе — разработка программного обеспечения систем при поддержке (с помощью) компьютера.

В настоящее время не существует общепринятого определения CASE, этот термин используется в весьма широком смысле. Первоначальное значение термина CASE, ограниченное вопросами автоматизации разработки только лишь программного обеспечения (ПО), в настоящее время приобрело новый смысл, охватывающий процесс разработки сложных автоматизированных информационных систем (АИС) в целом. Теперь под термином CASE-средства понимаются программные средства, поддерживающие процессы создания и сопровождения АИС, включая анализ и формулировку требований, проектирование прикладного ПО (приложений) и баз данных, генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управление и управление проектом, а также другие процессы [23, 24]. CASE-средства вместе с системным ПО и техническими средствами образуют *полную среду* разработки АИС.

CASE-средства позволяют не только создавать "правильные" продукты, но и обеспечивать "правильный" процесс их создания. Основная цель CASE состоит в том, чтобы отделить проектирование ПО от его кодирования и последующих этапов разработки. При использовании CASE-технологий изменяются все этапы жизненного цикла программного обеспечения (подробнее об этом будет сказано ниже) информационной системы, при этом наибольшие изменения касаются этапов анализа и проектирования. Большинство существующих CASE-средств основано на методологиях структурного (в основном) или объектно-ориентированного анализа и проектирования, использующих спецификации в виде диаграмм или текстов для описания внешних требований, связей между моделями системы, динамики поведения системы и архитектуры программных средств. Такие методологии обеспечивают строгое и наглядное описание проектируемой системы, которое начинается с ее общего обзора и затем детализируется, приобретая иерархическую структуру со все большим числом уровней (уместна аналогия с традиционной разработкой сначала обобщенных, а затем — детальных алгоритмов программ).

CASE-технологии успешно применяются для построения практически всех типов систем ПО, однако устойчивое положение они занимают в следующих областях:

• обеспечение разработки делового и коммерческого ПО, где широкое применение CASE-технологий обусловлено массовостью этой прикладной области, в которой CASE применяется не только для разработки ПО, но и для создания моделей систем, помогающих решать задачи стратегического планирования, управления финансам и определения политики фирм, обучения персонала и др. (это направление получило свое собственное название — бизнес-анализ);

• разработка системного и управляющего ПО, где активное применение CASE-технологий связано с большой сложностью данной проблематики и со стремлением повысить эффективность работ.

CASE — не революция в программотехнике, а результат естественного эволюционного развития всей отрасли средств, называемых ранее инструментальными или технологическими. С самого начала CASE-технологии развивались с целью преодоления ограничений при использовании структурных методологий проектирования 60–70-х гг. прошлого века (сложности понимания, большой трудоемкости и стоимости использования, трудности внесения изменений в проектные спецификации и т. д.) за счет их автоматизации и интеграции поддерживающих средств. Таким образом, CASE-технологии не могут считаться самостоятельными методологиями, они только развивают структурные методологии и делают более эффективным их применение за счет автоматизации.

Помимо автоматизации структурных методологий и, как следствие, возможности применения современных методов системной и программной инженерии, CASE-средства обладают *следующими основными достоинствами:*

• улучшают качество создаваемого ПО за счет средств автоматического контроля (прежде всего, контроля проекта);

• позволяют за короткое время создавать прототип будущей системы, что позволяет на начальных этапах работ оценить ожидаемый результат;

• ускоряют процесс проектирования и разработки;

• освобождают разработчика от рутинной работы, позволяя ему целиком сосредоточиться на творческой части разработки;

• поддерживают развитие и сопровождение разработки;

• поддерживают технологии повторного использования компонент разработки.

Появлению CASE-технологии и CASE-средств предшествовали исследования в области методологии программирования. Программирование обрело черты системного подхода с разработкой и внедрением языков высокого уровня, методов структурного и модульного программирования, языков проектирования и средств их поддержки, формальных и неформальных языков описаний системных требований и спецификаций и т. д. В 70-х и 80-х гг. XX в. стала на практике применяться структурная методология, предоставляющая в распоряжение разработчиков строгие формализованные методы описания АИС и принимаемых технических решений. Она основана на наглядной графической технике: для описания различного рода моделей АИС используются схемы и диаграммы. Наглядность и строгость средств структурного анализа позволяла разработчикам и будущим пользователям системы с самого начала неформально участвовать в ее создании, обсуждать и закреплять понимание основных технических решений. Однако широкое применение этой методологии и следование ее рекомендациям при разработке конкретных АИС встречалось достаточно редко, поскольку при неавтоматизированной (ручной) разработке это практически невозможно. Это и способствовало появлению программно-технологических средств особого класса — CASE-средств, реализующих CASE-технологию создания и сопровождения АИС.

Необходимо отметить, что успешное применение CASE-средств невозможно без понимания базовой технологии, на которой эти средства основаны. Сами по себе программные CASE-средства являются *средствами автоматизации* процессов проектирования и сопровождения информационных систем. Без понимания методологии проектирования ИС невозможно применение CASE-средств.

8.3.1. Жизненный цикл программного обеспечения информационной системы

Одним из базовых понятий методологии проектирования АИС является понятие *жизненного цикла* ее программного обеспечения (ЖЦ ПО). ЖЦ ПО — это непрерывный процесс, который начинается с момента принятия решения о необходимости его создания и заканчивается в момент его полного изъятия из эксплуатации [6].

Структура ЖЦ ПО базируется на трех группах процессов:

• *основные* процессы ЖЦ ПО (приобретение, поставка, разработка, эксплуатация, сопровождение);

• *вспомогательные* процессы, обеспечивающие выполнение основных процессов (документирование, управление конфигурацией, обеспечение качества, верификация, аттестация, оценка, аудит, решение проблем);

• *организационные* процессы (управление проектами, создание инфраструктуры проекта, определение, оценка и улучшение самого ЖЦ, обучение).

Разработка включает в себя все работы по созданию ПО и его компонент в соответствии с заданными требованиями, включая оформление проектной и эксплуатационной документации, подготовку материалов, необходимых для проверки работоспособности и соответствующего качества программных продуктов, материалов, необходимых для организации обучения персонала и т. д. Разработка ПО включает в себя, как правило, анализ, проектирование и реализацию (программирование).

Эксплуатация включает в себя работы по внедрению компонентов ПО в эксплуатацию, в том числе конфигурирование баз данных и рабочих мест пользователей, обеспечение эксплуатационной документацией, проведение обучения персонала и т. д., и непосредственно эксплуатацию, в том числе локализацию проблем и устранение причин их возникновения, модификацию ПО в рамках установленного регламента, подготовку предложений по совершенствованию, развитию и модернизации системы.

Управление проектом связано с вопросами планирования и организации работ, создания коллективов разработчиков и конт-

роля за сроками и качеством выполняемых работ. Техническое и организационное обеспечение проекта включает выбор методов и инструментальных средств для реализации проекта, определение методов описания промежуточных состояний разработки, разработку методов и средств испытаний ПО, обучение персонала и т. п. Обеспечение качества проекта связано с проблемами верификации, проверки и тестирования ПО. *Верификация* — это процесс определения того, отвечает ли текущее состояние разработки, достигнутое на данном этапе, требованиям этого этапа. *Проверка* позволяет оценить соответствие параметров разработки с исходными требованиями. Проверка частично совпадает с *тестированием,* которое связано с идентификацией различий между действительными и ожидаемыми результатами и оценкой соответствия характеристик ПО исходным требованиям. В процессе реализации проекта важное место занимают вопросы идентификации, описания и контроля конфигурации отдельных компонентов и всей системы в целом.

Управление конфигурацией является одним из вспомогательных процессов, поддерживающих основные процессы жизненного цикла ПО, прежде всего процессы разработки и сопровождения ПО. При создании проектов сложных ИС, состоящих из многих компонентов, каждый из которых может иметь разновидности или версии, возникает проблема учета их связей и функций, создания унифицированной структуры и обеспечения развития всей системы. Управление конфигурацией позволяет организовать, систематически учитывать и контролировать внесение изменений в ПО на всех стадиях ЖЦ. Общие принципы и рекомендации конфигурационного учета, планирования и управления конфигурациями ПО отражены в стандарте ISO 12207-2.

Каждый процесс характеризуется определенными задачами и методами их решения, исходными данными, полученными на предыдущем этапе, и результатами. Результатами анализа, в частности, являются функциональные модели, информационные модели и соответствующие им диаграммы. ЖЦ ПО носит итерационный характер: результаты очередного этапа часто вызывают изменения в проектных решениях, выработанных на более ранних этапах.

Существующие модели ЖЦ определяют порядок исполнения этапов в ходе разработки, а также критерии перехода от этапа к этапу. В соответствии с этим наибольшее распространение получили три следующие модели ЖЦ:

• *каскадная модель* (1970–1980 гг.) — предполагает переход на следующий этап после полного окончания работ по предыдущему этапу;

• *поэтапная модель с промежуточным контролем* (1980–1985 гг.) — итерационная модель разработки по с циклами обратной связи между этапами. Преимущество такой модели заключается в том, что межэтапные корректировки обеспечивают меньшую трудоемкость по сравнению с каскадной моделью однако, время жизни каждого из этапов растягивается на весь период разработки;

• *спиральная модель* (1986–1990 гг.) — делает упор на начальные этапы ЖЦ: анализ требований, проектирование спецификаций, предварительное и детальное проектирование. На этих этапах проверяется и обосновывается реализуемость технических решений путем создания прототипов. Каждый виток спирали соответствует поэтапной модели создания фрагмента или версии программного изделия, на нем уточняются цели и характеристики проекта, определяется его качество, планируются работы следующего витка спирали. Таким образом углубляются и последовательно конкретизируются детали проекта и в результате выбирается обоснованный вариант, который доводится до реализации.

Специалистами отмечаются следующие *преимущества* спиральной модели:

• накопление и повторное использование программных средств, моделей и прототипов;

• ориентация на развитие и модификацию ПО в процессе его проектирования;

• анализ рисков и издержек в процессе проектирования.

Главная особенность индустрии создания ПО состоит в концентрации сложности на начальных этапах ЖЦ (анализ, проектирование) при относительно невысокой сложности и трудоемкости последующих этапов. Более того, нерешенные вопросы и ошибки, допущенные на этапах анализа и проектирования, порождают

на последующих этапах трудные, часто неразрешимые проблемы, и в конечном счете приводят к неуспеху всего проекта.

8.3.2. RAD-технологии создания приложений

Одним из возможных подходов к разработке ПО в рамках спиральной модели ЖЦ является получившая в последнее время широкое распространение методология быстрой разработки приложений RAD (Rapid Application Development). Под этим термином обычно понимается процесс разработки ПО, содержащий три элемента:

- небольшую команду разработчиков (от 2 до 10 чел.);
- короткий, но тщательно проработанный производственный график (от 2 до 6 мес.);
- повторяющийся цикл, при котором разработчики по мере того, как приложение начинает обретать форму, запрашивают и реализуют в продукте требования, полученные от заказчика.

Команда разработчиков должна представлять из себя группу профессионалов, имеющих опыт в анализе, проектировании, генерации кода и тестировании ПО с использованием CASE-средств. Члены коллектива должны также уметь трансформировать в рабочие прототипы предложения конечных пользователей.

Жизненный цикл ПО по методологии RAD состоит из четырех фаз:

- *анализ и планирование требований*;
- *проектирование;*
- *построение;*
- *внедрение.*

На фазе *анализа и планирования* требований пользователи системы определяют функции, которые она должна выполнять, выделяют наиболее приоритетные из них, требующие проработки в первую очередь, описывают информационные потребности. Определение требований выполняется в основном силами пользователей под руководством специалистов-разработчиков. Ограничивается масштаб проекта, определяются временные рамки для каждой из последующих фаз. Кроме того, определяется сама возможность реализации данного проекта в установленных рамках фи-

нансирования, на данных аппаратных средствах и т. п. Результатом данной фазы должны быть список и приоритетность функций будущей АИС, предварительные функциональные и информационные модели ИС.

На фазе *проектирования* часть пользователей принимает участие в техническом проектировании системы под руководством специалистов-разработчиков. CASE-средства используются для быстрого получения работающих прототипов приложений. Пользователи, непосредственно взаимодействуя с ними, уточняют и дополняют требования к системе, которые не были выявлены на предыдущей фазе. Более подробно рассматриваются процессы системы. Анализируется и, при необходимости, корректируется функциональная модель. Каждый процесс рассматривается детально. При необходимости для каждого элементарного процесса создается частичный прототип: экран, диалог, отчет, устраняющий неясности или неоднозначности. Определяются требования разграничения доступа к данным. На этой же фазе происходит определение набора необходимой документации.

После детального определения состава процессов оценивается количество функциональных элементов разрабатываемой системы и принимается решение о разделении АИС на подсистемы, поддающиеся реализации одной командой разработчиков за приемлемое для RAD-проектов время — порядка 60–90 дней. С использованием CASE-средств проект распределяется между различными исполнителями (делятся работы по созданию функциональной модели). Результатом данной фазы должны быть:

- общая информационная модель системы;
- функциональные модели системы в целом и подсистем, реализуемых отдельными командами разработчиков;
- точно определенные с помощью CASE-средств интерфейсы между автономно разрабатываемыми подсистемами;
- построенные прототипы экранов, отчетов, диалогов.

Все модели и прототипы должны быть получены с применением тех CASE-средств, которые будут использоваться в дальнейшем при построении системы. Данное требование вызвано тем, что в традиционном подходе при передаче информации о проекте

с этапа на этап может произойти фактически неконтролируемое искажение данных. Применение единой среды хранения информации о проекте позволяет избежать этой опасности.

В отличие от традиционного подхода, при котором использовались специфические средства прототипирования, не предназначенные для построения реальных приложений, а прототипы выбрасывались после того, как выполняли задачу устранения неясностей в проекте, в подходе RAD каждый прототип развивается в часть будущей системы. Таким образом, на следующую фазу передается более полная и полезная информация.

На фазе *построения* выполняется непосредственно сама быстрая разработка приложения. На данной фазе разработчики производят итеративное построение реальной системы на основе полученных в предыдущей фазе моделей, а также требований нефункционального характера. Программный код частично формируется при помощи автоматических генераторов, получающих информацию непосредственно из репозитория (англ. repository — хранилище, склад) CASE-средств. Конечные пользователи на этой фазе оценивают получаемые результаты и вносят коррективы, если в процессе разработки система перестает удовлетворять определенным ранее требованиям. Тестирование системы осуществляется непосредственно в процессе разработки.

После окончания работ каждой отдельной команды разработчиков производится постепенная интеграция данной части системы с остальными, формируется полный программный код, выполняется тестирование совместной работы данной части приложения с остальными, а затем тестирование системы в целом. Завершается физическое проектирование системы следующим образом:

- определяется необходимость распределения данных;
- производится анализ использования данных;
- производится физическое проектирование базы данных;
- определяются требования к аппаратным ресурсам;
- определяются способы увеличения производительности;
- завершается разработка документации проекта.

Результатом фазы является готовая система, удовлетворяющая всем согласованным требованиям.

На фазе *внедрения* производится обучение пользователей, организационные изменения и параллельно с внедрением новой системы осуществляется работа с существующей системой (до полного внедрения новой). Так как фаза построения достаточно непродолжительна, планирование и подготовка к внедрению должны начинаться заранее, как правило, на этапе проектирования системы. Приведенная схема разработки АИС не является абсолютной. Возможны различные варианты, зависящие, например, от начальных условий, в которых ведется разработка: разрабатывается ли совершенно новая система; было ли проведено информационное обследование организации и существует ли информационная модель ее деятельности; существует ли в организации некоторая АИС, которая может быть использована в качестве начального прототипа или должна быть интегрирована с разрабатываемой и т. п.

Следует, однако, отметить, что методология RAD, как и любая другая, не может претендовать на универсальность, она хороша в первую очередь для относительно небольших проектов, разрабатываемых для конкретного заказчика. Если же разрабатывается типовая система, которая не является законченным продуктом, а представляет собой комплекс типовых компонент, централизованно сопровождаемых, адаптируемых к программно-техническим платформам, СУБД, средствам телекоммуникации, организационно-экономическим особенностям объектов внедрения и интегрируемых с существующими разработками, на первый план выступают такие показатели проекта, как управляемость и качество, которые могут войти в противоречие с простотой и скоростью разработки. Для таких проектов необходимы высокий уровень планирования и жесткая дисциплина проектирования, строгое следование заранее разработанным протоколам и интерфейсам, что снижает скорость разработки.

Методология RAD неприменима для построения сложных расчетных программ, операционных систем или программ управ-

ления космическими кораблями, т. е. программ, требующих написания большого объема (сотни тысяч строк) уникального кода.

Не подходят для разработки по методологии RAD приложения, в которых отсутствует ярко выраженная интерфейсная часть, наглядно определяющая логику работы системы (например, приложения реального времени) и приложения, от которых зависит безопасность людей (например, управление самолетом или атомной электростанцией), так как итеративный подход предполагает, что первые несколько версий наверняка не будут полностью работоспособны, что в данном случае исключает основные принципы методологии RAD:

- разработка приложений итерациями;
- необязательность полного завершения работ на каждом из этапов жизненного цикла;
- обязательное вовлечение пользователей в процесс разработки АИС;
- необходимое применение CASE-средств, обеспечивающих целостность проекта;
- применение средств управления конфигурацией, облегчающих внесение изменений в проект и сопровождение готовой системы;
- необходимое использование генераторов кода;
- использование прототипирования, позволяющее полнее выяснить и удовлетворить потребности конечного пользователя;
- тестирование и развитие проекта, осуществляемые одновременно с разработкой;
- ведение разработки немногочисленной хорошо управляемой командой профессионалов;
- грамотное руководство разработкой системы, четкое планирование и контроль выполнения работ.

8.3.3. Структурный метод разработки программного обеспечения

Сущность структурного подхода к разработке АИС заключается в ее *декомпозиции* (разбиении) на автоматизируемые функ-

ции: система разбивается на функциональные подсистемы, которые в свою очередь делятся на подфункции, подразделяемые на задачи и так далее. Процесс разбиения продолжается вплоть до конкретных процедур. При этом автоматизируемая система сохраняет целостное представление, в котором все составляющие компоненты взаимоувязаны. При разработке системы "снизу-вверх" от отдельных задач ко всей системе целостность теряется, возникают проблемы при информационной стыковке отдельных компонентов.

Все методологии структурного анализа базируются на ряде общих принципов, часть из которых регламентирует организацию работ на начальных этапах ЖЦ, а часть используется при выработке рекомендаций по организации работ [6]. В качестве двух базовых принципов используются следующие: принцип "*разделяй и властвуй"* и принцип *иерархического упорядочивания.* Первый является принципом решения трудных проблем путем разбиения их на множество меньших независимых задач, более легких для понимания и решения. Второй принцип декларирует, что устройство этих частей также существенно для понимания. Уровень уяснения проблемы резко повышается при представлении ее частей в виде древовидных иерархических структур, т. е. система может быть понята и построена по уровням, каждый из которых добавляет новые детали.

Выделение двух базовых принципов инженерии программного обеспечения не означает, что остальные принципы являются второстепенными, игнорирование любого из них может привести к непредсказуемым последствиям (в том числе и к неуспеху всего проекта). Отметим основные из таких принципов.

1. Принцип *абстрагирования* — заключается в выделении существенных с некоторых позиций аспектов системы и отвлечение от несущественных с целью представления проблемы в простом общем виде.

2. Принцип *формализации* — заключается в необходимости следованию строгому методическому подходу к решению проблемы.

3. Принцип *упрятывания* — заключается в сокрытии несущественной на конкретном этапе информации: каждая часть "знает" только необходимую ей информацию.

4. Принцип *концептуальной общности* — заключается в следовании единой философии на всех этапах ЖЦ (структурный анализ — структурное проектирование — структурное программирование — структурное тестирование).

5. Принцип *полноты* — заключается в контроле присутствия лишних элементов.

6. Принцип *непротиворечивости* — заключается в обоснованности и согласованности элементов.

7. Принцип *логической независимости* — заключается в концентрации внимания на логическом проектировании для обеспечения независимости от физического проектирования.

8. Принцип *независимости данных* — заключается в том, что модели данных должны быть проанализированы и спроектированы независимо от процессов их логической обработки, а также от их физической структуры и распределения.

9. Принцип *структурирования данных* — заключается в том, что данные должны быть структурированы и иерархически организованы.

10. Принцип *доступа конечного пользователя* — заключается в том, что пользователь должен иметь средства доступа к базе данных, которые он может использовать непосредственно (без программирования).

Соблюдение указанных принципов необходимо при организации работ на начальных этапах ЖЦ независимо от типа разрабатываемого ПО и используемых при этом методологий. Руководствуясь всеми принципами в комплексе, можно на более ранних стадиях разработки понять, что будет представлять собой создаваемая система, обнаружить промахи и недоработки, что в свою очередь облегчит работы на последующих этапах ЖЦ и понизит стоимость разработки.

В структурном анализе используются в основном две группы средств, иллюстрирующих функции, выполняемые системой, и отношения между данными. Каждой группе средств соответствуют определенные виды моделей (диаграмм), наиболее распространенными среди которых являются следующие:

• SADT (Structured Analysis and Design Technique) — техника (технология) структурного анализа и проектирования, использующая соответствующие функциональные диаграммы;

• DFD (Data Flow Diagrams) — диаграммы потоков данных;

• ERD (Entity-Relationship Diagrams) — диаграммы "сущность-связь";

• STD (State Transition Diagrams) — диаграммы переходов состояний.

На стадии проектирования АИС модели расширяются, уточняются и дополняются диаграммами, отражающими структуру программного обеспечения: архитектуру ПО, структурные схемы программ и диаграммы экранных форм.

Перечисленные модели в совокупности дают полное описание АИС независимо от того, является ли она существующей или вновь разрабатываемой. Состав диаграмм в каждом конкретном случае зависит от необходимой полноты описания системы.

Методология SADT

Методология SADT разработана Дугласом Россом. На ее основе разработана, в частности, известная методология IDEF0 (Icam DEFinition), которая является основной частью программы ICAM (Интеграция компьютерных и промышленных технологий), проводимой по инициативе США. Методология SADT представляет собой совокупность методов, правил и процедур, предназначенных для построения функциональной модели объекта какой-либо предметной области. Функциональная модель SADT отображает функциональную структуру объекта, т. е. производимые им действия и связи между этими действиями. Основные элементы этой методологии основываются на следующих концепциях:

• графическое представление блочного моделирования. Графика блоков и дуг SADT-диаграммы отображает функцию в виде блока, а интерфейсы входа/выхода представляются дугами, соответственно входящими в блок и выходящими из него. Взаимодействие блоков друг с другом описывается посредством интерфейсных дуг, выражающих "ограничения", которые в свою очередь

определяют, когда и каким образом функции выполняются и управляются;

• строгость и точность. Выполнение правил SADT требует достаточной строгости и точности, не накладывая в то же время чрезмерных ограничений на действия аналитика. Правила SADT включают:

• ограничение количества блоков на каждом уровне декомпозиции (правило 3–6 блоков);

• связность диаграмм (номера блоков);

• уникальность меток и наименований (отсутствие повторяющихся имен);

• синтаксические правила для графики (блоков и дуг);

• разделение входов и управлений (правило определения роли данных).

• отделение организации от функции, т. е. исключение влияния организационной структуры на функциональную модель.

Методология SADT может использоваться для моделирования широкого круга систем и определения требований и функций, а затем для разработки системы, которая удовлетворяет этим требованиям и реализует эти функции. Для уже существующих систем SADT может быть использована для анализа функций, выполняемых системой, а также для указания механизмов, посредством которых они осуществляются.

Результатом применения методологии SADT является *модель, которая состоит из диаграмм, фрагментов текстов и глоссария, имеющих ссылки друг на друга.* Диаграммы — главные компоненты модели, все функции АИС и интерфейсы на них представлены как блоки и дуги. Место соединения дуги с блоком определяет тип интерфейса. Управляющая информация входит в блок сверху, в то время как информация, которая подвергается обработке, показана с левой стороны блока, а результаты выхода показаны с правой стороны. Механизм (человек или автоматизированная система), который осуществляет операцию, представляется дугой, входящей в блок снизу (рис. 8.3.1).

Одной из наиболее важных особенностей методологии SADT является постепенное введение все больших уровней детализации по мере создания диаграмм, отображающих модель.

Рис. 8.3.1. Функциональный блок и интерфейсные дуги

Построение SADT-модели начинается с представления всей системы в виде простейшей компоненты — одного блока и дуг, изображающих интерфейсы с функциями вне системы. Поскольку единственный блок представляет всю систему как единое целое, имя, указанное в блоке, является общим. Это верно и для интерфейсных дуг — они также представляют полный набор внешних интерфейсов системы в целом.

Затем блок, который представляет систему в качестве единого модуля, детализируется на другой диаграмме с помощью нескольких блоков, соединенных интерфейсными дугами. Эти блоки представляют основные подфункции исходной функции. Данная декомпозиция выявляет полный набор подфункций, каждая из которых представлена как блок, границы которого определены интерфейсными дугами. Каждая из этих подфункций может быть декомпозирована подобным образом для более детального представления.

Во всех случаях каждая подфункция может содержать только те элементы, которые входят в исходную функцию. Кроме того, модель не может опустить какие-либо элементы, т. е., как уже отмечалось, так называемый родительский блок и его интерфейсы обеспечивают контекст. К нему нельзя ничего добавить, и из него не может быть ничего удалено.

Модель SADT представляет собой серию диаграмм с сопроводительной документацией, разбивающих сложный объект на составные части, которые представлены в виде блоков. Детали каждого из основных блоков показаны в виде блоков на других

диаграммах. На каждом шаге декомпозиции более общая диаграмма называется родительской для более детальной диаграммы (рис. 8.3.2 и 8.3.3).

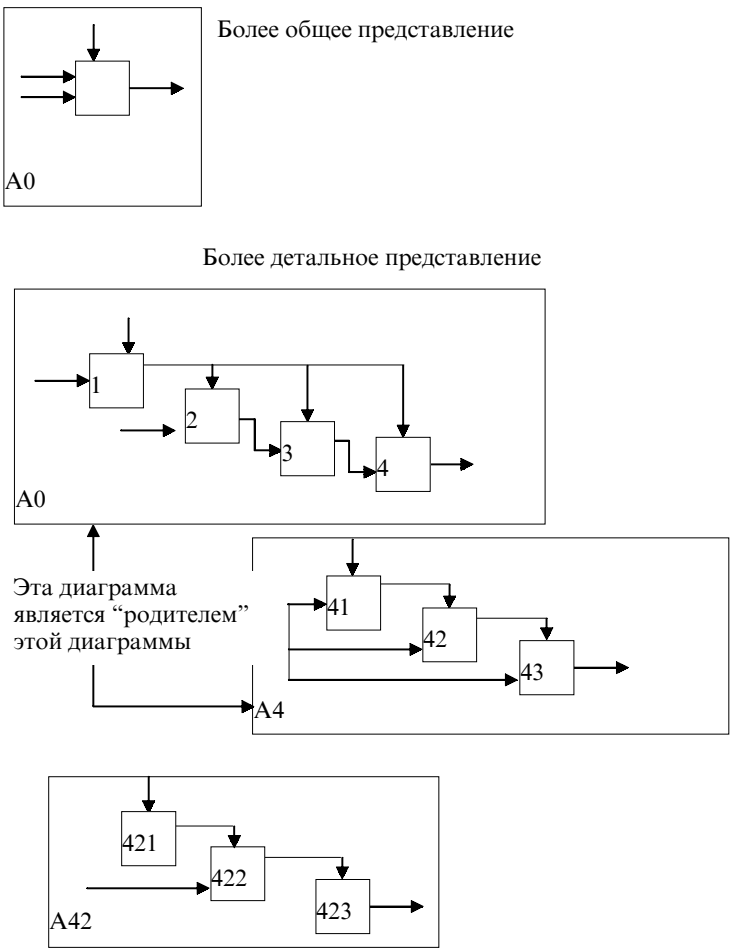
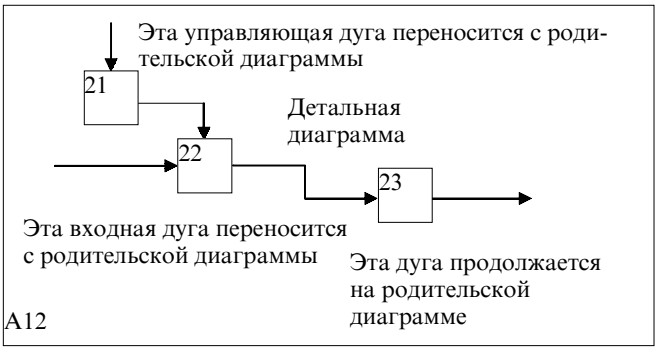


Рис. 8.3.2. Структура SADT-модели (декомпозиция диаграмм)



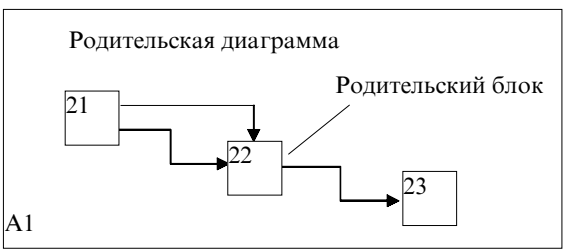


Рис. 8.3.3. Иерархия диаграмм

Дуги, входящие в блок и выходящие из него на диаграмме верхнего уровня, являются теми же самыми, что и дуги, входящие в диаграмму нижнего уровня и выходящие из нее, потому что блок и диаграмма представляют одну и ту же часть системы.

Некоторые дуги присоединены к блокам диаграммы обоими концами, у других же один конец остается неприсоединенным. Неприсоединенные дуги соответствуют входам, управлениям и выходам родительского блока. Источник или получатель этих пограничных дуг может быть обнаружен только на родительской диаграмме. Неприсоединенные концы должны соответствовать дугам на исходной диаграмме. Все граничные дуги должны продолжаться на родительской диаграмме, чтобы она была полной и непротиворечивой.

На SADT-диаграммах не указаны явно ни последовательность, ни время. Обратные связи, итерации, продолжающиеся процессы и перекрывающиеся (по времени) функции могут быть изображены с помощью дуг. Обратные связи могут выступать в виде комментариев, замечаний, исправлений и т. д.

Как было отмечено, механизмы (дуги с нижней стороны) показывают средства, с помощью которых осуществляется выполнение функций. Механизм может быть человеком, компьютером или любым другим устройством, которое помогает выполнять данную функцию.

Каждый блок на диаграмме имеет свой номер. Блок любой диаграммы может быть далее описан диаграммой нижнего уровня, которая, в свою очередь, может быть далее детализирована с помощью необходимого числа диаграмм. Таким образом формируется иерархия диаграмм.

Для того чтобы указать положение любой диаграммы или блока в иерархии, используются номера диаграмм. Например, A21 является диаграммой, которая детализирует блок 1 на диаграмме A2. Аналогично, A2 детализирует блок 2 на диаграмме A0, которая является самой верхней диаграммой модели.

Одним из важных моментов при проектировании ИС с помощью методологии SADT является точная согласованность типов связей между функциями. Различают по крайней мере семь типов связывания:

Тип связи	Относительная значимость
Случайная	0
Логическая	1
Временная	2
Процедурная	3
Коммуникационная	4
Последовательная	5
Функциональная	6

Ниже каждый тип связи кратко определен и проиллюстрирован с помощью типичного примера из SADT (табл. 8.3.1).

Таблица 8.3.1

Типы связности для функций и данных

Значимость	Тип связности	Для функций	Для данных
0	Случайная	Случайная	Случайная
1	Логическая	Функции одного и того же множества или типа (например, “редактировать все входы”)	Данные одного и того же множества или типа
2	Временная	Функции одного и того же периода времени (например, “операции инициализации”)	Данные, используемые в каком-либо временном интервале
3	Процедурная	Функции, работающие в одной и той же фазе или итерации (например, “первый проход компилятора”)	Данные, используемые во время одной и той же фазы или итерации
4	Коммуникационная	Функции, использующие одни и те же данные	Данные, на которые воздействует одна и та же деятельность
5	Последовательная	Функции, выполняющие последовательные преобразования одних и тех же данных	Данные, преобразуемые последовательными функциями
6	Функциональная	Функции, объединяемые для выполнения одной функции	Данные, связанные с одной функцией

(0) — тип *случайной* связности: наименее желательный. Случайная связность возникает, когда конкретная связь между функциями мала или полностью отсутствует. Это относится к ситуации, когда имена данных на SADT-дугах в одной диаграмме имеют малую связь друг с другом.

(1) — тип *логической* связности. Логическое связывание происходит тогда, когда данные и функции собираются вместе вследствие того, что они попадают в общий класс или набор элементов, но необходимых функциональных отношений между ними не обнаруживается.

(2) — тип *временной* связности. Связанные по времени элементы возникают вследствие того, что они представляют функции, связанные во времени, когда данные используются одновременно или функции включаются параллельно, а не последовательно.

(3) — тип *процедурной* связности. Процедурно-связанные элементы появляются сгруппированными вместе вследствие того, что они выполняются в течение одной и той же части цикла или процесса.

(4) — тип *коммуникационной* связности. Диаграммы демонстрируют коммуникационные связи, когда блоки группируются вследствие того, что они используют одни и те же входные данные и/или производят одни и те же выходные данные.

(5) — тип *последовательной* связности. На диаграммах, имеющих последовательные связи, выход одной функции служит входными данными для следующей функции. Связь между элементами на диаграмме является более тесной, чем на рассмотренных выше уровнях связок, поскольку моделируются причинно-следственные зависимости .

(6) — тип *функциональной* связности. Диаграмма отражает полную функциональную связность при наличии полной зависимости одной функции от другой. Диаграмма, которая является чисто функциональной, не содержит чужеродных элементов, относящихся к последовательному или более слабому типу связности. Одним из способов определения функционально-связанных диаграмм является рассмотрение двух блоков, связанных через управляющие дуги.

Моделирование потоков данных (процессов)

Основным средством моделирования функциональных требований АИС являются диаграммы потоков данных (DFD). С их помощью эти требования разбиваются на функциональные компо-

ненты (процессы) и представляются в виде сети, связанной потоками данных. Главная цель таких средств — продемонстрировать, как каждый процесс преобразует свои входные данные в выходные, а также выявить отношения между этими процессами.

Для изображения DFD традиционно используются две различные *нотации*: Йодана (Yourdon) и Гейна-Сарсона (Gane-Sarson) — рис. 8.3.4.

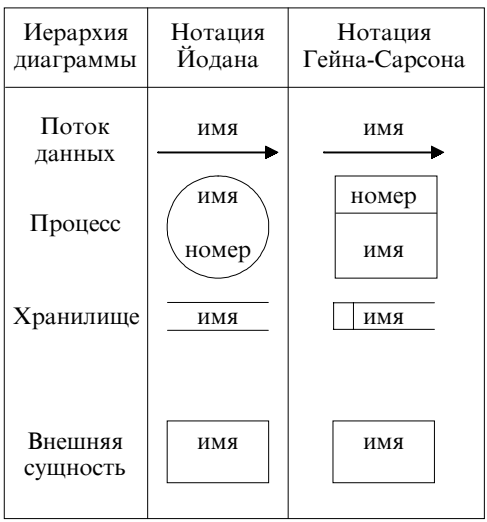


Рис. 8.3.4. Представление нотаций Йодана и Гейна-Сарсона

В соответствии с методологией модель системы определяется как иерархия диаграмм потоков данных (ДПД или DFD), описывающих асинхронный процесс преобразования информации от ее ввода в систему до выдачи пользователю. Диаграммы верхних уровней иерархии (контекстные диаграммы) определяют основные процессы или подсистемы ИС с внешними входами и выходами. Они детализируются при помощи диаграмм нижнего уровня. Такая декомпозиция продолжается, создавая многоуровневую иерархию диаграмм, до тех пор, пока не будет достигнут такой уровень декомпозиции, на котором процессы становятся элементарными и детализировать их далее невозможно.

Источники информации (внешние сущности) порождают информационные потоки (потоки данных), переносящие информацию к подсистемам или процессам. Те в свою очередь преобразуют информацию и порождают новые потоки, которые переносят информацию к другим процессам или подсистемам, накопителям данных или внешним сущностям — потребителям информации. Таким образом, основными компонентами диаграмм потоков данных являются:

- внешние сущности;
- системы/подсистемы;
- процессы;
- накопители данных;
- потоки данных.

Внешняя сущность представляет собой материальный предмет или физическое лицо, представляющее собой источник или приемник информации, например, заказчики, персонал, поставщики, клиенты, склад. Определение некоторого объекта или системы в качестве внешней сущности указывает на то, что она находится за пределами границ анализируемой АИС.

Процесс представляет собой преобразование входных потоков данных в выходные в соответствии с определенным алгоритмом. Физически процесс может быть реализован различными способами: это может быть подразделение организации (отдел), выполняющее обработку входных документов и выпуск отчетов, программа, аппаратно реализованное логическое устройство и т. д. В различных нотациях процесс может изображаться на диаграммах по-разному. Номер процесса служит для его идентификации. В поле имени вводится наименование процесса в виде предложения с активным недвусмысленным глаголом в неопределенной форме (вычислить, рассчитать, проверить, определить, создать, получить), за которым следуют существительные в винительном падеже, например:

- “Ввести сведения о клиентах”;
- “Выдать информацию о текущих расходах”;
- “Проверить кредитоспособность клиента”.

Использование таких глаголов, как “обработать”, “модернизировать” или “отредактировать” означает, как правило, недостаточно глубокое понимание данного процесса и требует дальнейшего анализа.

В последнее время принято использовать еще и поле физической реализации, информация в котором показывает, какое подразделение организации, программа или аппаратное устройство выполняет данный процесс.

Хранилище (накопитель данных) представляет собой абстрактное устройство для хранения информации, которую можно в любой момент поместить в накопитель и через некоторое время извлечь, причем способы помещения и извлечения могут быть любыми.

Накопитель данных может быть реализован физически в виде микрофиши, ящика в картотеке, таблицы в оперативной памяти, файла на магнитном носителе и т. д. Накопитель данных идентифицируется буквой “D” и произвольным числом. Имя накопителя выбирается из соображения наибольшей информативности для проектировщика.

Накопитель данных в общем случае является прообразом будущей базы данных, и описание хранящихся в нем данных должно быть увязано с информационной моделью. Поток данных определяет информацию, передаваемую через некоторое соединение от источника к приемнику. Реальный поток данных может быть информацией, передаваемой по кабелю между двумя устройствами, пересылаемыми по почте письмами, магнитными лентами или дискетами, переносимыми с одного компьютера на другой и т. д.

Поток данных на диаграмме изображается линией, оканчивающейся стрелкой, которая показывает направление. Каждый поток данных имеет имя, отражающее его содержание.

Первым шагом при построении иерархии DFD является построение контекстных диаграмм. Обычно при проектировании относительно простых АИС строится единственная контекстная диаграмма со звездообразной топологией, в центре которой находится так называемый главный процесс, соединенный с приемниками и источниками информации, посредством которых с систе-

мой взаимодействуют пользователи и другие внешние системы. Для сложных АИС строится иерархия контекстных диаграмм. При этом контекстная диаграмма верхнего уровня содержит не единственный главный процесс, а набор подсистем, соединенных потоками данных. Контекстные диаграммы следующего уровня детализируют контекст и структуру подсистем.

Иерархия контекстных диаграмм определяет взаимодействие основных функциональных подсистем проектируемой АИС как между собой, так и с внешними входными и выходными потоками данных и внешними объектами (источниками и приемниками информации), с которыми взаимодействует АИС.

Разработка контекстных диаграмм решает проблему строгого определения функциональной структуры АИС на самой ранней стадии ее проектирования, что особенно важно для сложных многофункциональных систем, в разработке которых участвуют разные организации и коллективы разработчиков.

После построения контекстных диаграмм полученную модель следует проверить на полноту исходных данных об объектах системы и изолированность объектов (отсутствие информационных связей с другими объектами). Для каждой подсистемы, присутствующей на контекстных диаграммах, выполняется ее детализация при помощи DFD. Каждый процесс на DFD, в свою очередь, может быть детализирован при помощи DFD или миниспецификации. При детализации должны выполняться следующие *правила*:

• правило *балансировки* — означает, что при детализации подсистемы или процесса детализирующая диаграмма в качестве внешних источников/приемников данных может иметь только те компоненты (подсистемы, процессы, внешние сущности, накопители данных), с которыми имеет информационную связь детализируемая подсистема или процесс на родительской диаграмме;

• правило *нумерации* — означает, что при детализации процессов должна поддерживаться их иерархическая нумерация. Например, процессы, детализирующие процесс с номером 12, получают номера 12.1, 12.2, 12.3 и т. д.

Мини-спецификация (описание логики процесса) должна формулировать его основные функции таким образом, чтобы в даль-

нейшем специалист, выполняющий реализацию проекта, смог выполнить их или разработать соответствующую программу.

Мини-спецификация является конечной вершиной иерархии DFD. Решение о завершении детализации процесса и использовании мини-спецификации принимается аналитиком, исходя из следующих критериев:

- наличия у процесса относительно небольшого количества входных и выходных потоков данных (2–3 потока);
- возможности описания преобразования данных процесса в виде последовательного алгоритма;
- выполнения процессом единственной логической функции преобразования входной информации в выходную;
- возможности описания логики процесса при помощи мини-спецификации небольшого объема (не более 20–30 строк).

При построении иерархии DFD переходить к детализации процессов следует только после определения содержания всех потоков и накопителей данных, которое описывается при помощи структур данных. Структуры данных конструируются из элементов данных и могут содержать альтернативы, условные вхождения и итерации. Условное вхождение означает, что данный компонент может отсутствовать в структуре. Альтернатива означает, что в структуру может входить один из перечисленных элементов. Итерация означает вхождение любого числа элементов в указанном диапазоне. Для каждого элемента данных может указываться его тип (непрерывные или дискретные данные). Для непрерывных данных может указываться единица измерения (кг, см и т. п.), диапазон значений, точность представления и форма физического кодирования. Для дискретных данных может указываться таблица допустимых значений.

После построения законченной модели системы ее необходимо *верифицировать* (проверить на полноту и согласованность). В полной модели все ее объекты (подсистемы, процессы, потоки данных) должны быть подробно описаны и детализированы. Выявленные недетализированные объекты следует детализировать, вернувшись на предыдущие шаги разработки. В согласованной модели для всех потоков данных и накопителей данных должно вы-

полняться правило сохранения информации: все поступающие куда-либо данные должны быть считаны, а все считываемые данные должны быть записаны.

Моделирование данных

Цель моделирования данных состоит в обеспечении разработчика АИС *концептуальной схемой базы данных* в форме одной модели или нескольких локальных моделей, которые относительно легко могут быть отображены в любую систему баз данных.

Наиболее распространенным средством моделирования данных являются диаграммы "сущность-связь" (ERD). С их помощью определяются важные для предметной области объекты (сущности), их свойства (атрибуты) и отношения друг с другом (связи). ERD непосредственно используются для проектирования реляционных баз данных (см. п. 9.1.4).

Нотация ERD была впервые введена П. Ченом (P. Chen) и получила дальнейшее развитие в работах Баркера.

Методология IDEF1

Метод IDEF1, разработанный Т. Рэмеем (T. Ramey), также основан на подходе П. Чена и позволяет построить модель данных, эквивалентную реляционной модели в третьей нормальной форме (см. п. 9.2.5). В настоящее время на основе совершенствования методологии IDEF1 создана ее новая версия — методология IDEF1X, которая разработана с учетом таких требований, как простота изучения и возможность автоматизации. IDEF1X-диаграммы используются рядом распространенных CASE-средств (в частности, ERwin, Design/IDEF).

8.3.4. Методологии проектирования программного обеспечения

Современные методологии и реализующие их технологии поставляются в электронном виде вместе с CASE-средствами и включают библиотеки процессов, шаблонов, методов, моделей и других компонент, предназначенных для построения ПО того класса систем, на который ориентирована методология. Электронные

методологии включают также средства, которые должны обеспечивать их адаптацию для конкретных пользователей и развитие методологии по результатам выполнения конкретных проектов.

Процесс адаптации заключается в удалении ненужных процессов, действий ЖЦ и других компонентов методологии, в изменении неподходящих или в добавлении собственных процессов и действий, а также методов, моделей, стандартов и руководств. Настройка методологии может осуществляться также по следующим аспектам: этапы и операции ЖЦ, участники проекта, используемые модели ЖЦ, поддерживаемые концепции и др.

Электронные методологии и технологии (и поддерживающие их CASE-средства) составляют ядро комплекса согласованных инструментальных средств среды разработки АИС.

Методология DATARUN

Одной из наиболее распространенных в мире электронных методологий является методология DATARUN. В соответствии с ней ЖЦ ПО разбивается на стадии, которые связываются с результатами выполнения основных процессов, определяемых стандартом ISO 12207. Каждую стадию помимо ее результатов должен завершать план работ на следующую стадию.

Стадия *формирования требований* и планирования включает в себя действия по определению начальных оценок объема и стоимости проекта. Должны быть сформулированы требования и экономическое обоснование для разработки АИС, функциональные модели (модели бизнес-процессов организации) и исходная концептуальная модель данных, которые дают основу для оценки технической реализуемости проекта. Основными результатами этой стадии должны быть модели деятельности организации (исходные модели процессов и данных организации), требования к системе, включая требования по сопряжению с существующими АИС, исходный бизнес-план.

Стадия *концептуального* проектирования начинается с детального анализа первичных данных и уточнения концептуальной модели данных, после чего проектируется архитектура системы. Архитектура включает в себя разделение концептуальной модели

на обозримые подмодели. Оценивается возможность использования существующих АИС и выбирается соответствующий метод их преобразования. После построения проекта уточняется исходный бизнес-план. Выходными компонентами этой стадии являются концептуальная модель данных, модель архитектуры системы и уточненный бизнес-план.

На стадии *спецификации* приложений продолжается процесс создания и детализации проекта. Концептуальная модель данных преобразуется в реляционную модель данных. Определяется структура приложения, необходимые интерфейсы приложения в виде экранов, отчетов и пакетных процессов вместе с логикой их вызова. Модель данных уточняется бизнес-правилами и методами для каждой таблицы. В конце этой стадии принимается окончательное решение о способе реализации приложений. По результатам стадии должен быть построен проект АИС, включающий модели архитектуры АИС, данных, функций, интерфейсов (с внешними системами и с пользователями), требований к разрабатываемым приложениям (модели данных, интерфейсов и функций), требований к доработкам существующих АИС, требований к интеграции приложений, а также сформирован окончательный план создания АИС.

На стадии *разработки, интеграции и тестирования* должна быть создана тестовая база данных, частные и комплексные тесты. Проводится разработка, прототипирование и тестирование баз данных и приложений в соответствии с проектом. Отлаживаются интерфейсы с существующими системами. Описывается конфигурация текущей версии ПО. На основе результатов тестирования проводится оптимизация базы данных и приложений. Приложения интегрируются в систему, проводится тестирование приложений в составе системы и испытания системы. Основными результатами стадии являются готовые приложения, проверенные в составе системы на комплексных тестах, текущее описание конфигурации ПО, скорректированная по результатам испытаний версия системы и эксплуатационная документация на систему.

Стадия *внедрения* включает в себя действия по установке и внедрению баз данных и приложений. Основными результатами

стадии должны быть готовая к эксплуатации и перенесенная на программно-аппаратную платформу заказчика версия системы, документация сопровождения и акт приемочных испытаний по результатам опытной эксплуатации.

Стадии *сопровождения и развития* включают процессы и операции, связанные с регистрацией, диагностикой и локализацией ошибок, внесением изменений и тестированием, проведением доработок, тиражированием и распространением новых версий ПО в места его эксплуатации, переносом приложений на новую платформу и масштабированием системы. Стадия развития фактически является повторной итерацией стадии разработки.

Методология DATARUN опирается на две модели, или на два представления:

- модель организации;
- модель АИС.

Методология DATARUN базируется на системном подходе к описанию деятельности организации. Построение моделей начинается с описания процессов, из которых затем извлекаются первичные данные (стабильное подмножество данных, которые организация должна использовать для своей деятельности). Первичные данные описывают продукты или услуги организации, выполняемые операции (транзакции) и потребляемые ресурсы. К первичным относятся данные, которые описывают внешние и внутренние сущности, такие как служащие, клиенты или агентства, а также данные, полученные в результате принятия решений, как, например, графики работ, цены на продукты.

Основной принцип DATARUN заключается в том, что первичные данные, если они должным образом организованы в модель данных, становятся основой для проектирования архитектуры АИС. Архитектура АИС будет более стабильной, если она основана на первичных данных, тесно связанных с основными деловыми операциями, определяющими природу бизнеса, а не на традиционной функциональной модели.

Любая АИС представляет собой набор модулей, исполняемых процессорами и взаимодействующих с базами данных. Базы данных и процессоры могут располагаться централизованно или

быть распределенными. События в системе могут инициироваться внешними сущностями. Все транзакции осуществляются через объекты или модули интерфейса, которые взаимодействуют с одной или более базами данных.

Подход DATARUN преследует *две цели*:

• определить *стабильную структуру*, на основе которой будет строиться АИС. Такой структурой является модель данных, полученная из первичных данных, представляющих фундаментальные процессы организации;

• *спроектировать* АИС на основании модели данных.

Объекты, формируемые на основании модели данных, являются объектами базы данных, обычно размещаемыми на серверах в среде клиент/сервер. Объекты интерфейса, определенные в архитектуре компьютерной системы, обычно размещаются на клиентской части. Модель данных, являющаяся основой для спецификации совместно используемых объектов базы данных и различных объектов интерфейса, обеспечивает сопровождаемость АИС.

В процессе разработки АИС создается ряд моделей:

• BPM (Business Process Model) — модель процессов (бизнес-процессов);

• PDS (Primary Data Structure) — структура первичных данных;

• CDM (Conceptual Data Model) — концептуальная модель данных;

• SPM (System Process Model) — модель процессов системы;

• ISA (Information System Architecture) — архитектура информационной системы;

• ADM (Application Data Model) — модель данных приложения;

• IPM (Interface Presentation Model) — модель представления интерфейса;

• ISM (Interface Specification Model) — модель спецификации интерфейса.

CASE-средство *Silverrun* обеспечивает автоматизацию проведения проектных работ в соответствии с методологией DATARUN, а также создание этих моделей. Предоставляемая эти-

ми средствами среда проектирования дает возможность руководителю проекта контролировать проведение работ, отслеживать их выполнение, вовремя замечать отклонения от графика. Каждый участник проекта, подключившись к этой среде, может выяснить содержание и сроки выполнения порученной ему работы, детально изучить технику ее выполнения в гипертексте по технологиям и вызвать инструмент (модуль Silverrun) для реального выполнения работы.

Информационная система создается последовательным построением ряда моделей, начиная с модели бизнес-процессов и заканчивая моделью программы, автоматизирующей эти процессы.

Создаваемая АИС должна основываться на функциях, выполняемых организацией. Поэтому первая создаваемая модель — это модель бизнес-процессов, построение которой осуществляется в модуле Silverrun BPM. Для этой модели используется специальная нотация BPM. В процессе анализа и спецификации бизнес-функций выявляются основные информационные объекты, которые документируются как структуры данных, связанные с потоками и хранилищами модели. Источниками для создания структур являются используемые в организации документы, должностные инструкции, описания производственных операций. Эти данные вводятся в том виде, как они существуют в деятельности организации. Нормализация и удаление избыточности производится позже при построении концептуальной модели данных в модуле Silverrun ERX. После создания модели бизнес-процессов информация сохраняется в репозитории проекта.

В процессе обследования работы организации выявляются и документируются структуры первичных данных. Эти структуры заносятся в репозиторий модуля BPM при описании циркулирующих в организации документов, сообщений, данных. В модели бизнес-процессов первичные структуры данных связаны с потоками и хранилищами информации.

На основе структур первичных данных в модуле Silverrun ERX создается концептуальная модель данных (ER-модель). От структур первичных данных концептуальная модель отличается удалением избыточности, стандартизацией наименований понятий и

нормализацией. Эти операции в модуле ERX выполняются при помощи встроенной экспертной системы. Цель концептуальной модели данных — описать используемую информацию без деталей возможной реализации в базе данных, но в хорошо структурированном нормализованном виде.

На основе модели бизнес-процессов и концептуальной модели данных проектируется архитектура АИС. Определяются входящие в систему приложения, для каждого приложения специфицируются используемые данные и реализуемые функции. Архитектура АИС создается в модуле Silverrun BPM с использованием специальной нотации ISA. Основное содержание этой модели — структурные компоненты системы и навигация между ними. Концептуальная модель данных разбивается на части, соответствующие входящим в состав системы приложениям.

Перед разработкой приложений должна быть спроектирована структура корпоративной базы данных. DATARUN предполагает использование базы данных, основанной на реляционной модели. Концептуальная модель данных после нормализации переносится в модуль реляционного моделирования Silverrun RDM с помощью специального моста ERX-RDM. Преобразование модели из формата ERX в формат RDM происходит автоматически без вмешательства пользователя. После преобразования форматов получается модель реляционной базы данных. Эта модель детализируется в модуле Silverrun RDM определением физической реализации (типов данных СУБД, ключей, индексов, триггеров, ограничений целостности). Правила обработки данных можно задавать как непосредственно на языке программирования СУБД, так и в декларативной форме, не привязанной к реализации. Мосты Silverrun к реляционным СУБД переводят эти декларативные правила на язык требуемой системы, что снижает трудоемкость программирования процедур сервера базы данных, а также позволяет из одной спецификации генерировать приложения для разных СУБД.

С помощью модели системных процессов детально документируется поведение каждого приложения. В модуле BPM создается модель системных процессов, определяющая, каким образом

реализуются бизнес-процессы. Эта модель создается отдельно для каждого приложения и тесно связана с моделью данных приложения.

Приложение состоит из интерфейсных объектов (экранных форм, отчетов, процедур обработки данных). Каждый интерфейс системы (экранная форма, отчет, процедура обработки данных) имеет дело с подмножеством базы данных. В модели данных приложения (созданной в модуле RDM) создается подсхема базы данных для каждого интерфейса этого приложения. Уточняются также правила обработки данных, специфичные для каждого интерфейса.

Модель представления интерфейса — это описание внешнего вида интерфейса с точки зрения конечного пользователя системы. Это может быть документ, показывающий внешний вид экрана или структуру отчета, или экран (отчет), созданный с помощью одного из средств визуальной разработки приложений — так называемых *языков четвертого поколения* (4GL — Fourth Generation Languages). Так как большинство языков 4GL позволяют быстро создавать работающие прототипы приложений, пользователь имеет возможность увидеть работающий прототип системы на ранних стадиях проектирования.

После создания подсхем реляционной модели для приложений проектируется детальная структура каждого приложения в виде схемы навигации экранов, отчетов, процедур пакетной обработки. На данном шаге эта структура детализируется до указания конкретных столбцов и таблиц базы данных, правил их обработки, вида экранных форм и отчетов. Полученная модель детально документирует приложение и непосредственно используется для программирования специфицированных интерфейсов.

Далее с помощью средств разработки приложений происходит физическое создание системы: приложения программируются и интегрируются в информационную систему.

Характеристика современных CASE-средств

Современные CASE-средства охватывают обширную область поддержки многочисленных технологий проектирования ИС: от

простых средств анализа и документирования до полномасштабных средств автоматизации, покрывающих весь жизненный цикл ПО [14].

Наиболее трудоемкими этапами разработки АИС являются этапы анализа и проектирования, в процессе которых CASE-средства обеспечивают качество принимаемых технических решений и подготовку проектной документации. При этом большую роль играют методы визуального представления информации. Это предполагает построение структурных или иных диаграмм в реальном масштабе времени, использование многообразной цветовой палитры, сквозную проверку синтаксических правил. Графические средства моделирования предметной области позволяют разработчикам в наглядном виде изучать существующую АИС, перестраивать ее в соответствии с поставленными целями и имеющимися ограничениями.

В разряд CASE-средств попадают как относительно дешевые системы для персональных компьютеров с весьма ограниченными возможностями, так и дорогостоящие системы для неоднородных вычислительных платформ и операционных сред. Так, современный рынок программных средств насчитывает около 300 различных CASE-средств, наиболее мощные из которых так или иначе используются практически всеми ведущими западными фирмами.

Обычно к CASE-средствам относят любое программное средство, автоматизирующее ту или иную совокупность процессов жизненного цикла ПО и обладающее следующими основными характерными *особенностями*:

- мощными графическими средствами для описания и документирования АИС, обеспечивающими удобный интерфейс с разработчиком и развивающие его творческие возможности;
- интеграцией отдельных компонент CASE-средств, обеспечивающей управляемость процессом разработки АИС;
- использованием специальным образом организованного хранилища проектных метаданных (репозитория).

Интегрированное CASE-средство (или комплекс средств, поддерживающих полный ЖЦ ПО) содержит следующие компоненты:

• репозиторий, являющийся основой CASE-средства. Он должен обеспечивать хранение версий проекта и его отдельных компонентов, синхронизацию поступления информации от различных разработчиков при групповой разработке, контроль метаданных на полноту и непротиворечивость;

• графические средства анализа и проектирования, обеспечивающие создание и редактирование иерархически связанных диаграмм (DFD, ERD и др.), образующих модели ИС;

• средства разработки приложений, включая языки 4GL и генераторы кодов;

• средства конфигурационного управления;

• средства документирования;

• средства тестирования;

• средства управления проектом;

• средства реинжиниринга.

Все современные CASE-средства могут быть классифицированы в основном по типам и категориям. Классификация по типам отражает функциональную ориентацию CASE-средств на те или иные процессы ЖЦ. Классификация по категориям определяет степень интегрированности по выполняемым функциям и включает отдельные локальные средства, решающие небольшие автономные задачи (tools), набор частично интегрированных средств, охватывающих большинство этапов жизненного цикла АИС (toolkit) и полностью интегрированные средства, поддерживающие весь ЖЦ ИС и связанные общим репозиторием. Помимо этого, CASE-средства можно классифицировать по следующим признакам:

• применяемым методологиям и моделям систем и БД;

• степени интегрированности с СУБД;

• доступным платформам.

Классификация по типам в основном совпадает с компонентным составом CASE-средств и включает следующие основные типы (после названия средства в скобках указана фирма-разработчик):

• *средства анализа* (Upper CASE), предназначенные для построения и анализа моделей предметной области (Design/IDEF (Meta Software), BPwin (Logic Works));

• *средства анализа и проектирования* (Middle CASE), поддерживающие наиболее распространенные методологии проектирования и использующиеся для создания проектных спецификаций (Vantage Team Builder (Cayenne), Designer/2000 (ORACLE), Silverrun (CSA), PRO-IV (McDonnell Douglas), CASE.Аналитик (МакроПроджект)). Выходом таких средств являются спецификации компонентов и интерфейсов системы, архитектуры системы, алгоритмов и структур данных;

• *средства проектирования баз данных*, обеспечивающие моделирование данных и генерацию схем баз данных (как правило, на языке SQL) для наиболее распространенных СУБД. К ним относятся ERwin (Logic Works), S-Designor (SDP) и DataBase Designer (ORACLE). Средства проектирования баз данных имеются также в составе CASE-средств Vantage Team Builder, Designer/2000, Silverrun и PRO-IV;

• средства разработки приложений. К ним относятся средства 4GL (Uniface (Compuware), JAM (JYACC), PowerBuilder (Sybase), Developer/2000 (ORACLE), New Era (Informix), SQL Windows (Gupta), Delphi (Borland) и др.) и генераторы кодов, входящие в состав Vantage Team Builder, PRO-IV и частично — в Silverrun;

• *средства реинжиниринга*, обеспечивающие анализ программных кодов и схем баз данных и формирование на их основе различных моделей и проектных спецификаций. Средства анализа схем БД и формирования ERD входят в состав Vantage Team Builder, PRO-IV, Silverrun, Designer/2000, ERwin и S-Designor. В области анализа программных кодов наибольшее распространение получают объектно-ориентированные CASE-средства, обеспечивающие реинжиниринг программ на языке C ++ (Rational Rose (Rational Software), Object Team (Cayenne)).

Вспомогательные типы включают:

• средства планирования и управления проектом (SE Companion, Microsoft Project и др.);

• средства конфигурационного управления (PVCS (Intersolv));

• средства тестирования (Quality Works (Segue Software));

• средства документирования (SoDA (Rational Software)).

На сегодняшний день российский рынок программного обеспечения располагает следующими наиболее развитыми CASE-средствами:

- Silverrun;
- Designer/2000;
- Vantage Team Builder (Westmount I-CASE);
- ERwin + BPwin;
- S-Designer;
- CASE.Аналитик.

Кроме того, на рынке постоянно появляются как новые для отечественных пользователей системы (например, CASE/4/0, PRO-IV, System Architect, Visible Analyst Workbench, EasyCASE), так и новые версии и модификации перечисленных систем.

Охарактеризуем основные возможности CASE-средств на примере имеющей широкое распространение системы Silverrun.

CASE-средство Silverrun американской фирмы Computer Systems Advisers, Inc. (CSA) используется для анализа и проектирования АИС бизнес-класса и ориентировано в большей степени на спиральную модель ЖЦ. Оно применимо для поддержки любой методологии, основанной на раздельном построении функциональной и информационной моделей (диаграмм потоков данных и диаграмм "сущность-связь").

Настройка на конкретную методологию обеспечивается выбором требуемой графической нотации моделей и набора правил проверки проектных спецификаций. В системе имеются готовые настройки для наиболее распространенных методологий: DATARUN (основная методология, поддерживаемая Silverrun), Gane/Sarson, Yourdon/DeMarco, Merise, Ward/Mellor, Information Engineering. Для каждого понятия, введенного в проекте, имеется возможность добавления собственных описателей. Архитектура Silverrun позволяет наращивать среду разработки по мере необходимости.

Silverrun имеет модульную структуру и состоит из четырех модулей, каждый из которых является самостоятельным продуктом и может приобретаться и использоваться без связи с остальными *модулями.*

Модуль построения моделей бизнес-процессов в форме диаграмм потоков данных (BPM — Business Process Modeler) позволяет мо-

делировать функционирование обследуемой организации или создаваемой АИС. В модуле BPM обеспечена возможность работы с моделями большой сложности: автоматическая перенумерация, работа с деревом процессов (включая визуальное перетаскивание ветвей), отсоединение и присоединение частей модели для коллективной разработки. Диаграммы могут изображаться в нескольких предопределенных нотациях, включая Yourdon/DeMarco и Gane/Sarson. Имеется также возможность создавать собственные нотации, в том числе добавлять в число изображаемых на схеме дескрипторов определенные пользователем поля.

Модуль *концептуального моделирования* данных (ERX — Entity-Relationship eXpert) обеспечивает построение моделей данных "сущность-связь", не привязанных к конкретной реализации. Этот модуль имеет встроенную экспертную систему, позволяющую создать корректную нормализованную модель данных посредством ответов на содержательные вопросы о взаимосвязи данных. Возможно автоматическое построение модели данных из описаний структур данных. Анализ функциональных зависимостей атрибутов дает возможность проверить соответствие модели требованиям третьей нормальной формы и обеспечить их выполнение. Проверенная модель передается в модуль RDM.

Модуль *реляционного моделирования* (RDM — Relational Data Modeler) позволяет создавать детализированные модели "сущность-связь", предназначенные для реализации в реляционной базе данных. В этом модуле документируются все конструкции, связанные с построением базы данных: индексы, триггеры, хранимые процедуры и т. д. Гибкая изменяемая нотация и расширяемость репозитория позволяют работать по любой методологии. Возможность создавать подсхемы соответствует подходу ANSI SPARC к представлению схемы базы данных. На языке подсхем моделируются как узлы распределенной обработки, так и пользовательские представления. Этот модуль обеспечивает проектирование и полное документирование реляционных баз данных.

Менеджер репозитория рабочей группы (WRM — Workgroup Repository Manager) применяется как словарь данных для хране-

ния общей для всех моделей информации, а также обеспечивает интеграцию модулей Silverrun в единую среду проектирования.

Платой за высокую гибкость и разнообразие изобразительных средств построения моделей является такой недостаток Silverrun, как отсутствие жесткого взаимного контроля между компонентами различных моделей (например, возможности автоматического распространения изменений между DFD различных уровней декомпозиции). Следует, однако, отметить, что этот недостаток может иметь существенное значение только в случае использования каскадной модели ЖЦ ПО.

Для автоматической генерации схем баз данных у Silverrun существуют так называемые мосты к наиболее распространенным СУБД: Oracle, Informix, DB2, Ingres, Progress, SQL Server, SQLBase, Sybase. Для передачи данных в средства разработки приложений имеются мосты к языкам 4GL: JAM, PowerBuilder, SQL Windows, Uniface, NewEra, Delphi. Все мосты позволяют загрузить в Silverrun RDM информацию из каталогов соответствующих СУБД или языков 4GL. Это позволяет документировать, перепроектировать или переносить на новые платформы уже находящиеся в эксплуатации базы данных и прикладные системы. При использовании моста Silverrun расширяет свой внутренний репозиторий специфичными для целевой системы атрибутами. После определения значений этих атрибутов генератор приложений переносит их во внутренний каталог среды разработки или использует при генерации кода на языке SQL. Таким образом, можно полностью определить ядро базы данных с использованием всех возможностей конкретной СУБД: триггеров, хранимых процедур, ограничений ссылочной целостности. При создании приложения на языке 4GL данные, перенесенные из репозитория Silverrun, используются либо для автоматической генерации интерфейсных объектов, либо для быстрого их создания вручную.

Для обмена данными с другими средствами автоматизации проектирования, создания специализированных процедур анализа и проверки проектных спецификаций, составления специализированных отчетов в соответствии с различными стандартами в системе Silverrun имеется три способа выдачи проектной информации во внешние файлы:

• система отчетов. Можно, определив содержимое отчета по репозиторию, выдать отчет в текстовый файл. Этот файл можно затем загрузить в текстовый редактор или включить в другой отчет;

• система экспорта/импорта. Для более полного контроля над структурой файлов в системе экспорта/импорта имеется возможность определять не только содержимое экспортного файла, но и разделители записей, полей в записях, маркеры начала и конца текстовых полей. Файлы с указанной структурой можно не только формировать, но и загружать в репозиторий. Это дает возможность обмениваться данными с различными системами: другими CASE-средствами, СУБД, текстовыми редакторами и электронными таблицами;

• хранение репозитория во внешних файлах через ODBC-драйверы. Для доступа к данным репозитория из наиболее распространенных систем управления базами данных обеспечена возможность хранить всю проектную информацию непосредственно в формате этих СУБД.

Групповая работа поддерживается в системе Silverrun двумя способами:

• в стандартной однопользовательской версии имеется механизм контролируемого разделения и слияния моделей. Разделив модель на части, можно раздать их нескольким разработчикам. После детальной доработки модели объединяются в единые спецификации;

• сетевая версия Silverrun позволяет осуществлять одновременную групповую работу с моделями, хранящимися в сетевом репозитории на базе СУБД Oracle, Sybase или Informix. При этом несколько разработчиков могут работать с одной и той же моделью, так как блокировка объектов происходит на уровне отдельных элементов модели.

Имеются реализации Silverrun трех платформ — MS Windows, Macintosh и OS/2 Presentation Manager — с возможностью обмена проектными данными между ними.

Помимо системы Silverrun, укажем назначение и других популярных CASE-средств и их групп.

Vantage Team Builder представляет собой интегрированный программный продукт, ориентированный на реализацию каскадной модели ЖЦ ПО и поддержку полного ЖЦ ПО.

Uniface 6.1 — продукт фирмы Compuware (США) — представляет собой среду разработки крупномасштабных приложений в архитектуре "клиент-сервер".

CASE-средство Designer/2000 2.0 фирмы ORACLE является интегрированным CASE-средством, обеспечивающим в совокупности со средствами разработки приложений Developer/2000 поддержку полного ЖЦ ПО для систем, использующих СУБД ORACLE.

Пакет CASE/4/0 (microTOOL GmbH), включающий структурные средства системного анализа, проектирования и программирования, обеспечивает поддержку всего жизненного цикла разработки (вплоть до сопровождения), на основе сетевого репозитория, контролирующего целостность проекта и поддерживающего согласованную работу всех его участников (системных аналитиков, проектировщиков, программистов).

Локальные средства

Пакет ERWin (Logic Works) используется при моделировании и создании баз данных произвольной сложности на основе диаграмм "сущность-связь". В настоящее время ERWin является наиболее популярным пакетом моделирования данных благодаря поддержке широкого спектра СУБД самых различных классов — SQL-серверов (Oracle, Informix, Sybase SQL Server, MS SQL Server, Progress, DB2, SQLBase, Ingress, Rdb и др.) и "настольных" СУБД типа xBase (Clipper, dBASE, FoxPro, MS Access, Paradox и др.).

BPWin — средство функционального моделирования, реализующее методологию IDEF0. Модель в BPWin представляет собой совокупность SADT-диаграмм, каждая из которых описывает отдельный процесс, разбивая его на шаги и подпроцессы.

S-Designer 4.2 (Sybase/Powersoft) представляет собой CASE-средство для проектирования реляционных баз данных. По своим функциональным возможностям и стоимости он близок к CASE-средству ERWin, отличаясь внешне используемой на диаграммах нотацией. S-Designor реализует стандартную методологию моделирования данных и генерирует описание БД для таких СУБД, как ORACLE, Informix, Ingres, Sybase, DB/2, MS SQL Server и др.

CASE. Аналитик 1.1 (Эйтекс) является практически единственным в настоящее время конкурентоспособным отечественным CASE-средством функционального моделирования и реализует построение диаграмм потоков данных в соответствии с описанной ранее методологией.

Объектно-ориентированные CASE-средства

Rational Rose — CASE-средство фирмы Rational Software Corporation (США) — предназначено для автоматизации этапов анализа и проектирования ПО, а также для генерации кодов на различных языках и выпуска проектной документации. Rational Rose использует синтез-методологию объектно-ориентированного анализа и проектирования, основанную на подходах трех ведущих специалистов в данной области: Буча, Рамбо и Джекобсона. Разработанная ими универсальная нотация для моделирования объектов (язык UML — Unified Modeling Language — универсальный язык моделирования) является в настоящее время и, очевидно, останется в будущем общепринятым стандартом в области объектно-ориентированного анализа и проектирования. Конкретный вариант Rational Rose определяется языком, на котором генерируются коды программ (C++ , SmallTalk, PowerBuilder, Ada, SQLWindows и ObjectPro). Основной вариант — Rational Rose/C++ — позволяет разрабатывать проектную документацию в виде диаграмм и спецификаций, а также генерировать программные коды на C++ . Кроме того, Rational Rose содержит средства реинжиниринга программ, обеспечивающие повторное использование программных компонент в новых проектах.

Средства конфигурационного управления

Цель конфигурационного управления (КУ) — обеспечить управляемость и контролируемость процессов разработки и сопровождения ПО. Для этого необходима точная и достоверная информация о состоянии ПО и его компонент в каждый момент времени, а также о всех предполагаемых и выполненных изменениях.

Для решения задач КУ применяются методы и средства обеспечивающие идентификацию состояния компонент, учет номенк-

латуры всех компонент и модификаций системы в целом, контроль за вносимыми изменениями в компоненты, структуру системы и ее функции, а также координированное управление развитием функций и улучшением характеристик системы.

Наиболее распространенным средством КУ является PVCS фирмы Intersolv (США), включающее ряд самостоятельных продуктов: PVCS Version Manager, PVCS Tracker, PVCS Configuration Builder и PVCS Notify.

Средства документирования

Для создания документации в процессе разработки АИС используются разнообразные средства формирования отчетов, а также компоненты издательских систем. Обычно средства документирования встроены в конкретные CASE-средства. Исключением являются некоторые пакеты, предоставляющие дополнительный сервис при документировании. Из них наиболее активно используется SoDA (Software Document Automation).

Продукт SoDA предназначен для автоматизации разработки проектной документации на всех фазах ЖЦ ПО. Он позволяет автоматически извлекать разнообразную информацию, получаемую на разных стадиях разработки проекта, и включать ее в выходные документы. При этом контролируется соответствие документации проекту, взаимосвязь документов, обеспечивается их своевременное обновление. Результирующая документация автоматически формируется из множества источников, число которых не ограничено.

Пакет включает в себя графический редактор для подготовки шаблонов документов. Он позволяет задавать необходимый стиль, фон, шрифт, определять расположение заголовков, резервировать места, где будет размещаться извлекаемая из разнообразных источников информация. Изменения автоматически вносятся только в те части документации, на которые они повлияли в программе. Это сокращает время подготовки документации за счет отказа от перегенерации всей документации.

SoDA реализована на базе издательской системы FrameBuilder и предоставляет полный набор средств по редактированию и верстке выпускаемой документации.

Итоговым результатом работы системы SoDA является готовый документ (или книга). Документ может храниться в файле формата SoDA (Frame Builder), который получается в результате генерации документа. Вывод на печать этого документа (или его части) возможен из системы SoDA.

Среда функционирования SoDA — ОС типа UNIX на рабочих станциях Sun SPARCstation, IBM RISC System/6000 или Hewlett Packard HP 9000 700/800.

Средства тестирования

Под тестированием понимается процесс исполнения программы с целью обнаружения ошибок. Регрессионное тестирование — это тестирование, проводимое после усовершенствования функций программы или внесения в нее изменений.

Одно из наиболее развитых средств тестирования QA (новое название — Quality Works) представляет собой интегрированную, многоплатформенную среду для разработки автоматизированных тестов любого уровня, включая тесты регрессии для приложений с графическим интерфейсом пользователя.

QA позволяет начинать тестирование на любой фазе ЖЦ, планировать и управлять процессом тестирования, отображать изменения в приложении и повторно использовать тесты для более чем 25 различных платформ.

В заключение приведем пример комплекса CASE-средств, обеспечивающего поддержку полного ЖЦ ПО. Нецелесообразно сравнивать отдельно взятые CASE-средства, поскольку ни одно из них не решает в целом все проблемы создания и сопровождения ПО. Это подтверждается также полным набором критериев оценки и выбора, которые затрагивают все этапы ЖЦ ПО. Сравниваться могут комплексы методологически и технологически согласованных инструментальных средств, поддерживающие полный ЖЦ ПО и обеспеченные необходимой технической и методической поддержкой со стороны фирм-поставщиков (отметим, что рациональное комплексирование инструментальных средств разработки ПО АИС является важнейшим условием обеспечения ка-

чества этого ПО, причем это замечание справедливо для всех предметных областей). На сегодняшний день наиболее развитым из всех поставляемых в Россию комплексов такого рода является комплекс технологий и инструментальных средств создания АИС, основанный на методологии и технологии DATARUN. В состав комплекса входят следующие инструментальные средства:

- CASE-средство Silverrun;
- средство разработки приложений JAM;
- мост Silverrun-RDM <-> JAM;
- комплекс средств тестирования QA;
- менеджер транзакций Tuxedo;
- комплекс средств планирования и управления проектом SE Companion;
- комплекс средств конфигурационного управления PVCS;
- объектно-ориентированное CASE-средство Rational Rose;
- средство документирования SoDA.

Вопросы для самопроверки

1. Поясните сущность CASE-технологий.
2. Назовите основные этапы жизненного цикла програамного обеспечения (ЖЦ ПО).
3. На каких группах процессов базируется ЖЦ ПО?
4. Какие элементы используются в методологии RAD?
5. Что является результатом применения методологии SADT?
6. Изобразите основные нотации диаграмм потоков данных (DFD).
7. На какие модели опирается методология DATDRUN?
8. Какие компоненты входят в любое современное интегрированное CASE-средство?
9. В чем сущность объектно-ориентированного подхода к проектированию ПО АИС?
10. Приведите пример объектно-ориентированного CASE-средства.

9. Централизованная и распределенная обработка данных

9.1. Принципы построения и этапы проектирования базы данных

Автоматизированные информационно-справочные системы (АИСС) в настоящее время получили весьма широкое распространение, что связано прежде всего со сравнительной простотой их создания и исключительно высоким эффектом от внедрения. Методологической основой информационных технологий, реализуемых в АИСС, являются концепции *централизованной* (в рамках разработки баз и банков данных) и *распределенной* (в рамках создания информационных сетей, в том числе и с распределенными банками данных в узлах) обработки информации. Современные компьютерные сети будут достаточно подробно рассмотрены в п. 9.3. Настоящий и последующий пункты посвящены вопросам проектирования и использования баз и банков данных.

9.1.1. Основные понятия и определения

В науке одним из наиболее сложных для строгого определения является понятие "информация". Согласно кибернетическому подходу [15] "информация — первоначальное сообщение данных, сведений, осведомление и т. п.". Кибернетика вывела понятие информации за пределы человеческой речи и других форм коммуникации между людьми, *связала его с целенаправленными системами любой природы*. Информация выступает *в трех формах*:

- *биологической* (биотоки; связи в генетических механизмах);
- *машинной* (сигналы в электрических цепях);
- *социальной* (движение знаний в общественных системах).

Иными словами, “информация — связь в любых целенаправленных системах, определяющая их целостность, устойчивость, уровень функционирования” [49]. Содержание и особенности информации раскрываются указанием действий, в которых она участвует:

• *хранение* (на некотором *носителе* информации);

• *преобразование* (в соответствии с некоторым *алгоритмом*);

• *передача* (с помощью *передатчика* и *приемника* по некоторой *линии связи*).

В соответствии с этим же подходом — “*данные — факты и идеи*, представленные в *формализованном виде*, позволяющем *передавать* или *обрабатывать* их при помощи некоторого *процесса* и соответствующих *технических устройств*” [15].

Толковый словарь по информатике [49, 53] определяет понятия “информация” и “данные” несколько иначе:

“*информация* — 1) *совокупность знаний о фактических данных и связях между ними;* 2) в вычислительной технике — *содержание, присваиваемое данным* посредством соглашений, распространяющихся на эти данные; *данные*, подлежащие *вводу в ЭВМ*, *хранимые в ее памяти, обрабатываемые на ЭВМ и выдаваемые пользователям*”; “*данные — информация*, представленная в виде, *пригодном для обработки автоматическими средствами при возможном участии человека”.*

Как легко заметить, приведенные определения вынужденно используют такие сложно определяемые понятия, как *“факты”, “идеи”* и, особенно, “*знания”.*

В дальнейшем *под информацией будем понимать любые сведения о процессах и явлениях, которые в той или иной форме передаются между объектами материального мира (людьми; животными; растениями; автоматами и др.).*

Если рассмотреть некоторый *объект материального мира*, информация о котором представляет интерес, и *наблюдателя* (в роли которого и выступают АИС), *способного фиксировать* эту информацию в определенной, понятной другим форме, то говорят, что *в памяти (“сознании”) наблюдателя находятся данные*, описывающие состояние объекта. Таким образом, *данными будем*

называть формализованную информацию, пригодную для последующей обработки, хранения и передачи средствами автоматизации профессиональной деятельности.

Информацию в ЭВМ можно хранить в виде различных данных (числовых; текстовых; визуальных и т. п.). Более того, для описания одной и той же информации можно предложить различные варианты их состава и структуры. Иными словами, *правомерно говорить о моделировании в АИС информации о некотором множестве объектов материального мира совокупностью взаимосвязанных данных.*

Информационное обеспечение (information support) АИС — совокупность единой *системы классификации и кодирования* информации; унифицированных *систем документации* и *используемых массивов информации* [53, 54]. В дальнейшем нас будет интересовать именно последний аспект данного определения.

В этой связи *в качестве главных задач создания информационного обеспечения* АИС можно выделить:

• во-первых, *определение состава и структуры данных*, достаточно "хорошо" описывающих требуемую информацию;

• во-вторых, *обоснование способов хранения и переработки да*нных с использованием ЭВМ.

Процесс создания информационного обеспечения включает несколько этапов, рассмотрению которых посвящен п. 8.2.4. В данном пункте остановимся на понятиях и определениях, связанных с технологией банков данных.

Прежде чем определить понятие "банк данных", необходимо остановиться на другом ключевом понятии — "предметная область".

Под *предметной областью* (ПО) будем понимать *информацию об объектах, процессах и явлениях окружающего мира*, которая *с точки зрения потенциальных пользователей должна храниться и обрабатываться в информационной системе.* В этом определении особое внимание следует уделить важности роли потенциальных потребителей информационных ресурсов АИС. Именно этот аспект обусловливает и структуру, и основные задачи, и вообще целесообразность создания того или иного банка.

Банк данных (БнД) — *информационная система*, включающая в свой состав *комплекс специальных методов и средств для поддержания динамической информационной модели предметной области с целью обеспечения информационных потребностей пользователей* [15, 39]. Очевидно, что БнД может рассматриваться как специальная обеспечивающая подсистема в составе старшей по иерархии АИС.

Поддержание динамической модели ПО предусматривает не только *хранение* информации о ней и своевременное *внесение изменений* в соответствии с реальным состоянием объектов, но и обеспечение возможности учета *изменений состава* этих объектов (в том числе появление новых) и *связей* между ними (т. е. изменений самой структуры хранимой информации).

Обеспечение информационных потребностей (запросов) пользователей имеет два аспекта [45]:

- *определение границ* конкретной ПО и *разработка описания* соответствующей информационной *модели*;
- *разработка БнД*, ориентированного на *эффективное обслуживание запросов* различных категорий пользователей.

С точки зрения *целевой направленности профессиональной деятельности* принято выделять пять основных категорий пользователей [45]:

- аналитики;
- системные программисты;
- прикладные программисты;
- администраторы;
- конечные пользователи.

Кроме того, различают пользователей *постоянных и разовых*; пользователей-*людей* и пользователей-*задач*; пользователей с различным *уровнем компетентности (приоритетом)* и др., причем каждый класс пользователей предъявляет собственные специфические требования к своему обслуживанию (прежде всего — с точки зрения организации диалога “запрос — ответ”). Так, например, постоянные пользователи, как правило, обращаются в БнД с фиксированными по форме (типовыми) запросами; пользователи-задачи должны иметь возможность получать информацию из БнД

в согласованной форме в указанные области памяти; пользователи с низким приоритетом могут получать ограниченную часть информации и т. д. Наличие столь разнообразного состава потребителей информации потребовало включения в БнД специального элемента — *словаря данных*, о чем будет сказано ниже.

Уровень сложности и важности задач информационного обеспечения АИС в рамках рассматриваемой технологии определяет ряд *основных требований к БнД* [53]:

- *адекватность* информации состоянию предметной области;
- *быстродействие и производительность*;
- *простота и удобство* использования;
- *массовость* использования;
- *защита* информации;
- *возможность расширения* круга решаемых задач.

(Отметим, что все названные требования можно предъявить и к любому финансовому банку.)

По сравнению с традиционным обеспечением монопольными файлами каждого приложения централизованное управление данными в БнД имеет *ряд важных преимуществ*:

- *сокращение избыточности* хранимых данных;
- *устранение противоречивости* хранимых данных;
- *многоаспектное использование* данных (при однократном вводе);
- *комплексная оптимизация* (с точки зрения удовлетворения разнообразных, в том числе и противоречивых, требований “в целом”);
- обеспечение *возможности стандартизации*;
- обеспечение *возможности санкционированного доступа* к данным и др.

Все названные преимущества, по существу, связаны с такими основополагающими *принципами* концепции БнД, как *интеграция* данных, *централизация управления* ими и *обеспечение независимости прикладных программ* обработки данных и самих данных.

Структура типового БнД, удовлетворяющего предъявляемым требованиям, представлена на рис. 9.1.1, где:

• *ВС* — вычислительная система, включающая технические средства (*ТС*) и общее программное обеспечение (*ОПО*);

• *БД* — базы данных;

• *СУБД* — система управления БД;

• *АБД* — администратор баз данных, а также обслуживающий *персонал* и *словарь данных*.

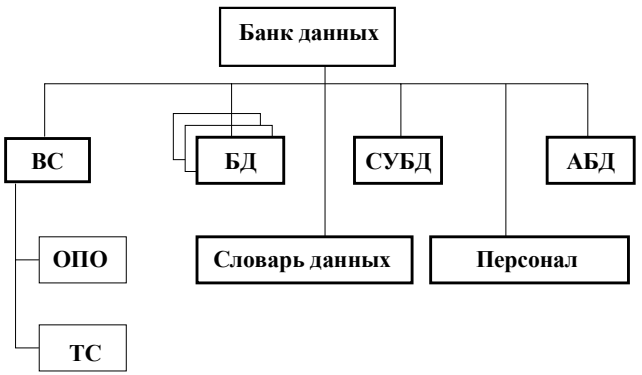


Рис. 9.1.1. Основные компоненты БнД

Подробнее остановимся на составляющих БнД, представляющих наибольший интерес. *БД — совокупность* специальным образом *организованных (структурированных) данных* и *связей* между ними. Иными словами, БД — это так называемое *датологическое* (от англ. data — данные) *представление информации о предметной области*. Если в состав БнД входит *одна БД*, банк принято называть *локальным*; если *БД несколько — интегрированным*.

СУБД — специальный комплекс программ и языков, посредством которого организуется централизованное управление базами данных и обеспечивается доступ к ним.

В состав *любой СУБД* входят *языки двух типов*:

• язык *описания данных* (с его помощью описываются *типы* данных, их *структура* и *связи*);

• язык *манипулирования данными* (его часто называют *язык запросов к БД*), предназначенный для организации работы с данными в интересах всех типов пользователей.

Словарь данных предназначен для хранения единообразной и централизованной *информации обо всех ресурсах данных* конкретного банка:

• об *объектах*, их *свойствах и отношениях* для данной ПО;

• о *данных*, хранимых в БД (наименование; смысловое описание; структура; связи и т. п.);

• о *возможных значениях* и *форматах* представления данных;

• об *источниках* возникновения данных;

• о *кодах защиты* и *разграничении доступа* пользователей к данным и т. п.

Администратор баз данных — это лицо (группа лиц), реализующее *управление БД*. В этой связи сам БнД можно рассматривать как *автоматизированную систему управления базами данных*. Функции АБД являются долгосрочными; он координирует все виды работ на этапах создания и применения БнД. На стадии проектирования АБД выступает как идеолог и главный конструктор системы; на стадии эксплуатации он отвечает за нормальное функционирование БнД, управляет режимом его работы и обеспечивает безопасность данных (последнее особенно важно при современном уровне развития средств коммуникации).

Основные функции АБД [15, 54]:

• решать вопросы организации данных об объектах ПО и установлении связей между этими данными с целью объединения информации о различных объектах; согласовывать представления пользователей;

• координировать все действия по проектированию, реализации и ведению БД; учитывать текущие и перспективные требования пользователей; следить, чтобы БД удовлетворяли актуальным потребностям;

• решать вопросы, связанные с расширением БД из-за изменения границ ПО;

• разрабатывать и реализовывать меры по обеспечению защиты данных от некомпетентного их использования, от сбоев технических средств, по обеспечению секретности определенной части данных и разграничению доступа к ним;

• выполнять работы по ведению словаря данных;

• контролировать избыточность и противоречивость данных, их достоверность;

• следить за тем, чтобы БнД отвечал заданным требованиям по производительности, т. е. чтобы обработка запросов выполнялась за приемлемое время;

• выполнять при необходимости изменения методов хранения данных, путей доступа к ним, связей между данными, их форматов; определять степень влияния изменений в данных на всю БД;

• координировать вопросы технического обеспечения системы аппаратными средствами, исходя из требований, предъявляемых БД к оборудованию;

• координировать работы системных программистов, разрабатывающих дополнительное программное обеспечение для улучшения эксплуатационных характеристик системы;

• координировать работы прикладных программистов, разрабатывающих новые прикладные программы, и выполнять их проверку и включение в состав ПО системы и т. п.

На рис. 9.1.2 представлен типовой состав группы АБД, отражающий основные направления деятельности специалистов.

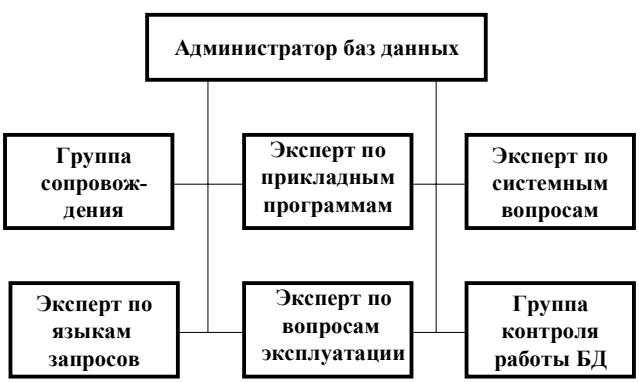


Рис. 9.1.2. Типовой состав группы АБД

9.1.2. Описательная модель предметной области

Процесс проектирования БД является весьма сложным. По сути, он заключается в определении перечня данных, хранимых на физических носителях (магнитных дисках и лентах), которые достаточно полно отражают информационные потребности потенциальных пользователей в конкретной ПО. Проектирование БД начинается с анализа предметной области и возможных запросов пользователей. В результате этого анализа определяется перечень данных и связей между ними, которые адекватно — с точки зрения будущих потребителей — отражают ПО. Завершается проектирование БД определением форм и способов хранения необходимых данных на физическом уровне.

Весь процесс проектирования БД можно разбить на ряд взаимосвязанных этапов, каждый из которых обладает своими особенностями и методами проведения. На рис. 9.1.3 представлены типовые этапы.

На этапе *инфологического (информационно-логического)* проектирования осуществляется построение семантической модели, описывающей сведения из предметной области, которые могут

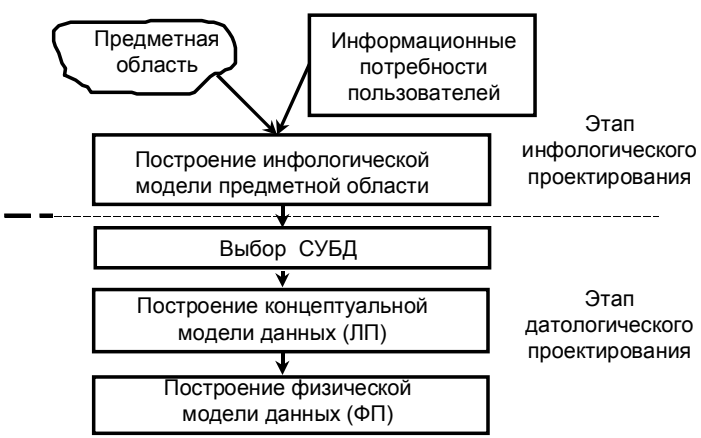


Рис. 9.1.3. Этапы проектирования БД

заинтересовать пользователей БД. Семантическая модель (semantic model) — представление совокупности о ПО понятий ... в виде *графа*, в вершинах которого расположены *понятия*, в терминальных вершинах — элементарные понятия, а *дуги* представляют *отношения между понятиями*.

Сначала из объективной реальности выделяется ПО, т. е. очерчиваются ее границы. Логический анализ выделенной ПО и потенциальных запросов пользователей завершается построением инфологической модели ПО — перечня сведений об объектах ПО, которые необходимо хранить в БД, и связях между ними.

Анализ информационных потребностей потенциальных пользователей имеет два аспекта:

- во-первых, *определение* собственно *сведений об объектах* ПО;
- во-вторых, *анализ возможных запросов к БД* и требований по оперативности их выполнения.

Анализ возможных запросов к БД позволяет уточнить связи между сведениями, которые необходимо хранить. Пусть, например, в БД по учебному процессу института хранятся сведения об учебных группах, читаемых курсах и кафедрах, а также связи "учебные группы — читаемые курсы" и "читаемые курсы — кафедры". Тогда запрос о том, проводит ли некоторая кафедра занятия в конкретной учебной группе, может быть выполнен только путем перебора всех читаемых в данной группе курсов.

Хранение большого числа связей усложняет БД и приводит к увеличению потребной памяти ЭВМ, но часто существенно ускоряет поиск нужной информации. Поэтому разработчику БД (АБД) приходится принимать компромиссное решение, причем процесс определения перечня хранимых связей, как правило, имеет итерационный характер.

Датологическое проектирование подразделяется на *логическое* (построение концептуальной модели данных) и *физическое* (построение физической модели) проектирование.

Главной задачей логического проектирования (ЛП) БД является представление выделенных на предыдущем этапе сведений в виде данных в форматах, поддерживаемых выбранной СУБД.

Задача физического проектирования (ФП) — выбор способа хранения данных на физических носителях и методов доступа к ним с использованием возможностей, предоставляемых СУБД.

Инфологическая модель "сущность-связь" (entity — relationship model; ER-model) П. Чена (P. Chen) представляет собой описательную (неформальную) модель ПО, семантически определяющую в ней сущности и связи [44].

Относительная простота и наглядность описания ПО позволяет использовать ее в процессе диалога с потенциальными пользователями с самого начала инфологического проектирования. Построение инфологической модели П. Чена, как и любой другой модели, является творческим процессом, поэтому единой методики ее создания нет. Однако при любом подходе к построению модели используют три основных конструктивных элемента:

- сущность;
- атрибут;
- связь.

Сущность — это собирательное понятие некоторого повторяющегося объекта, процесса или явления окружающего мира, о котором необходимо хранить информацию в системе. Сущность может определять как материальные (например, "студент", "грузовой автомобиль" и т. п.), так и нематериальные объекты (например, "экзамен", "проверка" и т. п.). Главной особенностью сущности является то, что вокруг нее сосредоточен сбор информации в конкретной ПО. *Тип сущности* определяет *набор однородных объектов*, а *экземпляр сущности — конкретный объект* в наборе. Каждая сущность в модели Чена именуется. Для идентификации конкретного экземпляра сущности и его описания используется один или несколько атрибутов.

Атрибут — это поименованная характеристика сущности, которая принимает значения из некоторого множества значений [46]. Например, у сущности "студент" могут быть атрибуты "фамилия", "имя", "отчество", "дата рождения", "средний балл за время обучения" и т. п.

Связи в инфологической модели выступают в качестве средства, с помощью которого представляются *отношения между сущностями*, имеющими место в ПО. При анализе связей между сущ-

ностями могут встречаться бинарные (между двумя сущностями) и, в общем случае, n-арные (между n сущностями) связи. Например, сущности “отец”, “мать” и “ребенок” могут находиться в трехарном отношении “семья” (“является членом семьи”).

Связи должны быть поименованы; между двумя типами сущностей могут существовать несколько типов связей.

Наиболее распространены бинарные связи. Учитывая, что любую n-арную связь можно представить в виде нескольких бинарных, подробнее остановимся именно на таких связях между двумя типами сущностей, устанавливающими соответствие между множествами экземпляров сущностей.

Различают четыре типа связей:

- связь один к одному (1:1);
- связь один ко многим (1:M);
- связь многие к одному (M:1);
- связь многие ко многим (M:N).

Связь *один к одному* определяет такой тип связи между типами сущностей A и B, при которой каждому экземпляру сущности A соответствует один и только один экземпляр сущности B, и наоборот. Таким образом, имея некоторый экземпляр сущности A, можно однозначно идентифицировать соответствующий ему экземпляр сущности B, а по экземпляру сущности B — экземпляр сущности A. Например, связь типа 1:1 “имеет” может быть определена между сущностями “автомобиль” и “двигатель”, так как на конкретном автомобиле может быть установлен только один двигатель, и этом двигатель, естественно, нельзя установить сразу на несколько автомобилей.

Связь *один ко многим* определяет такой тип связи между типами сущностей A и B, для которой одному экземпляру сущности A может соответствовать 0, 1 или несколько экземпляров сущности B, но каждому экземпляру сущности B соответствует один экземпляр сущности A. При этом однозначно идентифицировать можно только экземпляр сущности A по экземпляру сущности B. Примером связи типа 1:M является связь “учится” между сущностями “учебная группа” и “студент”. Для такой связи, зная конкретного студента, можно однозначно идентифицировать учебную группу,

в которой он учится, или, зная учебную группу, можно определить всех обучающихся в ней студентов.

Связь *многие к одному* по сути эквивалентна связи один ко многим. Различие заключается лишь в том, с точки зрения какой сущности (A или B) данная связь рассматривается.

Связь *многие ко многим* определяет такой тип связи между типами сущностей A и B, при котором каждому экземпляру сущности A может соответствовать 0, 1 или несколько экземпляров сущности B, и наоборот. При такой связи, зная экземпляр одной сущности, можно указать все экземпляры другой сущности, относящиеся к исходному, т. е. идентификация сущностей неуникальна в обоих направлениях. В качестве примера такой связи можно рассмотреть связь "изучает" между сущностями "учебная дисциплина" и "учебная группа".

Реально *все связи являются двунаправленными*, т. е., зная экземпляр одной из сущностей, можно идентифицировать (однозначно или многозначно) экземпляр (экземпляры) другой сущности. В некоторых случаях целесообразно рассматривать лишь однонаправленные связи между сущностями в целях экономии ресурсов ЭВМ. Возможность введения таких связей полностью определяется информационными потребностями пользователей. Различают простую и многозначную однонаправленные связи, которые являются аналогами связей типа 1:1 и 1:M с учетом направления идентификации. Так, для простой однонаправленной связи "староста" ("является старостой") между сущностями "учебная группа" и "студент" можно, зная учебную группу, однозначно определить ее старосту, но, зная конкретного студента, нельзя сказать, является ли он старостой учебной группы. Примером многозначной однонаправленной связи служит связь между сущностями "пациент" и "болезнь", для которой можно для каждого пациента можно указать его болезни, но нельзя выявить всех обладателей конкретного заболевания.

Введение однонаправленных связей означает, что в результате анализа потенциальных запросов потребителей установлено, что потребности в информации, аналогичной приведенной в двух последних примерах, у пользователей не будет (и они не будут формулировать соответствующие запросы к БД).

Графически типы сущностей, атрибуты и связи принято изображать прямоугольниками, овалами и ромбами соответственно. На рис. 9.1.4 представлены примеры связей различных типов; на рис. 9.1.5 и 9.1.6 — фрагменты инфологических моделей "студенты" (без указания атрибутов) и "учебный процесс факультета".

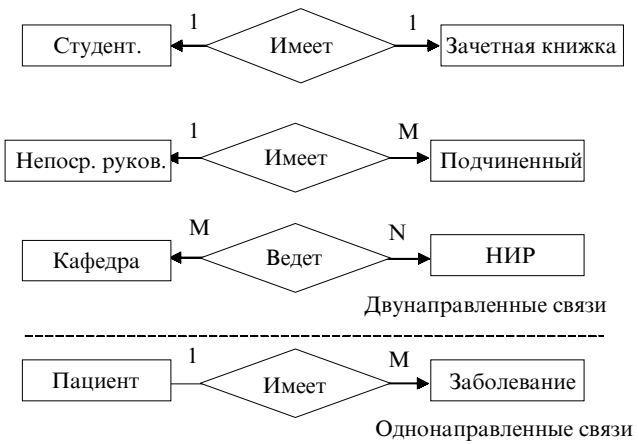


Рис. 9.1.4. Примеры связей между сущностями

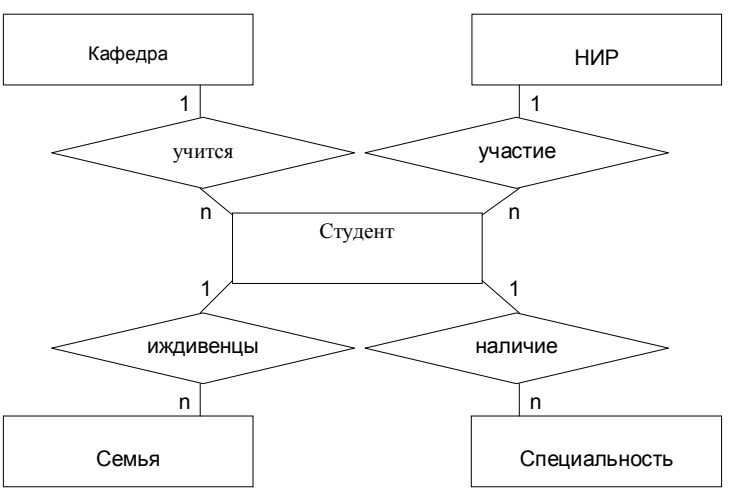


Рис. 9.1.5. Фрагмент ER-модели "Студенты"

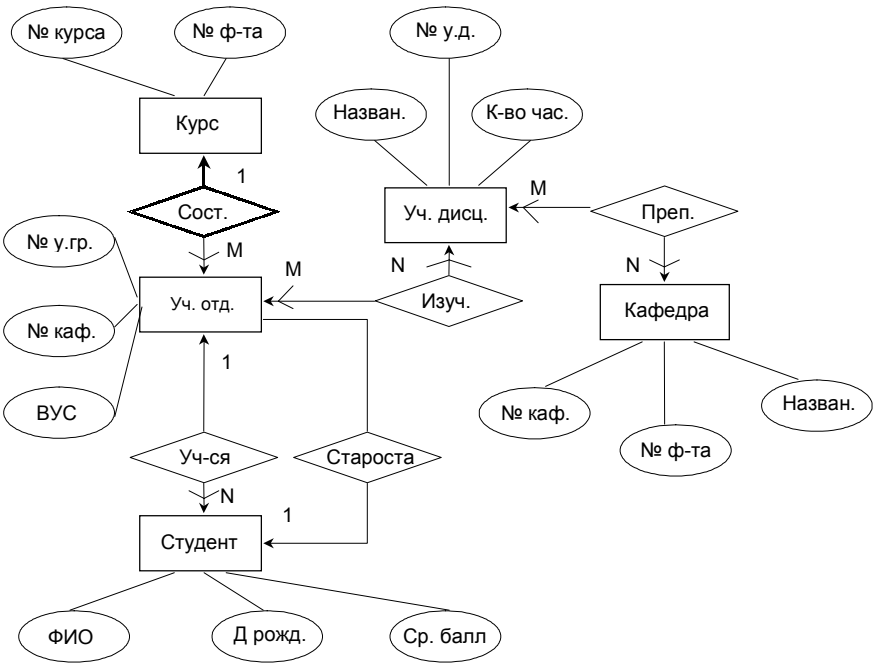


Рис. 9.1.6. Фрагмент ER-модели
"Учебный процесс факультета"

Несмотря на то что построение инфологической модели есть процесс творческий, можно указать *два основополагающих правила*, которыми следует пользоваться всем проектировщикам БД [15, 44]:

• при построении модели *должны использоваться только три типа конструктивных элементов: сущность, атрибут, связь;*

• *каждый компонент информации должен моделироваться только одним из приведенных выше конструктивных элементов* для исключения избыточности и противоречивости описания.

Моделирование ПО *начинают с выбора сущностей*, необходимых для ее описания. Каждая сущность должна соответствовать некоторому объекту (или группе объектов) ПО, о котором в системе будет накапливаться информация. Существует проблема выбора конструктивного элемента для моделирования той или иной "порции" информации, что существенно затрудняет процесс построения модели. Так, информацию о том, что некоторый студент входит в состав учебной группы (УГ), можно в модели представить:

- как связь "входит в состав" для сущностей "студент" и "УГ";
- как атрибут "имеет в составе "студента" сущности "УГ";
- как сущность "состав УГ".

В этих случаях приходится рассматривать несколько вариантов и с учетом информационных потребностей пользователей разбивать ПО на такие фрагменты, которые с их точки зрения представляют самостоятельный интерес.

При моделировании ПО следует обращать внимание на существующий в ней документооборот. Именно документы, циркулирующие в ПО, должны являться основой для формулирования сущностей. Это связано с двумя обстоятельствами:

- во-первых, эти документы, как правило, достаточно полно отражают информацию, которую необходимо хранить в БД, причем в виде конкретных данных;
- во-вторых, создаваемая информационная система должна предоставлять пользователям привычную для них информацию в привычном виде, что в последующем существенно облегчит ввод БД в эксплуатацию.

При описании атрибутов сущности необходимо выбрать *ряд атрибутов, позволяющих однозначно идентифицировать экземпляр сущности. Совокупность идентифицирующих атрибутов называют ключом.*

Помимо идентифицирующих используются и *описательные атрибуты*, предназначенные для более полного определения сущностей. Число атрибутов (их тип) определяется единственным образом — на основе анализа возможных запросов пользователей. Существует ряд рекомендаций по "работе с атрибутами" [15, 44],

например, по исключению повторяющихся групп атрибутов (см. рис. 9.1.7). Все они направлены на улучшение качества инфологической модели.

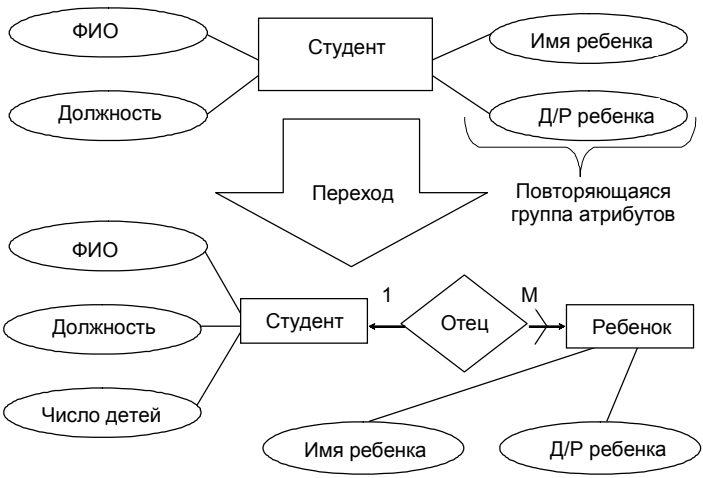


Рис. 9.1.7. Пример исключения повторяющейся группы атрибутов

При определении связей между сущностями следует избегать связей типа M:N, так как они приводят к существенным затратам ресурсов ЭВМ. Устранение таких связей предусматривает введение других (дополнительных) элементов — сущностей и связей. На рис. 9.1.8 приведен пример исключения связи "многие ко многим".

В заключение приведем типовую последовательность работ (действий) по построению инфологической модели:

• *выделение* в ПО *сущностей*;

• *введение множества атрибутов* для каждой сущности и выделение из них *ключевых*;

• *исключение множества повторяющихся* атрибутов (при необходимости);

• *формирование связей* между сущностями;

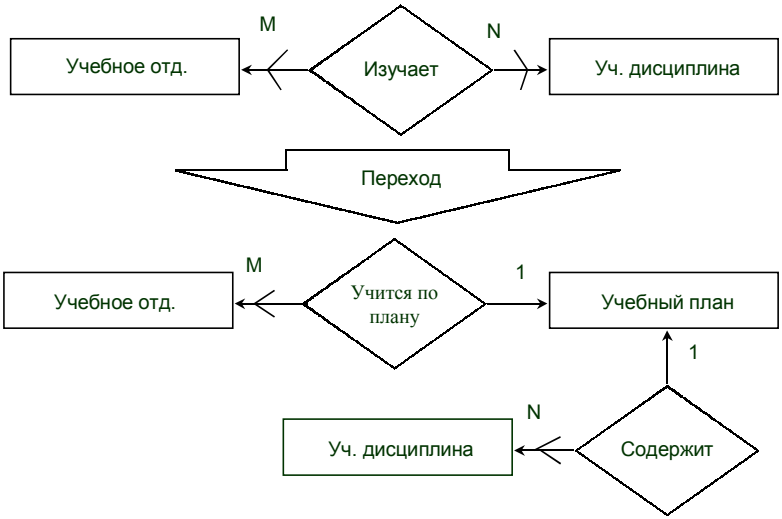


Рис. 9.1.8. Пример исключения связи типа M:N

• *исключение связей типа M:N* (при необходимости);

• *преобразование связей в однонаправленные* (по возможности).

Помимо модели Чена существуют и другие инфологические модели. Все они представляют собой описательные (неформальные) модели, использующие различные конструктивные элементы и соглашения по их использованию для представления в БД информации о ПО. Иными словами, первый этап построения БД всегда связан с моделированием предметной области.

9.1.3. Концептуальные модели данных

В отличие от инфологической модели ПО, описывающей по некоторым правилам сведения об объектах материального мира и связи между ними, которые следует иметь в БД, *концептуальная модель описывает хранимые в ЭВМ данные и связи*. В силу этого *каждая модель данных неразрывно связана с языком описания данных* конкретной СУБД (см. рис. 9.1.3).

По существу, модель данных — это совокупность трех составляющих [15, 44]:

• *типов (структур)* данных;
• *операций* над данными;
• *ограничений* целостности.

Другими словами, модель данных представляет собой некоторое интеллектуальное средство проектировщика, позволяющее реализовать интерпретацию сведений о ПО в виде формализованных данных в соответствии с определенными требованиями, т. е. средство абстракции, которое дает возможность увидеть "лес" (информационное содержание данных), а не отдельные "деревья" (конкретные значения данных).

Типы (структуры) данных

Среди широкого множества определений, обозначающих типы структур данных, наиболее распространена терминология КОДАСИЛ (COnference of DAta SYstems Language) — международной ассоциации по языкам систем обработки данных, созданной в 1959 г.

В соответствии с этой терминологией используют пять типовых структур (в порядке усложнения):

• *элемент* данных;
• *агрегат* данных;
• *запись;*
• *набор;*
• *база данных*.

Дадим краткие определения этих структур [15, 44, 45].

Элемент данных — наименьшая поименованная единица данных, к которой СУБД может адресоваться непосредственно и с помощью которой выполняется построение всех остальных структур данных.

Агрегат данных — поименованная совокупность элементов данных, которую можно рассматривать как единое целое. Агрегат может быть простым или составным (если он включает в себя другие агрегаты).

Запись — поименованная совокупность элементов данных и (или) агрегатов. Таким образом, запись — это агрегат, не входящий в другие агрегаты. Запись может иметь сложную иерархичес-

кую структуру, поскольку допускает многократное применение агрегации.

Набор — поименованная совокупность записей, образующих двухуровневую иерархическую структуру. Каждый тип набора представляет собой связь между двумя типами записей. Набор определяется путем объявления одного типа записи "записью-владельцем", а других типов записей — "записями-членами". При этом каждый экземпляр набора должен содержать один экземпляр "записи-владельца" и любое количество "записей-членов". Если запись представляет в модели данных сущность, то набор — связь между сущностями. Например, если рассматривать связь "учится" между сущностями "учебная группа" и "студент", то первая из сущностей объявляется "записью-владельцем" (она в экземпляре набора одна), а вторая — "записью-членом" (их в экземпляре набора может быть несколько).

База данных — поименованная совокупность экземпляров записей различного типа, содержащая ссылки между записями, представленные экземплярами наборов.

Отметим, что структуры БД строятся на основании следующих главных композиционных правил [15, 44]:

• БД может содержать *любое количество типов записей и типов наборов;*

• между *двумя типами записей* может быть определено *любое количество наборов;*

• *тип записи* может быть *владельцем* и *одновременно членом* нескольких типов наборов.

Следование данным правилам позволяет моделировать данные о сколь угодно сложной ПО с требуемым уровнем полноты и детализации.

Рассмотренные типы структур данных могут быть представлены в различной форме — графовой; табличной; в виде исходного текста языка описания данных конкретной СУБД.

Операции над данными

Операции, реализуемые СУБД, включают *селекцию* (поиск) данных; *действия* над данными.

Селекция данных выполняется с помощью критерия, основанного на использовании либо логической позиции данного (элемента; агрегата; записи), либо значения данного, либо связей между данными [46].

Селекция на основе логической позиции данного базируется на упорядоченности данных в памяти системы. При этом критерии поиска могут формулироваться следующим образом:

- найти следующее данное (запись);
- найти предыдущее данное;
- найти n-е данное;
- найти первое (последнее) данное.

Этот тип селекции называют *селекцией посредством текущей*, в качестве которой используется индикатор текущего состояния, автоматически поддерживаемый СУБД и, как правило, указывающий на некоторый экземпляр записи БД.

Критерий селекции по значениям данных формируется из простых или булевых условий отбора. Примерами простых условий поиска являются:

- УЧ_ДИСЦИПЛИНА = МЕНЕДЖМЕНТ;
- ВОЗРАСТ > 20;
- ДАТА < 19.04.2002 и т. п.

Булевое условие отбора формируется путем объединения простых условий с применением логических операций, например:

- (ДАТА_РОЖДЕНИЯ < 28.12.1953) И (СТАЖ> 35);
- (УЧЕНОЕ_ЗВАНИЕ = ДОЦЕНТ) ИЛИ (УЧЕНОЕ ЗВАНИЕ = = ПРОФЕССОР) и т. п.

Если модель данных, поддерживаемая некоторой СУБД, позволяет выполнить селекцию данных по связям, то можно найти данные, связанные с текущим значением какого-либо данного [46]. Например, если в модели данных реализована двунаправленная связь “учится” между сущностями “студент” и “учебная группа”, можно выявить учебные группы, в которых учатся юноши (если в составе описания студента входит атрибут “пол”).

Как правило, большинство современных СУБД позволяют осуществлять различные комбинации описанных выше видов селекции данных.

Ограничения целостности

Ограничения целостности — логические ограничения на данные — используются для обеспечения непротиворечивости данных некоторым заранее заданным условиям при выполнении операций над ними. По сути, ограничения целостности — это набор правил, используемых при создании конкретной модели данных на базе выбранной СУБД.

Различают *внутренние и явные* ограничения.

Ограничения, обусловленные возможностями конкретной СУБД, называют внутренними ограничениями целостности. Эти ограничения касаются типов хранимых данных (например, "текстовый элемент данных может состоять не более чем из 256 символов" или "запись может содержать не более 100 полей") и допустимых типов связей (например, СУБД может поддерживать только так называемые функциональные связи, т. е. связи типа 1:1, 1:M или M:1). Большинство существующих СУБД поддерживают прежде всего именно внутренние ограничения целостности [46], нарушения которых приводят к некорректности данных и достаточно легко контролируются.

Ограничения, обусловленные *особенностями хранимых данных о конкретной ПО, называют явными ограничениями целостности.* Эти ограничения также поддерживаются средствами выбранной СУБД, но они формируются обязательно с участием разработчика БД путем определения (программирования) специальных процедур, обеспечивающих непротиворечивость данных. Например, если элемент данных "зачетная книжка" в записи "студент" определен как ключ, он должен быть уникальным, т. е. в БД не должно быть двух записей с одинаковыми значениями ключа. Другой пример: пусть в той же записи предусмотрен элемент "военно-учетная специальность" и для него отведено 6 десятичных цифр. Тогда другие представления этого элемента данных в БД невозможны. С помощью явных ограничений целостности можно организовать как "простой" контроль вводимых данных (прежде всего на предмет принадлежности элементов данных фиксированному и заранее заданному множеству значений: например, элемент "ученое звание" не должен принимать значение "почетный до-

цент", если речь идет о российских ученых), так и более сложные процедуры (например, введение значения "профессор" элемента данных "ученое звание" в запись о преподавателе, имеющем возраст 25 лет, должно требовать, по крайней мере, дополнительного подтверждения).

Элементарная единица данных может быть реализована множеством способов, что, в частности, привело к многообразию известных моделей данных. Модель данных определяет правила, в соответствии с которыми структурируются данные. Обычно операции над данными соотносятся с их структурой.

Разнообразие существующих моделей данных соответствует разнообразию областей применения и предпочтений пользователей.

В специальной литературе встречается описание довольно большого количества различных моделей данных. Хотя наибольшее распространение получили иерархическая, сетевая и, бесспорно, реляционная модели, вместе с ними следует упомянуть и некоторые другие.

Используя в качестве классификационного признака особенности логической организации данных, можно привести следующий перечень известных моделей:

- иерархическая модель данных;
- сетевая модель данных;
- реляционная модель данных;
- бинарная модель данных;
- семантическая сеть.

Рассмотрим основные особенности перечисленных моделей.

Иерархическая модель данных

Наиболее давно используемой (можно сказать, классической) является модель данных, в основе которой лежит *иерархическая структура типа дерева*. Дерево — это орграф, в каждую вершину которого кроме первой (корневой) входит только одна дуга, а из любой вершины (кроме конечных) может исходить произвольное число дуг. В иерархической структуре подчиненный элемент данных всегда связан только с одним исходным.

На рис. 9.1.9 показан фрагмент объектной записи в иерархической модели данных. Часто используется также “упорядоченное дерево”, в котором значим относительный порядок поддеревьев.

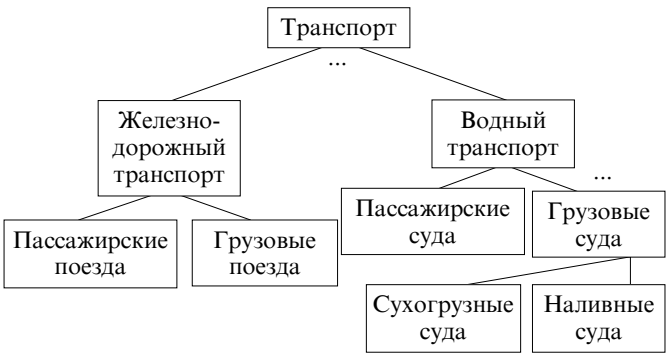


Рис. 9.1.9. Фрагмент иерархической модели данных

Достоинства такой модели несомненны: простота представления предметной области, наглядность, удобство анализа структур и простота их описания. К *недостаткам* следует отнести сложность добавления новых и удаления существующих типов записей, невозможность отображения отношений, отличающихся от иерархических, громоздкость описания и информационную избыточность.

Характерные примеры реализации иерархических структур — язык Кобол и СУБД семейства IMS (создана в рамках проекта высадки на Луну — “Аполлон”) и System 2000 (S2K).

Сетевая модель данных

В системе баз данных, предложенных CODASYL, за основу была взята сетевая структура. Существенное влияние на разработку этой модели оказали более ранние сетевые системы — IDS и Ассоциативный ПЛ/1. Необходимость в процессе получения одного отчета обрабатывать несколько файлов обусловила целесообразность установления перекрестных ссылок между файлами, что в конце концов и привело к сетевым структурам [54].

Сетевая модель данных основана на представлении информации в виде орграфа, в котором в каждую вершину может входить *произвольное число дуг*. Вершинам графа сопоставлены типы записей, дугам — связи между ними. На рис. 9.1.10 представлен пример структуры сетевой модели данных.

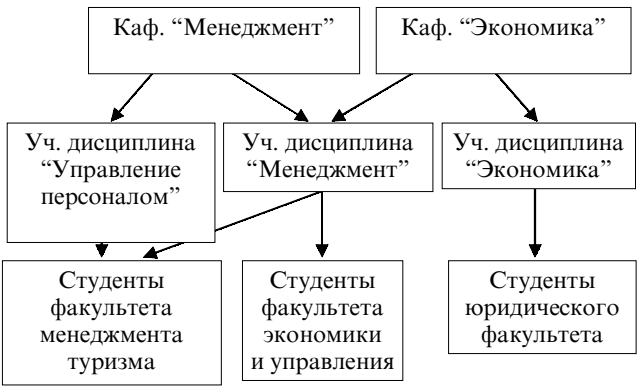


Рис. 9.1.10. Фрагмент сетевой модели данных

По сравнению с иерархическими сетевые модели обладают рядом существенных *преимуществ*: возможность отображения практически всего многообразия взаимоотношений объектов предметной области, непосредственный доступ к любой вершине сети (без указания других вершин), малая информационная избыточность. Вместе с тем в сетевой модели невозможно достичь полной независимости данных [54] — с ростом объема информации сетевая структура становится весьма сложной для описания и анализа.

Известно [62], что применение на практике иерархических и сетевых моделей данных в некоторых случаях требует разработки и сопровождения значительного объема кода приложения, что иногда может стать для информационной системы непосильным бременем.

Реляционная модель данных

В основе реляционной модели данных лежат не графические, а *табличные методы и средства представления данных* и манипу-

лирования ими (рис. 9.1.11). В реляционной модели для отображения информации о предметной области используется таблица, называемая *"отношением"*. Строка такой таблицы называется *кортежем*, столбец — *атрибутом*. Каждый атрибут может принимать некоторое подмножество значений из определенной области — *домена* [42].

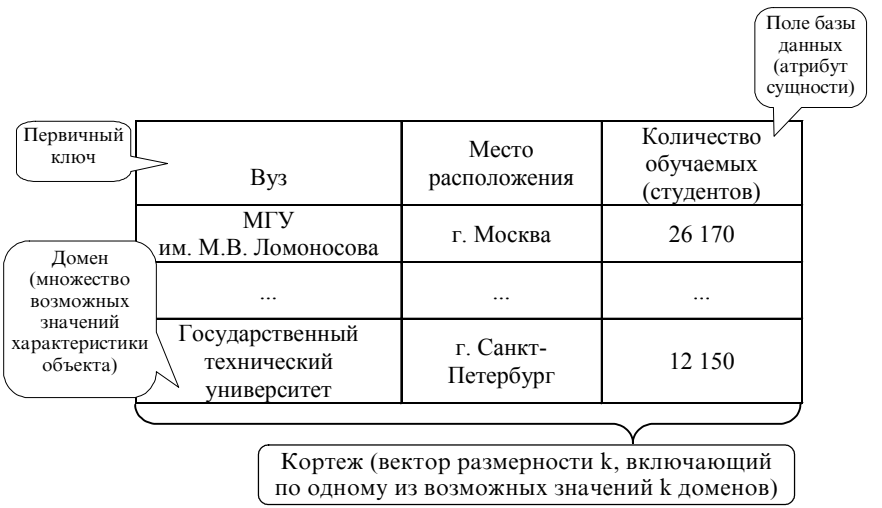


Рис. 9.1.11. Фрагмент реляционной модели данных

Табличная организация БД позволяет реализовать ее важнейшее *преимущество* перед другими моделями данных, а именно — возможность использования точных математических методов манипулирования данными, и прежде всего — аппарата реляционной алгебры и исчисления отношений [54]. К другим достоинствам реляционной модели можно отнести наглядность, простоту изменения данных и организации разграничения доступа к ним.

Основным *недостатком* реляционной модели данных является информационная избыточность, что ведет к перерасходу ресурсов вычислительных систем (отметим, что существует ряд приемов, позволяющих в значительной степени избавиться от этого недостатка — см. п. 9.2). Однако именно реляционная модель данных нахо-

дит все более широкое применение в практике автоматизации информационного обеспечения профессиональной деятельности.

Подавляющее большинство СУБД, ориентированных на персональные ЭВМ, являются системами, построенными на основе реляционной модели данных — так называемыми "реляционными СУБД".

Бинарная модель данных

Бинарная модель данных — это графовая модель, в которой *вершины являются представлениями простых однозначных атрибутов, а дуги — представлениями бинарных связей между атрибутами* (рис. 9.1.12).

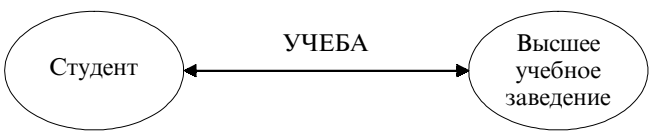


Рис. 9.1.12. Пример бинарного отношения

Бинарная модель не получила особо широкого распространения, но в ряде случаев находит практическое применение.

Так, в области искусственного интеллекта уже давно ведутся исследования с целью представления информации в виде бинарных отношений. Рассмотрим триаду (тройку) "объект — атрибут — значение" (более подробно об этом будет сказано в п. 10.2). Триада "Кузнецов — возраст — 20" означает, что возраст некоего Кузнецова равен 20 годам. Эта же информация может быть выражена, например, бинарным отношением ВОЗРАСТ. Понятие бинарного отношения положено в основу таких моделей данных, как, например, Data Semantics (автор Абриал) и DIAM II (автор Сенко).

Бинарные модели данных обладают возможностью представления связей любой сложности (и это их несомненное преимущество), но вместе с тем их ориентация на пользователя недостаточна [53].

Семантическая сеть

Семантические сети как модели данных были предложены исследователями, работавшими над различными проблемами *ис-*

кусственного интеллекта (см. п. 10). Так же, как в сетевой и бинарной моделях, базовые структуры семантической сети могут быть представлены графом, множество вершин и дуг которого образует сеть. Однако семантические сети предназначены для представления и систематизации знаний самого общего характера [53].

Таким образом, семантической сетью можно считать любую графовую модель (например — помеченный бинарный граф) при условии, что изначально четко определено, что обозначают вершины и дуги и как они используются.

Семантические сети являются богатыми источниками идей моделирования данных, чрезвычайно полезных в плане решения проблемы представления сложных ситуаций. Они могут быть использованы независимо или совместно с идеями, положенными в основу других моделей данных. Их интересной особенностью является то, что расстояние, измеренное на сети (семантическое расстояние или метрика), играет важную роль, определяя близость взаимосвязанных понятий. При этом предусмотрена возможность в явной форме подчеркнуть, что семантическое расстояние велико. Как показано на рис. 9.1.13, СПЕЦИАЛЬНОСТЬ соотносится с личностью ПРЕПОДАВАТЕЛЬ, и в то же время ПРЕПОДАВАТЕЛЮ присущ РОСТ. Взаимосвязь личности со специальностью очевидна, однако из этого не обязательно следует взаимосвязь СПЕЦИАЛЬНОСТИ и РОСТА.

Следует сказать, что моделям данных типа семантической сети при всем богатстве их возможностей при моделировании сложных ситуаций присуща усложненность и некоторая неэкономичность в концептуальном плане [53].

9.1.4. Реляционная модель данных

Как было отмечено в п. 9.1.3, в основе реляционной модели данных лежит их представление в виде таблиц, что в значительной степени облегчает работу проектировщика БД и — в последующем — пользователя в силу привычности и распространенности такого варианта использования информации. Данная модель была предложена Э. Ф. Коддом (E. F. Codd) в начале 70-х гг. прошлого века и вместе с иерархической и сетевой моделями составляет множество так называемых *великих* моделей. Можно сказать, что се-

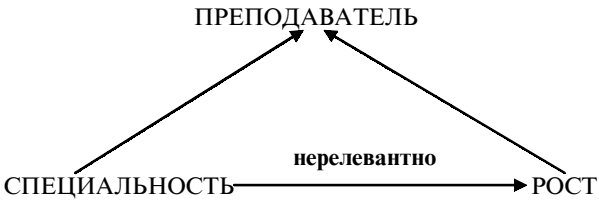


Рис. 9.1.13. Соотношение понятий в семантической сети

годня именно эта модель используется во всех наиболее распространенных СУБД.

Определение любой модели данных требует *описания трех элементов*:

- определение *типов (структур) данных*;
- определение *операций над данными*;
- определение *ограничений целостности*.

Сначала рассмотрим структуры данных и ограничения целостности, а затем более подробно остановимся на операциях реляционной алгебры.

Типы (структуры) данных

Рассмотрение этого вопроса требует введения определений нескольких основных понятий.

Множество возможных значений некоторой характеристики объекта называется доменом (англ. domain — область):

$D_i = \{d_{i1}, d_{i2}, ..., d_{ini}\}$.

Например, в качестве домена можно рассматривать такие характеристики студента, как его фамилия, курс, рост и т. п.:

$D_{курс} = \{1, 2, 3, 4, 5,\}$;

$D_{фамилия} = \{Иванов, Петров, Сидоров, ...\}$;

$D_{рост} = \{160, 161, 162, ..., 190\}$.

Очевидно, что можно сопоставить понятия “атрибут” инфологической модели предметной области и “домен” реляционной модели данных. Возможные значения характеристик объектов могут принимать числовые или текстовые значения, а их множества могут быть как конечными, так и бесконечными. Отметим, что в случае конечности домена можно организовать проверку

явных ограничений целостности: в нашем примере домен $D_{рост}$ определяет, что все студенты должны иметь рост от 160 до 190 см, а номер курса не может превышать 5.

Вектор размерности k, включающий в себя по одному из возможных значений *k* доменов, называется *кортежем*. Для приведенного выше примера кортежами являются:

(1, Иванов, 172);

(3, Сидоров, 181);

(5, Уткин, 184).

Часто количество кортежей в таблице (отношении) называют *кардинальным числом*, а количество атрибутов — *степенью*.

Если в кортеж входят значения всех характеристик объекта ПО (т. е. атрибутов сущности инфологической модели), ему можно сопоставить такую типовую структуру данных, как *запись* (объектная запись).

Декартовым произведением k доменов называется множество всех возможных значений кортежей:

$$D = D_1 \times D_2 \times ... \times D_k.$$

Пусть для того же примера определены три домена:

$D_1 = \{1, 4\}$;

$D_2 = \{$Иванов, Петров$\}$;

$D_3 = \{168, 181\}$.

Тогда их декартовым произведением будет множество D, состоящее из восьми записей:

$$D = D_1 \times D_2 \times D_3 = \begin{Bmatrix} (1,\ \text{Иванов},\ 168) \\ (1,\ \text{Иванов},\ 181) \\ (1,\ \text{Петров},\ 168) \\ (1,\ \text{Петров},\ 181) \\ (4,\ \text{Иванов},\ 168) \\ (4,\ \text{Иванов},\ 181) \\ (4,\ \text{Петров},\ 168) \\ (4,\ \text{Петров},\ 181) \end{Bmatrix}.$$

При увеличении размерности любого из доменов увеличивается и размерность их декартова произведения. Так, если в первом домене определены три элемента

$D_1 = \{1, 4, 5\}$, декартово произведение имеет вид:

$$D = D_1 \times D_2 \times D_3 = \left\{ \begin{array}{l} (1,\ \text{Иванов},\ 168) \\ (1,\ \text{Иванов},\ 181) \\ (1,\ \text{Петров},\ 168) \\ (1,\ \text{Петров},\ 181) \\ (4,\ \text{Иванов},\ 168) \\ (4,\ \text{Иванов},\ 181) \\ (4,\ \text{Петров},\ 168) \\ (4,\ \text{Петров},\ 181) \\ (5,\ \text{Иванов},\ 168) \\ (5,\ \text{Иванов},\ 181) \\ (5,\ \text{Петров},\ 168) \\ (5,\ \text{Петров},\ 181) \end{array} \right\}.$$

Иными словами, декартово произведение — множество всех возможных комбинаций элементов исходных доменов.

Наконец, важнейшее определение: *отношением (relation) R,* определенным на множествах доменов D_1, D_2, ..., D_k, называют *подмножество их декартова произведения:*

$R \subseteq D = D_1 \times D_2 \times ... \times D_k$.

Элементами отношения являются кортежи. Отношение может моделировать множество однотипных объектов (сущностей), причем экземпляр сущности может интерпретироваться как кортеж. С помощью отношения можно моделировать и связи, в которых находятся объекты ПО (сущности в ее инфологической модели). При этом кортеж такого отношения состоит из идентифицирующих атрибутов связываемых сущностей.

Таким образом, понятие отношения позволяет моделировать данные и связи между ними. В силу этого можно определить

реляционную базу данных (РБД) как совокупность экземпляров конечных отношений.

Если учесть, что результат обработки любого запроса к РБД также можно интерпретировать как отношение (возможно, не содержащее ни одного кортежа), то возникает возможность построения информационной системы, основным инструментом которой будет алгебра отношений (реляционная алгебра). Любой запрос в такой системе может быть представлен в виде формулы, состоящей из отношений, объединенных операциями реляционной алгебры. Создав СУБД, обеспечивающую выполнение этих операций, можно разрабатывать информационные системы, в которых *любой запрос потребителя программируется формулой.*

Ограничения целостности

Отношение может быть представлено таблицей, обладающей определенными свойствами (которые, по сути, и определяют внутренние ограничения целостности данных) [54]:

- каждая строка таблицы — кортеж;
- порядок строк может быть любым;
- повторение строк не допускается;
- порядок столбцов в отношении фиксирован.

Понятие "отношение" весьма схоже с понятием "файл данных". Поэтому в дальнейшем будем использовать следующую терминологию: отношение — файл; кортеж — запись; домен — поле. Идентификация конкретной записи файла осуществляется по ключу (набору полей, по значению которого можно однозначно идентифицировать запись). В файле можно определить несколько ключей. Один из них, включающий минимально возможное для идентификации записи число полей, называется *первичным ключом*.

Применительно к понятию "файл данных" внутренние ограничения целостности формулируются следующим образом:

- количество полей и их порядок в файле должно быть фиксированным (т. е. записи файла должны иметь одинаковые длину и формат);
- каждое поле должно моделировать элемент данных (неделимую единицу данных фиксированного формата, к которому СУБД может адресоваться непосредственно);

• в файле не должно быть повторяющихся записей.

СУБД, основанные на РБД, поддерживают и явные ограничения целостности. На практике они определяются зависимостями между атрибутами (см. п. 9.1.2).

9.1.5. Операции реляционной алгебры

Операции реляционной алгебры лежат в основе языка манипулирования данными СУБД, основанных на РМД. Эти операции *выполняются над файлами*, и результатом их выполнения также является *файл*, который в общем случае может оказаться и пустым.

При описании операций реляционной алгебры будем использовать обозначения:

• ИФ (ИФ1; ИФ2) — имя исходного (первого исходного; второго исходного) файла (файлов);

• ФР — имя файла-результата.

Некоторые операции накладывают на исходные файлы ограничения, которые в определенном смысле можно рассматривать как внутренние ограничения целостности.

Проектирование

Формальная запись:

ФР = proj [список имен полей] (ИФ).

Операция не накладывает ограничений на исходный файл и предусматривает следующие действия:

• из ИФ исключаются все поля, имена которых отсутствуют в списке имен полей;

• из полученного файла удаляются повторяющиеся записи.

Пример. Пусть ИФ (КАДРЫ) содержит 4 поля:

КАДРЫ

НОМЕР	ДОЛЖНОСТЬ	ФАМИЛИЯ	П/Я
01	инженер	Петров	34170
02	инженер	Горин	11280
03	ст. инженер	Сидоров	34170
04	нач. цеха	Фомин	27220
05	нач. цеха	Николаев	11280

и требуется выполнить операцию

ФР = proj [П/Я] (КАДРЫ).

Тогда после выполнения операции получим результат

ФР

П/Я
34 170
11 280
27 220

Заметим, что с помощью приведенной операции можно выявить, в каких почтовых ящиках работают сотрудники, информация о которых содержится в данном файле.

Селекция (выбор)

Формальная запись:

ФР = sel [условие] (ИФ).

Эта операция также не накладывает ограничений на ИФ. В файл результата заносятся те записи из ИФ, которые удовлетворяют условию поиска. Условие представляет собой логическое выражение, связывающее значения полей ИФ.

Пример. Пусть для приведенного выше ИФ КАДРЫ требуется выявить сотрудников П/Я 34 170, имеющих должность “ст. инженер”. Для отработки такого запроса достаточно выполнить операцию:

ФР = *sel* [ДОЛЖНОСТЬ = “ст. инженер” И П/Я = “34 170”] (КАДРЫ).

ФР

НОМЕР	ДОЛЖНОСТЬ	ФАМИЛИЯ	П/Я
03	ст. инженер	Сидоров	34170

Отметим, что данная операция не изменяет структуру ИФ. Кроме того, при такой формальной записи операции предполагается, что СУБД поддерживает отработку сложных (составных) запросов, в противном случае пришлось бы составное условие поиска отрабатывать последовательно — сначала выявить сотрудников, имеющих должность “ст. инженер”, а затем — из них выделить тех, кто работает на П/Я 34 170 (или наоборот). Иногда та-

кой (последовательный) порядок поиска имеет определенные преимущества — прежде всего в тех случаях, когда на сложный запрос дан отрицательный ответ и непонятно, что послужило причиной этого (в нашем примере — или нет сотрудников в должности “ст. инженер”, или никто из них не “работает” в указанном П/Я, или такого предприятия вообще “нет” в БД).

Соединение

Формальная запись:

ФР = ИФ1 ►◄ ИФ2.

(список полей)

В реляционной алгебре определено несколько операций соединения. Мы рассмотрим так называемое *естественное соединение*.

Условием выполнения данной операции является *наличие в соединяемых файлах одного или нескольких однотипных полей*, по которым и осуществляется соединение (эти поля указываются в списке; если список пуст, соединение осуществляется по всем однотипным полям).

В файл результата заносятся записи, являющиеся так называемыми конкатенациями (англ. concatenate — сцеплять, связывать) записей исходных файлов. Иными словами, в ФР попадают записи ИФ1 и ИФ2 с совпадающими значения полей, по которым осуществляется соединение (“сцепка”).

Пример. Пусть помимо файла КАДРЫ имеется файл ЦЕХ, в котором указаны порядковый НОМЕР сотрудника (как и в первом файле) и НОМЕР_ЦЕХА — номер цеха, в котором данный сотрудник работает.

ЦЕХ

НОМЕР	НОМЕР_ЦЕХА
01	Ц1
02	Ц2
03	Ц1
04	Ц2
05	Ц1

Тогда после выполнения операции
ФР = КАДРЫ ►◄ ЦЕХ
получим

ФР

НОМЕР	ДОЛЖНОСТЬ	ФАМИЛИЯ	П/Я	НОМЕР_ ЦЕХА
01	инженер	Петров	34 170	Ц1
02	инженер	Горин	11 280	Ц2
03	ст. инженер	Сидоров	34 170	Ц1
04	нач. цеха	Фомин	27 220	Ц2
05	нач. цеха	Николаев	11 280	Ц1

Следует обратить внимание, что в формате команды не указаны поля соединения. Следовательно, оно осуществляется по единственному однотипному полю (НОМЕР).

Пример. Пусть требуется выяснить, в каком цехе п/я 34 170 работает ст. инженер Сидоров.

Для этого требуется выполнить операции:

ФР1 = sel [ДОЛЖНОСТЬ = “ст. инженер” И П/Я = “34 170” И ФАМИЛИЯ = “Сидоров”] (КАДРЫ);

ФР2 = ФР1 ►◄ ЦЕХ;

ФР = proj [ДОЛЖНОСТЬ, ФАМИЛИЯ, П/Я, НОМЕР_ЦЕХА]ФР2 (ДОЛЖНОСТЬ).

В результате получим:

ФР

ДОЛЖНОСТЬ	ФАМИЛИЯ	П/Я	НОМЕР_ ЦЕХА
ст. инженер	Сидоров	34 170	Ц1

Объединение

Формальная запись:

ФР = ИФ1 ⋃ ИФ2.

Условием выполнения операции является *однотипность (одинаковая структура) исходных файлов.*

В файл результата заносятся неповторяющиеся записи исходных файлов.

Пример. Пусть в БД имеются два файла: УЧ_Д_КАФЕДРЫ_1 и УЧ_Д_КАФЕДРЫ_2, в которых содержатся данные о читаемых кафедрами № 1 и № 2 учебных дисциплинах:

УЧ_Д_КАФЕДРЫ_1

НОМЕР_ДИСЦИПЛИНЫ	УЧЕБНАЯ_ГРУППА
2011-12	99/ЭВ. 3-02
5300-43	96/ЭИ. 6-01
5140-11	98/ЭВ. 4-03

УЧ_Д_КАФЕДРЫ_2

НОМЕР_ДИСЦИПЛИНЫ	УЧЕБНАЯ_ГРУППА
5110-15	97/ЭИ. 5-02
5413-23	98/ЭВ. 4-01
2010-19	01/ЭИ. 1-03
5300-43	96/ЭИ. 6-01

Тогда после выполнения операции объединения

ФР = УЧ_Д_КАФЕДРЫ1 $\cup$ УЧ_Д_КАФЕДРЫ2.

получим данные об учебных дисциплинах, читаемых обеими кафедрами:

ФР

НОМЕР_ДИСЦИПЛИНЫ	УЧЕБНАЯ_ГРУППА
5110-15	97/ЭИ. 5-02
5413-23	98/ЭВ. 4-01
2010-19	01/ЭИ. 1-03
5300-43	96/ЭИ. 6-01
2011-12	99/ЭВ. 3-02
5140-11	98/ЭВ. 4-03

Напомним, что последовательность записей в файлах БД роли не играет.

Разность (вычитание)

Формальная запись:

ФР = ИФ1 – ИФ2.

Условием выполнения операции является *однотипность (одинаковая структура) исходных файлов.*

В файл результата заносятся записи первого исходного файла, которых нет во втором.

Пример. В условиях предыдущего примера выполним операцию

ФР = УЧ_Д_КАФЕДРЫ1 – УЧ_Д_КАФЕДРЫ2.

Получим данные об учебных дисциплинах, читаемых кафедрой № 1 без участия кафедры № 2.

ФР

НОМЕР_ДИСЦИПЛИНЫ	УЧЕБНАЯ_ГРУППА
2011-12	99/ЭВ. 3-02
5140-11	98/ЭВ. 4-03

Пересечение

Формальная запись:

ФР = ИФ1 ⋂ ИФ2.

Условием выполнения операции является *однотипность (одинаковая структура) исходных файлов.*

В результирующий файл заносятся записи, присутствующие в обоих исходных файлах.

Пример. Для уже известных файлов УЧ_Д_КАФЕДРЫ1 и УЧ_Д_КАФЕДРЫ2 выполним операцию пересечения

ФР = УЧ_Д_КАФЕДРЫ1 ⋂ УЧ_Д_КАФЕДРЫ2.

Получим данные о совместно читаемых обеими кафедрами дисциплинах:

ФР

НОМЕР_ДИСЦИПЛИНЫ	УЧЕБНАЯ_ГРУППА
5300-43	533

Деление

Формальная запись:

ФР = ИФ1 ÷ ИФ2.

Для выполнимости операции деления необходимо, чтобы *в первом исходном файле было больше полей, чем во втором, и для каждого поля второго исходного файла существовало однотипное ему поле в первом исходном файле*.

В файл результата, состоящий из полей первого исходного файла, не входящих во второй, заносятся те записи, которые *согласуются со всеми записями* второго исходного файла.

Пример. Пусть в БД хранятся два файла, содержащие данные об учебной литературе, выпущенной некоторой кафедрой:

АВТОРЫ

АВТОР	НАЗВАНИЕ	ГОД_ИЗДАНИЯ
Иванов	Теория вероятностей	1997
Петров	Методы оптимизации	1998
Петров	Теория вероятностей	1997
Иванов	Методы оптимизации	1998
Сидоров	Методы оптимизации	1998
Петров	Концепции естествознания	1996
Сидоров	Исследование операций	1999

ИЗДАНИЯ

НАЗВАНИЕ	ГОД_ИЗДАНИЯ
Теория вероятностей	1997
Методы оптимизации	1998

После выполнения операции деления первого файла на второй (а она возможна, так как в файле АВТОРЫ имеются все поля файла ИЗДАНИЯ) получим данные об авторах (соавторах), которые приняли участие в написании всех книг, информация о которых хранится во втором файле:

ФР

АВТОР
Иванов
Петров

Умножение

Формальная запись:

ФР = ИФ1 × ИФ2.

Условием выполнения операции умножения является *отсутствие в исходных файлах полей с одинаковыми именами.*

В файл результата, содержащий поля обоих исходных файлов, заносятся все возможные комбинации записей ИФ1 и ИФ2.

Пример. Пусть в БД хранятся данные об инженерах и старших инженерах (в файлах СТАРШИЕ_ИНЖЕНЕРЫ и ИНЖЕНЕРЫ соответственно).

СТАРШИЕ_ИНЖЕНЕРЫ

ДОЛЖНОСТЬ	ФАМИЛИЯ
ст. инженер Ц1	Иванов
ст. инженер Ц2	Петров

ИНЖЕНЕРЫ

ДОЛЖНОСТЬ	ФАМИЛИЯ
инженер Ц1	Сидоров
инженер Ц1	Леонидов
инженер Ц3	Дмитриев

Требуется получить данные о возможных вариантах комплектования дежурных смен управления предприятием в составе одного старшего инженера и одного инженера.

Поскольку имена полей в ИФ1 и ИФ2 совпадают, необходимо в одном из них (например, в ИФ2) поля переименовать (например, вместо ДОЛЖНОСТЬ — ДОЛЖНОСТЬ1; вместо ФАМИЛИЯ — ФАМИЛИЯ1). Тогда после выполнения операции

ФР = СТАРШИЕ_ИНЖЕНЕРЫ × ИНЖЕНЕРЫ

получим:

ФР

ДОЛЖНОСТЬ	ФАМИЛИЯ	ДОЛЖНОСТЬ1	ФАМИЛИЯ1
ст. инженер Ц1	Иванов	инженер Ц1	Сидоров
ст. инженер Ц1	Иванов	инженер Ц1	Леонидов

ст. инженер Ц1	Иванов	инженер Ц3	Дмитриев
ст. инженер Ц2	Петров	инженер Ц1	Сидоров
ст. инженер Ц2	Петров	инженер Ц1	Леонидов
ст. инженер Ц2	Петров	инженер Ц3	Дмитриев

С помощью приведенных выше восьми операций реляционной алгебры можно найти ответ на любой запрос к БД, если, конечно, интересующие пользователя данные в ней хранятся. *Типовые запросы* могут *быть запрограммированы заранее* и отрабатываться как процедуры (транзакции). *Обработка уникальных (нетиповых) запросов* должна предусматривать *оперативную разработку* последовательности необходимых операций и последующую ее реализацию.

Вопросы для самопроверки

1. Дайте определение понятия “информация”.
2. Дайте определение понятия “данные”.
3. Вспомните и охарактеризуйте основные элементы банка данных.
4. Для чего создается словарь данных?
5. Назовите этапы создания базы данных.
6. Какие элементы входят в модель П. Чена?
7. Дайте определение понятия “концептуальная модель данных”.
8. Какие модели данных принято называть великими?
9. Дайте определения основных понятий реляционной алгебры.
10. Каким требованиям должно удовлетворять отношение?
11. Какие операции реляционной алгебры не накладывают ограничений на исходные файлы (файл)?

9.2. Нормализация файлов базы данных

В предыдущих пунктах файлы рассматривались как своеобразные хранилища данных и связей между ними, причем было показано, что при соблюдении определенных правил эти файлы

можно считать отношениями и применять к ним операции реляционной алгебры. Открытым пока остался вопрос о том, какие файлы хранить в БД и какие в них должны быть поля, чтобы иметь модель предметной области с определенными положительными свойствами.

9.2.1. Полная декомпозиция файла

Пример. Пусть имеется исходный файл, в котором хранятся данные о сотрудниках, осуществлявших управление непрерывным производственным циклом предприятия в качестве оперативного дежурного (ОД) или его помощника (ПОД) и имеющих номера рабочих телефонов, указанные в поле ТЕЛЕФОН:

ИФ

ДЕЖУРСТВО	СОТРУДНИК	ТЕЛЕФОН
ПОД	Иванов	3-12
ОД	Сидоров	3-12
ОД	Фомин	8-44
ПОД	Семин	8-44

Найдем две проекции ИФ:

ПФ1 = proj [ДЕЖУРСТВО, СОТРУДНИК] (ИФ);

ПФ2 = proj [СОТРУДНИК, ТЕЛЕФОН] (ИФ).

ПФ1

ДЕЖУРСТВО	СОТРУДНИК
ПОД	Иванов
ОД	Сидоров
ОД	Фомин
ПОД	Семин

ПФ2

СОТРУДНИК	ТЕЛЕФОН
Иванов	3-12
Сидоров	3-12
Фомин	8-44
Семин	8-44

Нетрудно убедиться, что соединение этих двух проекций образует исходный файл:

ПФ1 ►◄ ПФ2 = ИФ.

Полной декомпозицией файла называется совокупность произвольного числа его проекций, соединение которых идентично исходному файлу.

Говоря о полной декомпозиции файла, следует иметь в виду *два обстоятельства*:

• во-первых, у одного и того же файла может быть *несколько полных декомпозиций;*

• во-вторых, *не всякая совокупность проекций файла образует его полную декомпозицию.*

Для последнего примера найдем другую проекцию ИФ:

ПФ3 = proj [ДЕЖУРСТВО, ТЕЛЕФОН] (ИФ).

ПФ3

ДЕЖУРСТВО	ТЕЛЕФОН
ПОД	3-12
ОД	3-12
ОД	8-44
ПОД	8-44

В результате соединения ПФ2 и ПФ3 получим файл результата ПФ2 ►◄ ПФ3 = ФР ≠ ИФ:

ФР

ДЕЖУРСТВО	ТЕЛЕФОН	СОТРУДНИК
ПОД	3-12	Иванов
ПОД	3-12	*Сидоров*
ОД	3-12	*Иванов*
ОД	3-12	Сидоров
ОД	8-44	Фомин
ОД	8-44	*Семин*
ПОД	8-44	*Фомин*
ПОД	8-44	Семин

В ФР курсивом выделены записи*, которых не было в исходном файле.*

Методы анализа, позволяющие определить, образует ли данная совокупность проекций файла его полную декомпозицию, будут рассмотрены в п. 9.2.4.

Возможность нахождения полной декомпозиции файла ставит вопросы о том, *в каком виде хранить данные в БД? Дает ли декомпозиция файла какие-либо преимущества? В каких условиях эти преимущества проявляются? И т. п.*

9.2.2. Проблема дублирования информации

В некоторых случаях замена исходного файла его полной декомпозицией позволяет избежать дублирования информации.

Пример. Пусть в исходном файле (как и в примере в п. 9.2.1) хранятся данные о сотрудниках, дежуривших в составе оперативной группы управления предприятием (ДАТА — дата дежурства; ТЕЛЕФОН — рабочий телефон сотрудника).

ИФ

ДАТА	СОТРУДНИК	ТЕЛЕФОН
11.01	Иванов	3-12
15.01	Сидоров	4-21
24.01	Сидоров	4-21
30.01	Сидоров	4-21
1.02	Иванов	3-12

Рассмотрим две проекции файла:
ПФ1 = proj [ДАТА, СОТРУДНИК] (ИФ);
ПФ2 = proj [СОТРУДНИК, ТЕЛЕФОН] (ИФ).

ПФ1

ДАТА	СОТРУДНИК
11.01	Иванов
15.01	Сидоров
24.01	Сидоров
30.01	Сидоров
1.02	Иванов

ПФ2

СОТРУДНИК	ТЕЛЕФОН
Иванов	3-12
Сидоров	4-21

Данные проекции образуют полную декомпозицию исходного файла. В ПФ2 номер рабочего телефона каждого сотрудника упоминается однократно, тогда как в ИФ — столько раз, сколько этот сотрудник заступал на дежурство. Очевидно, что для нашего примера разбиение ИФ на проекции позволяет избежать дублирования информации.

Устранение дублирования информации *важно по двум причинам*:

• устранив дублирование, можно добиться *существенной экономии памяти*;

• если некоторое значение поля повторяется несколько раз, то *при корректировке данных необходимо менять содержимое всех этих полей*, в противном случае нарушится целостность данных.

Для того чтобы найти критерий, позволяющий объективно судить о целесообразности использования полной декомпозиции файла с точки зрения исключения дублирования информации, *воспользуемся понятием первичного ключа*. Напомним, что *первичным ключом называют минимальный набор полей файла, по значениям которых можно однозначно идентифицировать запись*. Если значение первичного ключа не определено, то запись не может быть помещена в файл БД.

Можно показать, что для нашего примера проекции

ПФ1 = proj [ДАТА, СОТРУДНИК] (ИФ);

ПФ2 = proj [ДАТА, ТЕЛЕФОН] (ИФ)

образуют полную декомпозицию ИФ, однако они не исключают дублирования информации.

Причина этого заключается в том, что обе приведенные проекции содержат первичный ключ исходного файла (таковым в рассмотренном примере является поле ДАТА, если, конечно, сотрудник не может одновременно находиться в двух и более оперативных группах).

Можно доказать, что *дублирование информации неизбежно, если проекции*, порождающие полную декомпозицию, *содержат общий первичный ключ* исходного файла.

Рассмотрим исходный файл:

ИФ

FX	FY	FZ
x	y	z
x'	y	z

и две его проекции:

ПФ1

FX	FY
x	y
x'	y

ПФ2

FY	FZ
y	z

Для того чтобы вторая запись ИФ отличалась от первой (в противном случае имели бы в файле БД две одинаковые записи, что недопустимо), она формально должна быть представлена одним из семи вариантов:

(x', y, z); (x, y', z); (x, y, z'); (x', y', z); (x, y', z'); (x', y', z'); (x', y, z').

Пусть FY — первичный ключ. Для того чтобы дублирования информации не было, вторая запись ИФ должна быть или (x', y, z), или (x, y, z'), но это противоречит тому, что FY — первичный ключ. Следовательно, для того чтобы дублирования информации не было, необходимо исключить наличие первичного ключа исходного файла в проекциях, образующих его полную декомпозицию.

Другими словами, если существует такая полная композиция файла, которая образована проекциями*, не имеющими первичного ключа исходного файла*, то замена исходного файла этой декомпозицией исключает дублирование информации. Если же полная декомпозиция файла содержит проекции, имеющие общий первичный ключ исходного файла, то замена его полной декомпозицией не исключает дублирования информации.

9.2.3. Проблема присоединенных записей

Рассмотрим уже использованный в п. 9.2.1 *пример*. Пусть в исходном файле хранятся данные о сотрудниках, дежуривших в

составе оперативной группы предприятия (ДАТА — дата дежурства; ТЕЛЕФОН — рабочий телефон сотрудника).

ИФ

ДАТА	СОТРУДНИК	ТЕЛЕФОН
11.01	Иванов	3-12
15.01	Сидоров	4-21
24.01	Сидоров	4-21
30.01	Сидоров	4-21
01.02	Иванов	3-12

Рассмотрим две проекции файла:
ПФ1 = proj [ДАТА, СОТРУДНИК] (ИФ);
ПФ2 = proj [СОТРУДНИК, ТЕЛЕФОН] (ИФ).

ПФ1

ДАТА	СОТРУДНИК
11.01	Иванов
15.01	Сидоров
24.01	Сидоров
30.01	Сидоров
01.02	Иванов

ПФ2

СОТРУДНИК	ТЕЛЕФОН
Иванов	3-12
Сидоров	4-21

В ИФ поле ДАТА является ключом и не может быть пустым. Как поступить, если нужно запомнить данные и номер рабочего телефона нового сотрудника, который еще не дежурил (например, о Смирнове с номером телефона 7-35)? Записать эти данные в ИФ нельзя (первичный ключ не может быть пустым), но можно поместить эти сведения в проекцию ПФ2. При этом ПФ2 формально перестает быть проекцией ИФ, хотя соединение ПФ1 и ПФ2 дает исходный файл (без сведений о Смирнове).

Записи, вносимые в отдельные проекции исходного файла, называются присоединенными. Представление файла в виде его полной декомпозиции может позволить решить проблему присоединен-

ных записей, но важно помнить, что *соединение проекций ИФ может привести к их потере.*

Целесообразность представления ИФ в виде полной декомпозиции с точки зрения решения проблемы присоединенных записей, как и проблемы дублирования информации, *полностью определяется наличием или отсутствием в проекциях ИФ общего первичного ключа.*

Пусть в ИФ БД хранятся данные о сотрудниках, исполняющих обязанности в дежурном расчете (НОМЕР_Р — номер в составе дежурного расчета; ТЕЛЕФОН — номер рабочего телефона).

ИФ

НОМЕР_Р	СОТРУДНИК	ТЕЛЕФОН
1	Иванов	3-12
2	Сидоров	4-21
3	Фомин	8-61

Если считать, что один и тот же сотрудник не может исполнять обязанности нескольких номеров дежурного расчета, то в качестве первичного ключа можно использовать НОМЕР_Р. Полную декомпозицию исходного файла составляют проекции:

ПФ1

НОМЕР_Р	СОТРУДНИК
1	Иванов
2	Сидоров
3	Фомин

ПФ2

НОМЕР_Р	ТЕЛЕФОН
1	3-12
2	4-21
3	8-61

В качестве присоединенных записей можно рассматривать либо добавление нового номера дежурного расчета и фамилии сотрудника, либо нового номера расчета и телефона без указания фамилии сотрудника, однако эту информацию можно внести и в ИФ путем формирования записей типа

НОМЕР_Р	СОТРУДНИК	ТЕЛЕФОН
4	Смирнов	

или

НОМЕР_Р	СОТРУДНИК	ТЕЛЕФОН
4		7-35

Таким образом, представление ИФ в виде проекций, содержащих общий первичный ключ исходного файла, не дает преимуществ с точки зрения решения проблемы присоединенных записей.

Обобщая сказанное, можно сформулировать общее требование к файлу, представление которого в виде полной декомпозиции не имеет смысла.

Говорят, что файл находится в пятой нормальной форме (5 НФ), если у него или нет ни одной полной декомпозиции, или нет ни одной полной декомпозиции, в которую входили бы проекции, не имеющие общего первичного ключа исходного файла.

Если файл не находится в 5 НФ, имеется возможность избежать дублирования информации и потери присоединенных записей, переходя от исходного файла к такой его полной декомпозиции, которая образована проекциями, не содержащими первичный ключ. Если полученные таким образом файлы проекций не находятся в 5 НФ, то каждую из них можно заменить полной декомпозицией и т. д.

Процесс последовательного перехода к полным декомпозициям файлов БД называется нормализацией файлов БД, главная цель которой — исключение дублирования информации и потери присоединенных записей.

9.2.4. Функциональная зависимость полей файла

При обсуждении в предыдущих пунктах полной декомпозиции файла остался открытым вопрос о том, *при каких же условиях некоторые проекции исходного файла образуют его полную декомпозицию.* Естественно, существует возможность взять конкретный файл, заполненный данными, и непосредственно проверить, образуют ли те или иные его проекции при соединении исходный файл. Однако такой путь не является конструктивным, так как,

во-первых, может оказаться достаточно *много вариантов разбиения исходного файла* и, во-вторых, что более важно, *нет гарантии, что при добавлении записей в исходный файл его проекции будут по-прежнему составлять полную декомпозицию.*

Очевидно, необходимо сформулировать критерий, позволяющий даже для незаполненного файла, исходя из возможных значений его полей, судить о возможности получения полной декомпозиции файла из тех или иных его проекций. *Такой критерий строится на понятии функциональной зависимости полей файла* [34].

Пусть X и Y — некоторые непересекающиеся совокупности полей файла. Говорят, что Y находится в функциональной зависимости от X тогда и только тогда, когда с каждым значением X связано не более одного значения Y.

Любые две записи файла, содержащие одинаковые значения X, должны содержать одинаковые значения Y, причем это ограничение действует не только на текущие значения записей файла, но и на все возможные значения, которые могут появиться в файле. Вместе с тем одинаковым значениям Y могут соответствовать различные значения X.

Рассмотрим уже знакомый пример. Пусть в ИФ имеются поля:

ДЕЖУРСТВО	ДОЛЖНОСТЬ	ФАМИЛИЯ	ТЕЛЕФОН

Поле ТЕЛЕФОН находится в функциональной зависимости от полей ДОЛЖНОСТЬ и ФАМИЛИЯ (считаем, что в данном файле не будет храниться информация о сотрудниках-однофамильцах, имеющих одинаковые должности). Понятно, что один и тот же номер рабочего телефона могут иметь несколько сотрудников, т. е. по значению поля ТЕЛЕФОН нельзя однозначно определить должность и фамилию сотрудника.

Пусть X состоит из нескольких полей. Говорят, что Y находится в полной функциональной зависимости от X, если Y функционально зависит от X и функционально не зависит от любого подмножества X′, не совпадающего с X (X′ ⊂ X) [54].

В условиях предыдущего примера поле ТЕЛЕФОН находится в полной функциональной зависимости от совокупности полей ДОЛЖНОСТЬ и ФАМИЛИЯ, поскольку оно не зависит функци-

онально ни от поля ДОЛЖНОСТЬ, ни от поля ФАМИЛИЯ “по отдельности”.

Теперь можно сформулировать критерий (правило), по которому следует исходный файл разбивать на проекции для получения его полной декомпозиции (это утверждение называют *теоремой Хита*).

Пусть имеются *три непересекающиеся совокупности полей исходного файла:* H, J, K. *Если* K *функционально зависит от* J, *то проекции proj* [H, J] (ИФ) и *proj* [J, K] (ИФ) *образуют полную декомпозицию исходного файла.*

Докажем это утверждение. Введем вспомогательный файл

ИФ1 = proj [H, J] (ИФ) ►◄ proj [J, K] (ИФ).

Покажем, что *каждая запись ИФ присутствует в ИФ1, и наоборот*.

А. Возьмем произвольную запись исходного файла: (h, j, k). Очевидно, что ее часть (h, j) принадлежит первой проекции, (j, k) — второй проекции ИФ. По определению операции соединения можно утверждать, что запись (h, j, k) должна присутствовать в файле ИФ1.

Б. Возьмем произвольную запись вспомогательного файла: (h', j', k'). Согласно определению файла ИФ1 можно записать: *proj* [H, J] (ИФ1) = *proj* [H, J] (ИФ). Следовательно, в файле ИФ должна находиться хотя бы одна запись типа (h', j', k"), где k" пока не определено. По аналогии можно записать: *proj* [J, K] (ИФ1) = *proj* [J, K] (ИФ). Следовательно, в файле ИФ должна находиться хотя бы одна запись типа (h", j', k'), где h" пока не определено.

Таким образом, в исходном файле ИФ должны содержаться записи (h', j', k") и (h", j', k'). Но поскольку K функционально зависит от J, можно заключить, что k" = k' и, следовательно, в ИФ имеется запись (h', j', k'), которую мы определили как произвольную запись ИФ1. *Доказательство закончено*.

Вернемся к примеру. Пусть в ИФ имеются поля:

ДЕЖУРСТВО	ДОЛЖНОСТЬ	ФАМИЛИЯ	ТЕЛЕФОН

Так как поле ТЕЛЕФОН находится в функциональной зависимости от полей ДОЛЖНОСТЬ и ФАМИЛИЯ, можно заключить, что полную декомпозицию ИФ следует искать в виде проекций:

ПФ1 = proj [ДЕЖУРСТВО, ДОЛЖНОСТЬ, ФАМИЛИЯ] (ИФ);
ПФ2 = proj [ДОЛЖНОСТЬ, ФАМИЛИЯ, ТЕЛЕФОН] (ИФ).

Для этого примера можно обозначить: H = [ДЕЖУРСТВО]; J = [ДОЛЖНОСТЬ, ФАМИЛИЯ]; K = [ТЕЛЕФОН].

Таким образом, *с помощью понятия функциональной зависимости полей файла можно получать его полные декомпозиции без анализа хранящихся в файле данных еще на этапе проектирования БД, что весьма важно на практике.*

9.2.5. Нормальные формы файла

Как было показано в пп. 9.2.2 и 9.2.3, при некоторых условиях замена файла его полной декомпозицией позволяет исключить дублирование информации и решить проблему присоединенных записей. Таким условием является отсутствие в проекциях, образующих полную декомпозицию файла, общего первичного ключа исходного файла. Теорема Хита создает основу для построения различных полных декомпозиций и поэтому может служить основным инструментом в процессе *нормализации файлов* БД, под которой понимают процесс последовательного приведения файлов БД ко все более совершенному виду, позволяющему решить проблемы дублирования информации и присоединения записей.

При проведении нормализации на каждом шаге проверяется принадлежность файла некоторой нормальной форме. Если он принадлежит этой нормальной форме, проверяется, находится ли он в следующей, и т. д. до 5 НФ. *Принадлежность файла некоторой форме задает необходимые, но недостаточные условия для нахождения в следующей по старшинству форме.*

Еще раз подчеркнем основное достоинство механизма нормализации файлов с помощью исследования функциональной зависимости полей файла: *возможность проведения этой операции на этапе проектирования БД.*

Перечислим основные нормальные формы файлов [54].

Первая нормальная форма (1 НФ)

Файл находится в 1 НФ, если каждое его поле является атомарным (т. е. не содержит более одного значения) и *ни одно из клю-*

чевых полей не является пустым. По существу, принадлежность файла 1 НФ означает, что он соответствует понятию отношения и его ключевые поля заполнены.

Пример. Принципиально существует возможность хранить информацию о профессорско-преподавательском составе кафедры в следующем виде:

ДОЛЖНОСТЬ	ФАМИЛИЯ
заведующий	Иванов
профессор	Петров, Сидоров
доцент	Фомин
преподаватель	Семин, Савин, Федоров

Однако такой файл не находится в 1 НФ (так как поле ФАМИЛИЯ не является атомарным).

Вторая нормальная форма (2 НФ)

Файл находится во 2 НФ, если он находится в 1 НФ и все его поля, не входящие в первичный ключ, связаны полной функциональной зависимостью с первичным ключом.

Третья нормальная форма (3 НФ)

Файл находится в 3 НФ, если он находится во 2 НФ и ни одно из его неключевых полей не зависит функционально от любого другого неключевого поля.

Нормальная форма Бойса — Кодда (усиленная 3 НФ)

Файл находится в НФ Бойса — Кодда, если любая функциональная зависимость между его полями сводится к полной функциональной зависимости от первичного ключа.

Можно показать [54], что *рассмотренные НФ подчиняются правилу вложенности по возрастанию номеров, т. е. если файл находится в 4 НФ, он находится и в 3 НФ, 2 НФ, 1 НФ, и наоборот* (см. рис. 9.2.1).

Помимо описанных выше нормальных форм используется *четвертая* НФ, *основанная на понятии обобщенной функциональной зависимости* [46]. На практике, приведя все файлы *к нормальной форме Бойса — Кодда, можно с большой долей уверенности ут-*

Рис. 9.2.1. Соотношение нормальных форм файлов

верждать, что они находятся и в 5 НФ, т. е. что нормализация файлов БД завершена.

Отметим, что существующие СУБД (например, широко распространенная СУБД Access из пакета MS Office) содержат средства для *автоматического выполнения операций нормализации* (подобные мастеру по анализу таблиц), хотя качество этого анализа зачастую требуют последующего вмешательства разработчика БД [16, 59].

Необходимость нормализации файлов БД (кроме решения уже рассмотренных проблем исключения дублирования и потери присоединенных записей) определяется еще по крайней мере *двумя обстоятельствами* [43]: во-первых, *разумным желанием группировать данные по их содержимому*, что позволяет упростить многие процедуры в БД — от организации разграничения доступа до повышения оперативности поиска данных; во-вторых, *стремлением разработать БД в виде множества унифицированных блоков*, что может облегчить модернизацию отдельных частей базы, а также использовать таблицы одной БД в других.

Важным направлением совершенствования СУБД является их интеллектуализация [см., например, 27], что подробнее будет рассмотрено в п. 10 учебника.

Вопросы для самопроверки

1. Что называют “полной декомпозицией файла”?
2. В чем состоит проблема дублирования информации?

3. В чем состоит проблема присоединенных записей?

4. Дайте определение функциональной зависимости полей файла. Приведите примеры.

5. Дайте определение полной функциональной зависимости полей файла. Приведите примеры.

6. Сформулируйте теорему Хита.

7. В чем идея ее доказательства?

8. Что называют “нормализацией файлов БД”?

9. Когда говорят, что файл находится в первой нормальной форме?

10. Когда говорят, что файл находится в пятой нормальной форме?

9.3. Современные информационные сети

В предыдущих пунктах (9.1 и 9.2) речь шла о реализации концепции централизованной обработки данных в ходе создания и эксплуатации баз и банков данных. Настоящий пункт посвящен рассмотрению особенностей использования другой концепции — концепции распределенной обработки данных, заключающейся в применении различных компьютерных сетей.

Дадим несколько определений.

Сеть (англ. — network) — 1) связный ориентированный *граф*; 2) средство *теледоступа* — сеть передачи данных, вычислительная сеть.

Вычислительная сеть, сеть ЭВМ (англ. — computer network) — сеть передачи данных, в одном или нескольких узлах которой размещены ЭВМ.

Информационная сеть (англ. information network) — *совокупность АИС*, объединенных в единую сеть с помощью средств передачи данных. (Пользователь имеет доступ к информации любой АИС.)

В настоящее время широко используется еще один (в определенной степени обобщающий) термин — *компьютерная сеть,* применяемый для обозначения и вычислительных, и информационных сетей.

Создание ВС позволило:

- расширить круг решаемых задач обработки информации;
- повысить надежность работы за счет дублирования ЭВМ;
- создать новые виды информационного сервиса;
- снизить стоимость обработки информации;
- создать большие распределенные хранилища информации в полном смысле коллективного пользования и др.

Основные требования к ВС:

- открытость;
- ресурсоемкость;
- надежность;
- динамичность;
- комфортность;
- защищенность и др.

На рис. 9.3.1 представлена компьютерная сеть с точки зрения конечного пользователя: он использует ресурсы сети, не задумываясь о ее "устройстве".

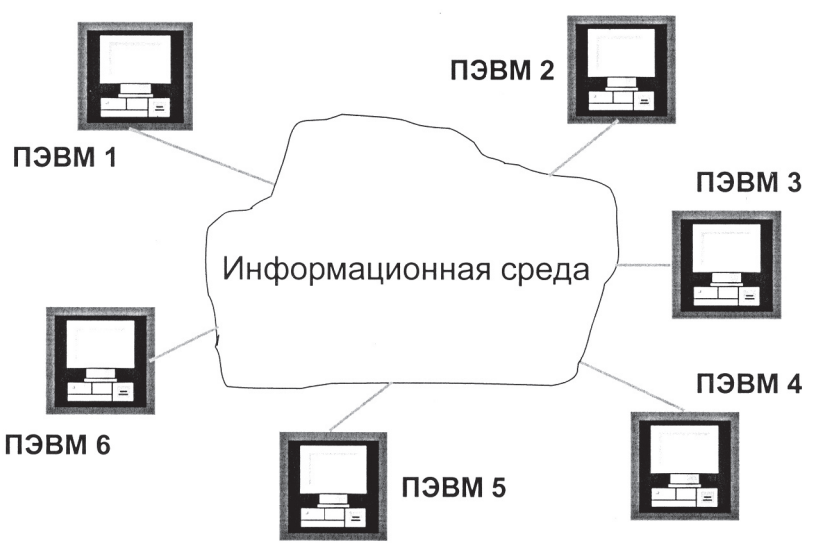


Рис. 9.3.1. Компьютерная сеть с точки зрения пользователя

Структура сети изображена на рис. 9.3.2.

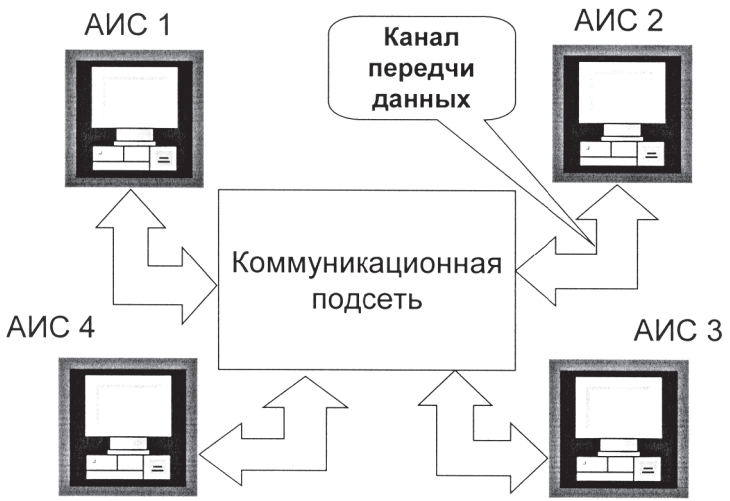


Рис. 9.3.2. Структура компьютерной сети

Все известные компьютерные сети по организационному признаку и по предоставляемому пользователю множеству возможностей для использования информационных ресурсов можно классифицировать следующим образом:

- локальные вычислительные сети;
- сеть Internet (Интернет);
- корпоративные сети Intranet (Интранет);
- сети электронных досок объявлений (сети BBS);
- компьютерные сети на основе FTN-технологий.

В рамках приведенной классификации существуют, создаются и развиваются сети, ориентированные на научную, учебную и учебно-научную проблематику. Бурный рост и повсеместное распространение компьютерных технологий вызывает необходимость более детального изучения организации и построения современных компьютерных сетей, а также рассмотрения возможностей, предоставляемых ими конечному пользователю.

9.3.1. Локальные вычислительные сети

Локальная вычислительная сеть (ЛВС) [32] — это *совокупность технических средств (компьютеров, кабелей, сетевых адаптеров и др.), работающих под управлением сетевой операционной системы и прикладного программного обеспечения*. Внутри одного кабинета, здания или в пределах небольшой территории ЛВС позволяет соединить между собой группу ПК для совместного использования информационных ресурсов. Без ЛВС для обмена данными пришлось бы копировать файлы на дискеты или другие носители информации и передавать их другому пользователю. Такой метод переноса информации на дискетах не позволяет нескольким пользователям одновременно работать с одними и теми же файлами данных. ЛВС предоставляет такую возможность, если используется многопользовательское прикладное программное обеспечение.

Кроме простого *разделения файлов,* пользователи сети *могут разделять принтер, накопитель для компакт-дисков (CD-ROM), модем, факсимильный аппарат, сканер или другое техническое устройство, совместимое с ПК и поддерживающее сетевой режим работы.* Данное обстоятельство является особенно важным при достаточно высоких ценах на соответствующие устройства. Таким образом, пользователи могут разделять ресурсы компьютеров и информацию, находясь на удалении друг от друга; они могут совместно работать над проектами и задачами, которые требуют тесной координации и взаимодействия. Кроме того, даже если компьютерная сеть выйдет из строя, возможно продолжение работы на вашем ПК.

Существует *два типа ЛВС* (данная возможность поддерживается соответствующими операционными системами): *одноранговые сети* и *сети с выделенным сервером (файл-сервером*). Одноранговые сети не предусматривают выделение специальных компьютеров, организующих работу сети. Каждый пользователь, подключаясь к сети, выделяет в сеть какие-либо ресурсы (дисковое пространство, принтеры) и подключается к ресурсам, предоставленным в сеть другими пользователями. Такие сети просты в установке, наладке; они существенно дешевле сетей с выделенным сер-

вером. В свою очередь, сети с выделенным сервером, несмотря на сложность настройки и относительную дороговизну, позволяют осуществлять централизованное управление. В данном случае все ПК, кроме сервера, называются рабочими станциями.

Ниже перечислены семь задач [28], которые решаются с помощью ПК, работающего в составе ЛВС, и которые достаточно трудно решить с помощью отдельного ПК.

1. *Разделение файлов*. ЛВС позволяет многим пользователям одновременно работать с одним файлом, хранящимся на центральном файл-сервере. Например, на предприятии или фирме несколько сотрудников могут одновременно использовать одни и те же руководящие документы.

2. *Передача файлов.* ЛВС позволяет быстро и надежно копировать файлы любого размера с одной машины на другую.

3. *Доступ к информации и файлам.* ЛВС позволяет запускать прикладные программы с любой из рабочих станций, где бы она ни была расположена.

4. *Разделение прикладных программ и баз данных***.** ЛВС позволяет двум пользователям использовать одну и ту же копию программы. При этом, конечно, они не могут одновременно редактировать один и тот же документ или запись в базе данных.

5. *Одновременный ввод данных в прикладные программы.* Сетевые прикладные программы позволяют нескольким пользователям одновременно вводить данные, необходимые для работы этих программ. Например, вести записи в базе данных так, что они не будут мешать друг другу. Однако только специальные сетевые версии программ позволяют одновременный ввод информации. Обычные компьютерные программы позволяют работать с набором файлов только одному пользователю.

6. *Разделение принтера или другого технического устройства***.** ЛВС позволяет нескольким пользователям на различных рабочих станциях совместно использовать один или несколько дорогостоящих лазерных принтеров или других устройств.

7. *Электронная почта***.** Вы можете использовать ЛВС как почтовую службу и рассылать служебные записки, доклады, сообщения и т. п. другим пользователям. В отличие от телефона элек-

тронная почта передаст ваше сообщение даже в том случае, если в данный момент абонент (группа абонентов) отсутствует на своем рабочем месте, причем для этого ей не потребуется бумаги.

При помощи кабеля каждая рабочая станция соединяется с другими станциями и с сервером. В качестве кабеля используются *"толстый" коаксиальный кабель* ("толстый" Ethernet), *"тонкий" коаксиальный кабель* ("тонкий" Ethernet), *витая пара*, *волоконно-оптический кабель*. В последнее время все шире применяются ЛВС, в основе которых для связи между ПК используется *инфракрасный сигнал*. Очевидно, что ПК в этом случае должны находиться в пределах прямой видимости на небольшом расстоянии друг от друга (в пределах одного кабинета). "Толстый" кабель, в свою очередь, используется на участках большой протяженности при требованиях высокой пропускной способности. Волоконно-оптический кабель позволяет создавать протяженные участки без ретрансляторов при недостижимой с помощью других кабелей скорости и надежности. Однако стоимость кабельной сети на его основе высока, и поэтому он не нашел пока широкого распространения в локальных сетях и применяется в более масштабных сетях. В настоящее время локальные компьютерные сети в основном создаются на базе "тонкого" кабеля или витой пары.

Первоначально ЛВС создавались в своем большинстве по принципу "тонкого" Ethernet. В основе его — несколько компьютеров с сетевыми адаптерами, соединенные последовательно коаксиальным кабелем, причем все сетевые адаптеры выдают свой сигнал на него одновременно. Недостатки этого принципа выявились позже.

С ростом размеров ЛВС параллельная работа многих компьютеров на одну единую шину стала практически невозможной из-за значительного взаимного влияния ПК друг на друга. Случайные выходы из строя коаксиального кабеля (например, внутренний обрыв жилы) надолго выводили всю сеть из строя, а определить место обрыва или возникновения программной неисправности, вызвавшей "зависание" сети, становилось практически невозможно.

Сегодня все большее распространение получают технологии беспроводной сетевой связи (WiHi и т. п.), открывающие для пользователей совершенно новые горизонты, прежде всего — за счет упрощения и ускорения создания сети и ее использования в новых сферах.

Поэтому *дальнейшее развитие ЛВС происходит на принципах структурирования.* В этом случае каждая сеть складывается из набора взаимосвязанных участков — структур.

Отдельная структура (отдельная рабочая группа) представляет собой несколько компьютеров с сетевыми адаптерами, каждый из которых соединен отдельным проводом — витой парой — с коммутатором (неким техническим распределительным устройством). При необходимости развития к ЛВС просто добавляют новую структуру (новую рабочую группу).

9.3.2. Всемирная информационная сеть Интернет

Сеть Интернет (Internet) из-за используемых высокоскоростных оптико-волоконных и (или) спутниковых каналов связи часто называют *сетью супермагистралей*, а из-за обилия информационных ресурсов и оперативного применения в сети новейших достижений компьютерных технологий — даже *кибернетическим пространством*. Организационно *Интернет — это сеть, связывающая высокоскоростными каналами связи другие сети, например сети отдельных организаций, ЛВС и др.*

По данным американского центра информации NUA Internet Surveys [62], во всем мире сейчас насчитывается 148 млн пользователей Интернета. Из них в Канаде и США — 87 млн, в Европе — 33 млн. По оценкам ROCIT (Russian Non-Profit Center for Internet Technologies) и РосНИИРОС (Российский НИИ развития общественных сетей), число пользователей Internet в России росло приблизительно следующим образом: январь 1997 г. — 300 тыс.; октябрь 1997 г. — 600 тыс.; июль 1998 г. — 1 млн. Удвоение трафика (объема пересылаемой сетью информации) по данным московских компаний-представителей сервиса Internet (так называемых провайдеров) происходит каждые полгода, а количества пользо-

вателей — каждые 2 года. Недельная аудитория Интернет только в Москве составляет порядка 440 тыс. чел.

Для подключения к удаленным компьютерным сетям используются телефонные линии связи (или более совершенные). Процесс передачи данных по телефонным линиям происходит в форме электрических колебаний — аналога звукового сигнала, в то время как в компьютере информация хранится в виде кодов. Для передачи информации от компьютера через телефонную линию, коды должны быть преобразованы в электрические колебания. Этот процесс называется модуляция. Для того чтобы адресат смог прочитать на своем компьютере то, что ему отправлено, электрические колебания обратно преобразуются в машинные коды — демодулируются. Устройство, осуществляющее преобразование данных из цифровой формы, в которой они хранятся в компьютере, в аналоговую (электрические колебания), в которой они могут быть переданы по телефонной линии, и обратно называется *модемом* (сокращенно от МОдулятор-ДЕМодулятор). Таким образом, отдельный ПК при помощи специальной телекоммуникационной программы, управляющей модемом, связывается по телефонной линии с провайдером, а далее через провайдера — по высокоскоростным каналам связи (оптико-волоконная или спутниковая связь) с необходимым адресатом в Интернете.

Сеть Интернет появилась в США в результате исследования методов построения сетей, устойчивых к частичным повреждениям, получаемым, например, при бомбардировке их авиацией, и способных в таких условиях продолжать нормальное функционирование. В 1960-х гг. исследователи начали экспериментировать с соединением множества различных типов компьютеров посредством телефонных линий, пользуясь фондами Агентства перспективных исследований (ARPA) Министерства обороны США. Созданная сеть получила название ARPANET (Advanced Research Projects Agency Network). Она была предназначена для облегчения обмена информацией между военными ведомствами и их субподрядчиками по различным государственным проектам. Многие ведущие специалисты по компьютерам из промышленных организаций и академий получили доступ к данной сети благодаря

CSNET (Computer Science Network) — проекту, созданному Национальным научным фондом (NSF), еще одним государственным агентством [34]. Вскоре все военные ведомства США были подключены к сети ARPANET, что ознаменовало переход к ее практическому использованию. Агентство ARPA задалось целью проверить, можно ли соединить компьютеры, расположенные в разных местах на значительном расстоянии, при помощи новой технологии, получившей название "*коммутация пакетов*". *Коммутация пакетов* позволяла нескольким пользователям использовать один канал связи, посредством которого пакеты могли передаваться по сети к адресату, где восстанавливалось их исходное содержание. Прежде для работы в сети каждому компьютеру требовалась отдельная линия. Разработки, выполненные NSF, помогли создать высокоскоростную глобальную сеть, доступную для всех образовательных учреждений, государственных служащих, международных исследовательских организаций и т. п. Эта сеть позволила создать магистраль для передачи данных и подключить к ней множество компьютеров, совместно использующих один и тот же канал связи. Данные разбивались на пакеты, которые передавались на другую станцию. Каждому пакету присваивался компьютерный эквивалент места назначения (адрес) и временная метка, что позволяло передавать его в нужный пункт. Когда пакеты достигали адресата (пусть даже и по разным маршрутам), они собирались принимающим компьютером в связное сообщение.

Созданная на основе новой технологии *сеть обеспечила независимую передачу данных между пунктами назначения и дала возможность компьютерам совместно использовать данные, а исследователям — обмениваться электронными сообщениями.* Собственно, *изобретение электронной почты произвело революцию*. До этого передача документов должна была осуществляться при помощи факсов, почтовых курьеров или государственной почты. Электронная почта давала возможность отправлять подробные письма со скоростью и по ценам телефонного звонка.

По мере роста сети ARPANET предприимчивые студенты разработали способ ее использования *для проведения конференций в реальном времени*. Сначала эти конференции имели научную тема-

тику, но скоро они охватили *практически все сферы интересов*, поскольку люди оценили преимущества возможности общаться с сотнями или даже тысячами собеседников по всей стране, познакомившихся друг с другом при помощи электронной связи.

Сегодня эта сеть связывает компьютеры различных типов по всему миру при помощи протокола (стандарта передачи пакетов информации), получившего название TCP/IP (Transmission Control Protocol/Internet Protocol). В конце 1970-х гг. были созданы каналы связи между ARPANET и подобными ей сетями в других странах. Теперь мир был опутан компьютерной "паутиной" (общеизвестное сокращение WWW и означает World Wide Web — всемирная паутина).

В 1980-х гг. эта сеть сетей, получившая общее название "Интернет" (название произошло от термина "internetworking" — "межсетевое взаимодействие"), стала расти с феноменальной скоростью. Тысячи исследовательских организаций и государственных ведомств, колледжей и университетов начали подключать свои компьютеры к этой всемирной сети.

Типовой адрес компьютера, используемого в сети Интернет, выглядит так: **www.name.ru**, где **www** — информация об общепринятом соглашении использования адресатом протокола передачи и приема данных; **name** — (условное имя — название организации, предоставляющей или имеющей свой адрес в сети); **ru** — географическая привязка нахождения компьютера в мировой сети (**ru** — Россия, **com** и **su** — Америка и т. д.).

Каким способом происходит подключение к Интернету? Есть *два способа подключения. Первый,* более простой, — *открыть у провайдера* сервисов Интернета (Internet Service Provider — ISP) *счет для получения доступа по телефонной линии*. ISP может предоставить счет, обеспечивающий связь по протоколам SLIP (Serial Line Internet Protocol) или PPP (Point-to-Point Protocol).

Другой метод подключения к Интернету — *подключение по выделенной линии*. Этот способ более эффективен для больших организаций. Тип выделенной линии и скорость связи зависит от способа использования Интернету в организации и требуемого для этого диапазона. Существует множество типов и скоростей выде-

ленных линий: от линий со скоростью 56 кбит/с до линий ISDN (цифровых сетей с интегрированными сервисами) и систем ретрансляции кадров, а также частичных или полных линий.

Вне зависимости от способа получения доступа к Интернету вам потребуется IP-адрес (IP — Internet Protocol) для учетной записи, обеспечивающей доступ к сети. Этот IP-адрес может выделяться провайдером либо *динамически* (это означает, что IP-адрес может меняться каждый раз при доступе к Интернету), либо *статически* (IP-адрес всегда остается одним и тем же).

На рис. 9.3.3 схематично показана организация пересылки данных в Интернете.

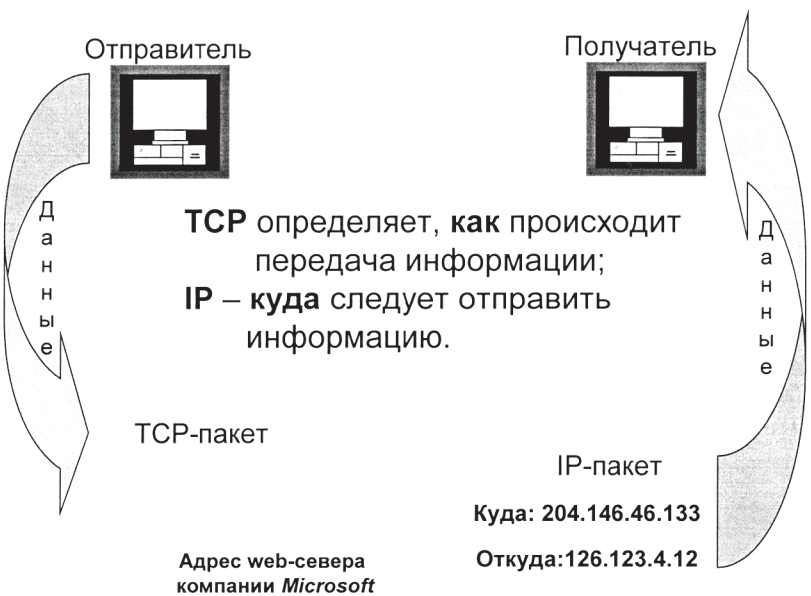


Рис. 9.3.3. Организация передачи данных в Интернете

Рассмотрим некоторые возможности сети Интернет. Существует восемь основных путей ее использования.

1. Электронная почта.

2. Отправка и получение файлов с помощью FTP (File Transfer Protocol).

3. Чтение и посылка текстов в USENET.

4. Поиск информации через GOPHER и WWW (World Wide Web).

5. Удаленное управление — запрос и запуск программ на удаленном компьютере.

6. Интернет-пейджинг с помощью ICQ.

7. Chat-разговор с помощью сети IRC и электронной почты.

8. Видеоконференции и игровые формы работы через Интернет.

Перед более подробным раскрытием приведенных возможностей следует отметить, что развитие современных компьютерных технологий и разрабатываемого программного обеспечения имеет тенденцию сближения указанных возможностей и интеграции их в одном программном продукте. Множество программ, разрабатываемых для этих целей (таких, например, как Internet Explorer, Netscape Communicator и др.), обеспечивающих отдельные функции Интернета, называются *клиентами*. Они удобны в использовании и предоставляют дружественный интерфейс для пользователей Интернета [38].

Электронная почта. Отправка и получение писем остается пока *наиболее популярным видом использования* Интернета. В рамках Интернета для электронной почты существует система LISTSERV, позволяющая создавать группы пользователей с общей групповой адресацией. Таким образом, письмо, направленное на групповой адрес, будет получено всеми членами группы. Например, существует LISTSERV Netterain, объединяющий группу специалистов, обучающих пользованию Интернетом. Они объединились для того, чтобы обменяться идеями или задать вопросы своим коллегам, дать знать, что с ними можно связаться по электронной почте. В случае, если известно, что конкретное лицо или компания имеют адрес в Интернете, но сам адрес не известен, существуют способы узнать его с помощью системы NETFIND. Адрес электронной почты в Интернете часто называют e-mail. Типовой адрес выглядит так:

ignik@orc.ru,

где ignik — определенное пользователем имя при организации электронного почтового ящика у провайдера;

@ — разделительный знак в электронном адресе, называемый "собака";

orc — зарегистрированное имя сервера провайдера в сети Интернет.

Отправка и получение файлов с помощью FTP. При работе возможно использование FTP — одного из самых распространенных протоколов передачи файлов по Интернет. В начале это была терминальная программа с командной строкой, сейчас многие FTP-клиенты отличаются удобным интерфейсом и дополнительными возможностями, например возможностью "докачки" файла после обрыва связи с провайдером.

Чтение и посылка текстов в USENET. USENET — это сеть информационных серверов в Интернете, часто называемой *сетью новостей*. В сети Usenet порядка 500 000 так называемых конференций (по сути, это каталог, куда стекаются со всего мира сообщения на определенную тему). Практически на любую тему в сети отведена своя собственная группа. Каждая отдельная группа сети USENET называется *телеконференцией*. Серверы по всему миру, организованные в отдельную сеть, постоянно обмениваются между собой информацией, в результате происходит естественное обновление новостей. Среди множества телеконференцией USENET имеются конференции, отражающие новости в науке (по отдельным ее областям) и в экономике. *Многие телеконференции русифицированы.*

Поиск информации в Интернете. Пользователь часто ищет информацию в Интернете либо чтобы узнать, имеется ли в мировых информационных ресурсах интересующий его материал по соответствующей теме и получить к данному материалу доступ; либо просто "осматривается" в информационном пространстве. Для работы в этих целях используются так называемые *просмотрщики* — браузеры (от англ. browsing — беспорядочное чтение) — специально ориентированные программы-клиенты.

Все информационные операции в Интернете возможны благодаря действию системы протоколов, основные из которых приведены в табл. 9.3.1. Можно сказать, что эти протоколы обеспечивают совместимость информационных систем, входящих в Интернет, на физическом уровне (к сожалению, об их совместимости на логическом уровне говорить не приходится, что дает повод для критики глобальной системы — с бурным развитием Интернета ее часто стали называть "свалкой информации", "мусорной ямой" и т. п., что, безусловно, верно лишь частично).

Таблица 9.3.1

Основные протоколы Интернета

Транспортные протоколы	TCP — Transmission Control Protocol (протокол управления передачей данных) — управляет передачей данных между компьютерами
Протоколы маршрутизации	IP — Internet Protocol (протокол Интернета) — обеспечивает фактическую передачу данных, обрабатывает адресацию данных, определяет наилучший путь к адресату
Протоколы поддержки сетевого адреса	DNS — Domain Name System (доменная система имен) — обеспечивает определение уникального адреса компьютера
Протоколы прикладных серверов	FTP — File Transfer Protocol (протокол передачи файлов); HTTP — Hyper Text Transfer Protocol (протокол передачи гипертекста) — используются для получения доступа к различным услугам Интернета
Шлюзовые протоколы	EGP — Exterior Gateway Protocol (внешний шлюзовый протокол) — помогает передавать по сети, а также обрабатывать данные для локальных сетей
Почтовые протоколы	POP — Post Office Protocol (протокол приема почты); SMTP — Simple Mail Transfer Protocol (протокол передачи почты)

Значительная часть мировых информационных ресурсов представлена в Интернете. Можно потратить определенное количество времени, просто переходя с одного *сайта* (сервера Интернет) на другой и определяя, какая информация имеется в наличии. Эффект взрыва произвело в свое время появление таких средств уп-

равления поиском информации, как GOPHER и WWW.GOPHER использует систему меню, позволяющую пользователям осуществлять выбор информации. Наиболее развитое сегодня средство для поиска в Интернете — технология WWW. В последнее время часто под понятиями Интернет и web подразумевают одно и то же. Технология web позволяет свободно перемещаться среди информационных ресурсов (от одного документа на одном сервере Интернет к другому документу на другом сервере Интернета) *по ключевым словам в документах*, поддерживающих систему *гипертекста* и протокол HTTP. Переходы в Интернете между документами (web-страницами) осуществляются с использованием гипертекстовых связей, так называемых URL (Universal Resource Locators) — универсальных локаторов ресурсов. Большое число организаций, школ и людей создают собственные элементы WWW, так называемые Home Pages (домашние web-страницы), которые могут иметь гипертекстовые связи с информацией, находящейся на том же компьютере, или которая может быть найдена на любом компьютере в сети INTERNET. Для разработки web-документов используются специальные редакторы и язык разметки гипертекста HTML. *Web-документ отличается от обычного текстового документа возможностью представления в нем текстовой информации на экране компьютера в сочетании с графическими изображениями (в том числе с возможностью анимации изображений)*, а также с *видимыми гипертекстовыми ссылками на другие документы* Интернета.

Интернет настолько велик и полон информационных ресурсов, что основной проблемой, с которой сталкиваются пользователи, является *поиск нужных* им данных. В дополнение к электронной почте, системам WWW и USENET существуют и другие полезные инструменты, которые были созданы специально для помощи путешественникам по "информационному пространству". Наиболее полезной из них является система FTP-клиент, позволяющая среди множества FTP-серверов (гигантских хранилищ архивов программ и документов) отыскивать необходимые архивы и программы.

Следует также сказать, что прогресс в компьютерных технологиях бурно развивает специально ориентированные на поиск документов поисковые системы. Для этих целей в Интернете выделяются *специальные серверы — мета-машины*, обеспечивающие поиск информации по отдельным алгоритмам с явными и неявными критериями (по отдельным словам в документе или по их словосочетаниям). Число таких машин постоянно растет. Наиболее известные из них — мировые серверы Altavista (www.altavista.com), Yahoo (http://www.yahoo.com) и др. Среди русскоязычных наиболее популярны "Апорт!" (http://www.aport.ru), "Созвездие Интернет" (http://www.stars.ru), Яndex (http://www.yandex.ru) и др. Больше того, поисковые серверы объединяются по определенным признакам (по темам и интересам) в свои сети и передают поисковые запросы друг через друга по всему миру.

Удаленное управление. Эта возможность очень полезна, когда при выполнении некоторой работы на отдельном компьютере требуются ресурсы больших систем. Существует несколько различных типов удаленного управления. Некоторые из них работают на основе команд, подаваемых последовательно шаг за шагом (так называемых *ping — пингов*). Таким образом, исполнение запроса удаленного компьютера заключается в том, чтобы некоторая специфическая команда или их последовательность была выполнена на некотором другом компьютере. Более развитые версии запросов сами выбирают систему и компьютер, которые будут к тому моменту в сети свободными. Существует также удаленный вызов процедуры, который позволяет программе запускать подпрограмму на другом компьютере и затем использовать результат ее работы.

Интернет-пейджинг с помощью ICQ. Система радиопейджинга давно известна и широко используется во всем мире. В 1998 г. разработчики израильской компании Mirabilis создали специальное программное обеспечение ICQ (I seek you — "я ищу тебя") и *перенесли возможности вызова абонентов (пейджинг) в сеть* INTERNET. Программа получила название интернет-пейджера. У абонента интернет-пейджинга имеется уникальный ICQ-номер (UIN), по которому его можно вызвать. Стать абонентом сети

достаточно просто — необходимо зарегистрироваться на сервере http://wwp.mirabilis.com или на русскоязычном сервере www.icq.ru. Серверы, поддерживающие ICQ, также часто объединяются в интернет-пейджинговые сети. Сервис позволяет вести переговоры (в русском языке появилось даже новое словообразование "чатиться", от слова chat — "дружеский разговор, беседа") в реальном времени вдвоем или многим пользователям одновременно, обмениваться сообщениями, пересылать файлы и т. д. Любой человек, не имеющий ICQ, может послать другому с ICQ сообщение на его ICQ-пейджер, либо зайдя в сети на адрес http://wwp.mirabilis.com/******, где вместо звездочек его номер, либо послав e-mail на адрес *****@pager.mirabilis.com. Владелец ICQ имеет возможность заблокировать прием сообщений с www-пейджера или с e-mail-express.

Возможность разговаривать с многими людьми с помощью IRC. IRC (Internet relay chat) — это *связка крупных сетей* (Efnet, Dalnet, Undernet и др.), в каждой из которых сотни чатов и десятки тысяч пользователей. Официальный отсчет истории IRC ведется с 1988 г. Именно тогда финский студент Джако, некоторое время поговорив на многолинейных электронных досках объявлений (BBS), задался целью создать *нечто похожее в Internet, но более глобального масштаба.* Тогда и появилась первая сеть IRC — Efnet. Сегодня серверы Internet, поддерживающие IRC, объединены в единую сеть по всему миру.

Каналы в IRC (channels) можно сравнить с комнатами — вы "заходите" на канал, и после этого любая ваша фраза может быть услышана всеми, кто находится на том же канале — вне зависимости от того, что один ваш собеседник живет в Австралии, а другой — в Южной Африке. При необходимости вы можете общаться лично — ваше сообщение увидит только тот, кому вы его послали.

Видеоконференции и игровые формы работы через Интернет. Игровые формы работы (компьютерные игры, в частности) занимают много времени пользователей. Именно компьютерные игры подталкивают разработчиков на создание и внедрение новых передовых компьютерных технологий и аппаратного обеспечения, а в некоторых сферах деятельности человека, например в эконо-

мике, игровые формы являются действенным способом апробации результатов — моделирования последствий принятых управленческих решений. На игровых принципах работы проводятся (моделируются) различные деловые игры. Играть можно против компьютера, против одного конкурента (человека) с помощью модема, против многих конкурентов с помощью локальных сетей или через Интернет. Для реализации этих форм игры необходима разработка специализированного программного обеспечения. Существует много серверов, которые предназначеные исключительно для компьютерных игр, таких как: Quake, Quake II, Team Fortress, Warcraft II, Starcraft и множество других. Имеются сведения, что сейчас разрабатываются и испытываются специализированные серверы, предназначенные исключительно для моделирования и обслуживания игровых форм работы в экономике. Для того чтобы качество игры было приемлемым, необходимо обеспечить стабильную и высокоскоростную связь с Интернетом.

Немалую пользу приносит организация между пользователями *системы видеоконференций*. Применение данного вида сервиса наиболее удачно для проведения оперативных совещаний и сбора докладов от подчиненных, находящихся на значительном расстоянии друг от друга. Работа в данном режиме позволяет пользователям видеть друг друга и даже демонстрировать друг другу различные предметы, образцы, документы. Весьма перспективным оказалось применение видеоконференций в медицине — при проведении консультаций, консилиумов и даже операций. Совершенно очевидна перспектива их использования и в экономике.

Объединение системы видеоконференций с одновременным проведением игровых форм работы предоставляет для пользователей новые возможности моделирования, обмена и уточнения информации.

9.3.3. Корпоративная сеть ИНТРАНЕТ

Перечень услуг сети Интернет достаточно широк и разнообразен [43]. Желание разработчиков *перенести данные возможности на большие внутриведомственные сети, а также обеспечить соответствующий режим защиты внутриведомственной информа-*

ции вызвало необходимость создания новой сети, получившей название INTRANET (ИНТРАНЕТ). Практика создания подобных сетей раскрыла главную их особенность: объединить возможность работы пользователей над общими проектами с высоким уровнем сервиса Интернета. Итак, *сеть* ИНТРАНЕТ — *это та же сеть* Интернет, но *организованная и работающая в рамках отдельной организации (корпорации*). Именно поэтому сеть ИНТРАНЕТ и *называют современной корпоративной сетью*. Существуют различные типы сервисов (перечень услуг), которые могут обеспечиваться в сети ИНТРАНЕТ. Рассмотрим данные виды сервиса более подробнее, тем более что некоторые из них по своим возможностям шире, чем услуги Интернета [46, 47].

Почтовые сервисы. Существующая сетевая среда в рамках корпорации, как правило, уже располагает средствами пересылки сообщений между отдельными компьютерами и рабочими группами внутри организации, например, используя возможности ЛВС. Если это не так, то ИНТРАНЕТ предоставляет возможность организовать для пользователей корпорации функционирование данного сервиса. Для этого необходимо выбрать почтовый пакет, поддерживающий электронную почту на основе интернет-навигатора (клиента). При выборе почтового пакета может возникнуть необходимость в установке дополнительного протокола SMTP для обмена сообщениями между внутренней сетью организации и адресатами Интернета. Для поддержания этого сервиса потребуется выделенный либо совместно используемый сервер организации.

Файловые сервисы. При наличии в сети корпорации выделенного файл-сервера, появляется возможность организации файлового сервиса (приема-передачи файлов между пользователями сети ИНТРАНЕТ). Однако для работы ИНТРАНЕТ на уровне возможностей сети Интернет потребуется установка на серверы корпорации дополнительного программного обеспечения (ПО) для предоставления доступа к файлам на базе протокола FTP. Это может потребовать инсталляции на существующие файловые серверы и (или) рабочие станции протокола TCP/IP и набора необходимых программ.

Web-сервисы. World Wide Web — новейший вид сервиса, обеспечиваемый в рамках ИНТРАНЕТ. С целью использования данного сервиса внутри большой организации и началось развитие ИНТРАНЕТ-сетей. Web-сервис может обеспечиваться в корпорации или отдельным web-сервером, или для этого будет использоваться уже существующий файловый сервер. Для эффективной работы необходимо ПО web-сервера. Также может возникнуть необходимость в расширении памяти или дискового пространства сервера. Кроме того, следует учитывать, что на работу и производительность сервера могут оказывать влияние другие программы или загружаемые модули рабочих станций сети. Любой аварийно завершающийся процесс остановит работу как файловых серверов, так и ИНТРАНЕТ в целом.

Аудиосервис. Одним из преимуществ сетей ИНТРАНЕТ является *надежность и доступность сетевого диапазона*. Это дает возможность предоставления ИНТРАНЕТ-услуг, которые ранее были бы недоступны в Интернет. Одной из таких услуг является передача аудиоданных по сети. В число аудиоуслуг может входить передача музыки, клиентских копий рекламных сообщений и даже выдержек из корпоративных заявлений или речей. Создав в ИНТРАНЕТ-сети web-страницу новостей, можно разместить на ней аудиозапись сообщений руководителя (начальника), ставящего и уточняющего задачу подчиненным, или другую информацию.

Аудиосервис определяет выбор оборудования и операционной системы, с которой он будет работать. Поэтому, если использование аудиосервиса является важной качественной характеристикой ИНТРАНЕТ-сети, может возникнуть необходимость в пересмотре выбранного ИНТРАНЕТ-сервера или включения в сеть дополнительного, специально выделенного сервера для установки на нем аудиосервиса.

Видеосервис. Для организации в сети ИНТРАНЕТ качественного видеосервиса необходимо *выделение специального видеосервера*. Видеосервис не ограничен диапазоном ИНТРАНЕТа. Видеосерверы могут обеспечивать несколько видеопотоков, что позволит одновременно отображать несколько видеоклипов на одной web-странице. Видеоклипы могут содержать сведения об отдель-

ных аспектах деятельности организации, новых достижениях, техническую информацию или предназначаться для обучения персонала и т. д.

Видеосерверы требуют большей мощности, чем традиционные web-серверы, и должны устанавливаться на *выделенной машине*. Объем дискового пространства видеосервера тоже является важным фактором, поскольку видеоклипы сами по себе являются файлами большого объема. Как и аудиосервер, видеосервер определяет аппаратную платформу и операционную систему, с которой он будет работать. Если видеосервис также является важной частью сети ИНТРАНЕТ, как и в случае аудиосервера, то может возникнуть *необходимость добавления к сети специально выделенного видеосервера*.

На основе всего сказанного следует сказать еще об одной из особенностей ИНТРАНЕТа, а именно то, что данная сеть позволяет без особых на то усилий объединить ПК, изготовленные на различных аппаратных платформах и работающие на различных операционных системах. Последнее обстоятельство в сочетании с перечисленными достоинствами ИНТРАНЕТ-сетей определяет их перспективность для массового использования в различных (в том числе — экономических) организациях.

9.3.4. Сети электронных досок объявлений

Электронная доска объявлений (Bulletin Board System, в дальнейшем BBS) физически представляет *собой достаточно мощный ПК со специальным программным обеспечением, позволяющим удаленному пользователю дистанционно обращаться к системе и во время связи (в режиме on-line) знакомиться с электронными объявлениями*. Однако сегодня BBS — это уже не простая система *обмена сообщениями*, как это было в 1980-х гг., когда и возник этот английский термин. Современная *BBS является мощным телекоммуникационным узлом, способным предоставить своим пользователям широкий спектр услуг, в котором сами по себе электронные объявления зачастую играют второстепенную роль* [47]. Еще большие возможности открываются у пользователей при объединении

BBS в единую сеть по тематическому, организационному, территориальному или иному признаку.

Предоставляемая пользователю *информация на электронных досках объявлений строго структуризирована*. Используемое на BBS программное обеспечение позволяет вместе с тем осуществить оперативный поиск объявлений по ключевым словам, фразам, темам сообщений или их комбинации.

Как правило, узел BBS содержит большое количество полезных программных продуктов самой разной направленности, логически разбитых по тематике. Работая в системе в режиме online, возможно ознакомление со списком предлагаемых файлов. Пользователь BBS, в соответствии с установленным для него уровнем доступа на станцию, может "перекачать" (download) на свой компьютер заинтересовавшую его информацию: от отдельных сообщений до необходимых пользователю файлов и программ или "закачать" (upload) некоторую информацию. Помимо этого, на BBS доступны *территории личной и публичной переписки между пользователями данной станции*. Таким образом, на BBS можно размещать общие сообщения, рекламу, объявления о розыске ПО, анонимные послания и другую информацию.

За нарушение устанавливаемых на BBS правил по воле *Системного Оператора* (в дальнейшем СисОп) — отвечающего за работу станции человека — можно лишиться дальнейшего доступа к данной BBS. Каждый зарегистрированный на BBS пользователь получает строго ограниченный СисОп *суточный период времени для реализации своих намерений и желаний*. Этого иногда бывает недостаточно даже для того, чтобы принять список доступных на данной BBS файлов (так называемый Filelist). При удовлетворении пользователем определенных требований СисОп может *повысить уровень доступа* (Access Level) пользователя к данной BBS.

Существует множество классификаций узлов BBS. Они бывают любительскими или профессиональными, строго ориентированными на определенную тему или совокупность тем, коммерческими и бесплатными, 24-часовыми и с ограниченным временем работы (как правило, работающие ночью; днем же —

это обычный голосовой телефон), однолинейные и многолинейные и т. д.

К профессиональным станциям относятся крупные сетевые серверы или целые сети BBS, подобные Elvis, Izhma, Kiae, Simte, Chci и др., а также небольшие узкоспециализированные станции или сети. Особенностью сетей BBS является то, что каждый узел в сети под общим ее названием имеет свой порядковый номер и, как правило, некоторую часть единого сервиса, характерного для всей сети в целом. Отдельные узлы сети могут иметь *шлюзы для выхода на другие сети* или *отдельные узлы* BBS. Главные отличия организованных таким образом сетей заключаются в предоставлении доступа по использованию информационных ресурсов за абонентскую плату, 24-часовой график работы, большой выбор предлагаемого ПО, совместимость данных BBS с внутренними ЛВС организаций, другой сервис. С бурным развитием компьютерных технологий и проникновением Интернета во все сферы общества подобный сервис появляется и на серверах всемирной информационной сети.

9.3.5. Компьютерные сети на основе FTN-технологий

Наряду с достаточно широким распространением Интернета и ИНТРАНЕТа большой популярностью пользуются сети, в основе которых при использовании информационных ресурсов лежит непосредственный доступ к информации по коммутируемым — чаще всего телефонным — каналам связи. Более того, *компьютерные сети на основе так называемых FTN-технологий были своего рода родителями идей развития современных сетей* Интернет и ИНТРАНЕТ. В свое время широкое распространение персональных компьютеров и быстрое внедрение новых недорогих средств связи (модемов) сделало возможной передачу данных по телефонным линиям напрямую от одного компьютера к другому, без промежуточных звеньев в виде больших машин или дополнительных технических устройств; при этом удаленность отправителя от адресата имела малое (или вообще не имела) значение. Каждый пользователь персонального компьютера получил возможность стать профессионалом на собственном компьютере и сам предос-

тавлять другим информационные услуги. Так создавались *компьютерные сети передачи данных с добровольным распределением обязанностей по обмену информацией*. Первая такая сеть появилась всего через три года после выхода на рынок первых IBM PC. Это была сеть Fidonet, задуманная именно для *объединения персональных компьютеров, используемых в качестве независимых телекоммуникационных систем.*

Неформальный дух сети проявился уже в ее названии: создатель сети Том Дженнингс назвал сеть в честь своей собаки Fido, изображение которой стало символом Fidonet. С самого начала сеть носила и носит *любительский и некоммерческий характер*. Участники сети тратят свои собственные деньги и время, чтобы она работала в интересах всех ее пользователей.

Технология Fidonet оказалась столь популярной, что на ее основе по всему миру созданы и функционируют около тысячи любительских и коммерческих телекоммуникационных сетей, совместимых с Fidonet по программному обеспечению, многие из них имеют *шлюзы* (ворота, каналы доступа) в Fidonet. *Около десятка подобных сетей ориентированы на экономические, научно-исследовательские и учебные цели*. В сети Fidonet также существует большое количество *шлюзов с сетью* Интернет. Еще на самом начальном этапе развития в структуру адресов Fidonet была заложена *иерархичность и многоуровневость*, что позволило в дальнейшем разработать *принципы децентрализованного управления и поддержки развития сети.*

С момента возникновения Fidonet ее технологические стандарты разрабатывались самими членами сети. Вначале это были просто дополнительные возможности, вводимые создателями первых программ для Fidonet, однако со временем рост сети вызвал *необходимость более жесткой стандартизации*. Также постоянно росло количество предлагаемых членами Fidonet изменений и добавлений к технологии Fidonet. Для решения возникших проблем был создан *Комитет по стандартам технологии Fidonet (Fidonet Technology Standards Comittee, FTSC или Fidonet Technology Network, FTN)*, который за время своего существования разработал на основе многочисленных предложений членов сети несколько

десятков стандартов различных компонентов технологии Fidonet. Разработка новых стандартов продолжается и в настоящее время.

Изначально сеть Fidonet предназначалась *для обмена личной электронной почтой между узлами, по сути, — между операторами узлов.* Вскоре была разработана *технология эхоконференций* (аналог телеконференции в сети Интернет). Данная технология позволила впервые объединить почтовые ящики разрозненных BBS или их сетей и создать общую систему электронного обмена информацией. Технология эхоконференций дала мощный толчок развитию как сети Fidonet, так и самих BBS — разработчики программного обеспечения BBS и почтовых программ Fidonet стали обеспечивать в своих продуктах возможность интеграции BBS и узлов Fidonet на одном компьютере, и Fidonet стала похожа на "сеть BBS": на большей части узлов Fidonet были развернуты BBS, а большинство BBS стремились получить и получали адрес в сети Fidonet. В настоящее время порядка 80% узлов Fidonet предоставляют доступ к своим ресурсам не только другим узлам сети в автоматическом режиме, но и пользователям BBS в интерактивном режиме. Однако Fidonet *была и остается именно сетью для автоматического обмена данными*, и большинство крупных узлов Fidonet, через которые проходят основные маршруты распространения почты, *не поддерживают входящие звонки пользователей* BBS.

Первое, что необходимо для того, чтобы достаточное количество телекоммуникационных узлов, объединенных в сеть, могли обмениваться информацией — это *наличие в сети определенной структуры*. В Fidonet *структура определяется в первую очередь сетевым адресом* узла. Адрес узла в Fidonet (и любой FTN-совместимой сети) имеет числовую форму и строится по схеме: зона: сеть_или_регион/узел.пойнт.

Узел (Node) является наименьшей структурной единицей Fidonet; в то же время это основная единица Fidonet.

Пойнт (Point) — пользователь FTN-сети, прикрепленный к узлу.

Сеть (Network) — это объединение узлов некой локальной географической области, обычно определяемое областью с удобной (т. е. бесплатной) телефонной связью между узлами сети.

Регион (Region) — это определенная достаточно крупная географическая область, включающая узлы, которые могут быть объединены либо не объединены в сети; типичный регион содержит множество узлов, объединенных в сети, и несколько независимых узлов, не являющихся частью какой-либо сети. В адрес сети, как правило, входит как составная часть адрес региона, которому принадлежит эта сеть.

Зона (Zone) — это наиболее крупная структурная единица Fidonet, большая географическая область, включающая множество регионов и охватывающая одну или несколько стран и (или) континентов. Fidonet *насчитывает шесть зон*: 1 — Северная Америка; 2 — Европа и территория бывшего СССР; 3 — Австралия и Океания; 4 — Южная Америка; 5 — Африка; 6 — Азия.

Таким образом, сетевая принадлежность конкретного узла, например 2:5020/113, определяется как узел 113 сети 5020 региона 50 зоны 2 Fidonet. Географическое местоположение узла можно также определить из сетевого адреса: 2 — Европа, 50 — Россия, 20 — Москва.

Итак, структуру сети Fidonet можно представить на рис. 9.3.4.

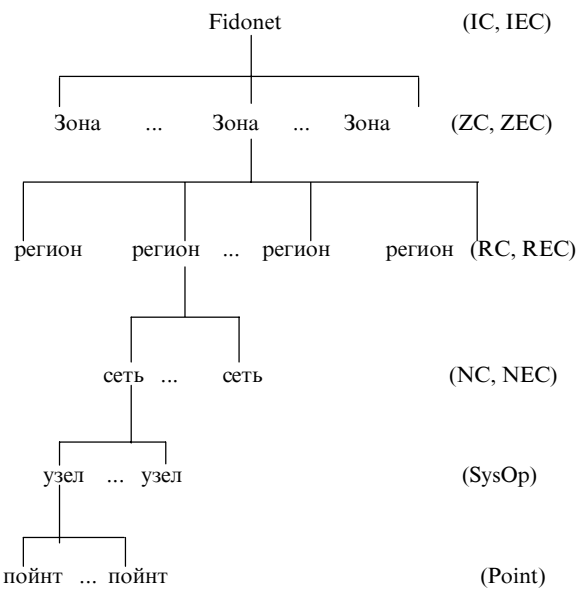


Рис. 9.3.4. Структура типовой FTN совместимой сети

В правой части рисунка в соответствии с иерархическим уровнем приведено название должности, выборной в сети ее членами для всей сети — *Интернациональный координатор* (IC) *и интернациональный координатор эхоконференций* (IEC); для зоны — *зональный координатор* (ZC) и *зональный координатор эхоконференций* (ZEC); далее — национальный координатор (NC) и национальный координатор эхоконференций (NEC); системный оператор (SysOp) и Point.

По мере развития сети и самих модемов возникли и развились *файловые конференции* (File-Echo Conference), где *в качестве элементарных единиц выступали не сообщения, а файлы*. Тем самым член сети, разработавший, по его мнению, гениальную программу, мог разослать ее посредством файловой конференции всем на нее подписанным пользователям. Правда, ежедневный поток (Traffic) в таких конференциях составляет от одного до нескольких мегабайтов в день, но существующие на данный момент мощности модемов позволяют поддерживать их без особых на то усилий. Число же официально существующих на сегодняшний день файловых конференций в сети Fido *превышает 2000*. Для каждой из них разрабатываются *определенные правила пользования* конференцией, а *пользователи сети голосованием выбирают модератора — ответственного за порядок в файловой конференции (эхоконференции)*. *Основополагающим принципом* Fidonet является *обеспечение возможности передачи данных напрямую от любого узла Fidonet к любому другому узлу*. Это обеспечивается распространением среди всех узлов сети списка-справочника узлов, или *нодлиста* (Nodelist). Нодлист представляет собой структурированное текущее описание узлов Fidonet и по сути дела определяет саму сеть. Актуальность нодлиста поддерживается выпуском еженедельных файлов изменений и добавлений с рассылкой их по сети.

С расширением Fidonet и ростом ее популярности появилось достаточно большое количество людей, стремящихся к общению в Fidonet, желающих отправлять и принимать почту и файлы в автоматическом режиме, а не через BBS, но не имеющих возможности поддерживать узел Fidonet. Согласно первоначальным стандартам Fidonet для таких пользователей на узлах, к которым они подключались, образовывались *"псевдо-сети"* (fakenets) с произ-

вольным номером сети; при отправке писем этих пользователей с узла Fidonet в них подставлялся реальный Fidonet-адрес узла-отправителя. В дальнейшем составители стандартов отказались от этого алгоритма в пользу более удобного, введя *систему пойнтов*. Пойнт, посылающий почту через определенный узел, пользуется адресом узла, к которому через точку добавлен номер пойнта, например 2:5020/113.1.

Пойнт не обязан соблюдать технические процедуры, установленные для узла Fidonet. Фактически пойнт представляет собой пользователя BBS, наделенного сетевым адресом и использующего Fidonet-совместимое программное обеспечение для работы с почтой. В Fidonet ведутся и распространяются списки пойнтов отдельных сетей в формате, аналогичном нодлисту.

Появление Fidonet в России весной 1990 г. было вполне в духе сети — первой Fidonet-совместимой почтовой системой на территории России был пойнт одного из польских узлов, расположенный в Новосибирске (!). Благодаря тому что в структуре адресов Fidonet заранее было зарезервировано адресное пространство для России, на всей территории страны сеть смогла развиваться в большей мере как единое целое.

Можно с уверенностью сказать, что почти за десять лет развития Fidonet в России стала не просто сетью электронной почты, а крупнейшим явлением, объединяющим сотни тысяч человек во всех концах страны. Российская Fidonet предлагает пользователям русскоязычную среду для общения по самому широкому кругу вопросов — от сугубо технических до узконаправленных тем, включая экономические.

В заключение данного пункта отметим, что дальнейшее развитие автоматизации информационного обеспечения деятельности должностных лиц в любой профессиональной сфере — в том числе и в экономической — немыслимо без применения сетевых технологий — от локальных до глобальных.

Вопросы для самопроверки

1. Дайте определения понятий “сеть”, “вычислительная сеть”, “информационная сеть”.

2. Что дает создание компьютерных сетей пользователям?

3. Какую сеть называют локальной?

4. Каковы ее достоинства и недостатки?

5. Что сегодня называют Интернетом?

6. Вспомните основные пути использования возможностей Интернета.

7. Поясните назначение основных сетевых протоколов.

8. Каким образом организуется передача данных в Интернете?

9. Почему иногда Интернет называют “информационной свалкой”?

10. Поясните идеи организации сетей Интранет, BBS, FTN.

10. Основы построения и использования интеллектуальных информационных систем

10.1. Методологические основы теории искусственного интеллекта

Термин “искусственный интеллект” относится к группе терминов и понятий, получивших весьма широкое распространение, в том числе и у неспециалистов. Не пытаясь сразу четко и полно определить это понятие, заметим, что большинство людей трактуют искусственный интеллект как *сравнительно новое научно-техническое направление, с которым связывают надежды на резкое увеличение функциональных возможностей технических объектов, в частности вычислительных систем, используемых в качестве средств автоматизации различных сфер профессиональной деятельности человека: управления, проектирования, производства, обучения, индустрии обслуживания и развлечений и т. п.*

Для того чтобы методически верно подойти к определению основных понятий теории искусственного интеллекта, сначала кратко рассмотрим основные этапы его истории.

10.1.1. Краткая историческая справка

История искусственного интеллекта насчитывает всего *несколько десятилетий*, хотя создать “думающую машину” мечтали многие поколения людей. Первый международный конгресс по искусственному интеллекту состоялся в США в 1969 г., ему предшествовали исследования в разных странах и направлениях:

— в 1950-е гг. были начаты работы по *машинному переводу* с одного языка на другой;

— в 1957 г. Ф. Розенблатом было предложено *устройство для распознавания образов* — ***персептрон***, положившее начало разработке большого числа других устройств подобного типа (в том числе и в СССР — например, в Риге Л. Гореликом);

— в 1959 г. Г. Саймон и А. Ньюэлл разработали программу GPS (General Problem Solving programme) — универсальный *решатель*, предназначенный для разрешения различных задач из самых разнообразных областей [12], и др.

Как явствует из перечисленных примеров, первоначально на системы, связанные с искусственным интеллектом, пытались возложить различные, но весьма универсальные задачи.

Однако надежды, порожденные первыми успехами в данной области, полностью не оправдались. Задача машинного перевода оказалась гораздо сложнее, чем предполагалось, и ее реализация, прогнозировавшаяся на 1960-е гг., отодвинулась на два десятилетия (кстати, гораздо большее распространение получили автоматизированные словари, позволяющие получить из исходного текста так называемый подстрочный перевод, а не "переводчики" в полном смысле этого слова). Универсальный решатель задач, довольно успешно доказывавший достаточно простые логические теоремы, оказался крайне неэффективным при решении других задач, в частности при многочисленных попытках автоматизировать игру в шахматы. Не удавалось также достаточно эффективно распознавать реальные изображения устройствами персептронного типа.

Вместе с тем, несмотря на неудачи, этот этап имел большое значение для искусственного интеллекта с точки зрения двух обстоятельств:

• во-первых, он *показал возможности использования ЭВМ в автоматическом режиме* при решении задач, ранее решаемых только человеком;

• во-вторых, на этом этапе были *отработаны различные способы и приемы решения интеллектуальных задач*, хотя строгой теории искусственного интеллекта еще не было.

Работы по искусственному интеллекту продолжались, разрабатывались глубинные теоретические вопросы и их программная

реализация. В частности, многие исследователи продолжили разработку алгоритмов и программ шахматной игры, завершившейся в 1967 г. первым чемпионатом мира по шахматам среди ЭВМ (его выиграла советская программа "Каисса", разработанная в Институте проблем управления АН СССР). Отметим, что до последнего времени продолжаются попытки решить спор о том, способна ли программа на ЭВМ переиграть человека (иногда такой процесс называют соревнованиями "кремниевого" и "белкового" шахматистов). В частности, чемпион мира Г. Каспаров неоднократно играл с шахматными программами, разработанными в разных странах (США, Германии), с переменным успехом. Многие специалисты связывают успех или неудачу человека в игре с машиной с регламентом проведения матча: при проведении блицпартий человек, действующий по интуиции, имеет больше шансов на выигрыш; при ограничении времени на партию пятью или двадцатью пятью минутами ("быстрые шахматы") возможности ЭВМ по просчету вариантов в ряде случаев превышают человеческие; сложнее отдать преимущество одной из сторон при классической игре.

Новое развитие работы по искусственному интеллекту получили в 1980-е гг. Теоретический задел, созданный на первом этапе развития интеллектуальных систем при решении достаточно "мелких" задач (машинные игры в шашки, шахматы; сочинение стихов и музыки; перевод и т. п.), привел к важным практическим результатам — таким, как создание *экспертных систем, интеллектуальных расчетно-логических и информационно-поисковых систем, интеллектуальных пакетов прикладных программ* и т. д. Кроме того, практические успехи в создании систем искусственного интеллекта вызвали к жизни новые проекты, в частности проекты разработки ЭВМ следующего (по наиболее распространенной классификации 5-го) поколения, которые должны уметь общаться с пользователем на естественном (или близком к нему) языке; решать не вполне структурированные задачи; давать разумные советы по широкому кругу проблем и т. д.

Вместе с тем специалисты продолжают обсуждать многие основополагающие вопросы, например, что такое искусственный

интеллект, искусственный разум? в чем их отличия; что является предметом теории искусственного интеллекта. И т. п.

10.1.2. Основные понятия и определения теории интеллектуальных информационных систем

Следует отметить, что строгого (формального, научного) определения понятия "естественный интеллект", вообще говоря, не существует. В силу этого еще труднее определить понятие "искусственный интеллект". Для того чтобы решить эту задачу, необходимо уяснить значение таких терминов, как *интеллект; психика; сознание; разум*.

Интеллект. Различают формулировки данного понятия по нескольким направлениям [21, 22]:

- философскую;
- биологическую;
- психологическую.

В *философии* под интеллектом понимают познание, понимание, рассудочную способность к абстрактно-аналитическому расчленению (Г. Гегель), способность к образованию понятий (И. Кант).

В *психологии* под интеллектом понимают характеристику умственного развития индивидуума, определяющую его способность целенаправленно действовать, рационально мыслить и эффективно взаимодействовать с окружающим миром.

В *биологии* под интеллектом понимают способность адекватно реагировать (принимать решения) в ответ на изменение окружающей обстановки.

Важно отметить: *интеллект — это свойство отдельного субъекта*. В частности, интеллектом может обладать не только человек, но и *любой объект*, обладающий указанными выше качествами — способностью к образованию понятий, абстрактно-аналитическому мышлению, целенаправленному действию.

Разум. В отличие от интеллекта *разум — категория сугубо человеческая*, опирающаяся на сознание как высшую форму психологической деятельности. Принципиальным моментом в опре-

делении разума, так же как и сознания, является их общественный, социальный характер, поскольку и то и другое понятие сформировались в результате совместной человеческой деятельности.

Часто используют совместно понятия *рассудок* и *разум*. Интересно, что в античной философии считалось, что если рассудок — способность рассуждения — познает все относительное, земное и конечное, то разум, сущность которого состоит в целеполагании, открывает абсолютное, божественное и бесконечное. В настоящее время с рассудком связывают способность строго: 1) оперировать понятиями; 2) правильно классифицировать факты и явления; 3) приводить знания в определенную систему. Опираясь на рассудок, разум выступает как творческая познавательная деятельность, раскрывающая сущность действительности. Посредством разума мышление синтезирует результаты познания, создает новые идеи, выходящие за пределы сложившихся систем знания.

Сознание. Это понятие также трактуется различными науками неоднозначно.

С точки зрения *философии* сознание — свойство высокоорганизованной материи — мозга, выступающее как осознанное бытие, субъективный образ объективного мира, субъективная реальность.

При *социологическом подходе* сознание рассматривается прежде всего как отображение в духовной жизни людей интересов и представлений различных социальных групп, классов, наций, общества в целом.

В *психологии* сознание трактуется как особый, высший уровень организации психической жизни субъекта, выделяющего себя из окружающей действительности, отражающего эту действительность в форме психических образов, которые служат регуляторами целенаправленной деятельности [26]. Важнейшей функцией сознания является мысленное построение действий и предвидение их последствий, контроль и управление поведением личности, ее способность отдавать себе отчет в том, что происходит как в окружающем, так и в собственном духовном мире.

Психика. Психика — это свойство высокоорганизованной материи — мозга, являющееся особой формой отражения действительности и включающее такие понятия, как ощущение, восприятие, память, чувства, воля, мышление и др. Отметим, что мышление и память, которыми обычно характеризуют интеллект, входят в понятие психики составными частями.

В психике выделяют две компоненты: *чувственную* (ощущения, восприятие, эмоции) и *рациональную*, мыслительную (интеллект, мышление). Другие составляющие психики — память и волю — можно разделить на память *чувств* и память *мыслей*; волю *чувств* и волю *мыслей* (инстинкты и долг перед собой и обществом соответственно).

Например, можно помнить, как берется сложный интеграл (память мыслей), а можно помнить ощущение напряжения и усталости при изучении способа его взятия (память чувств), когда воля чувств (инстинкт самосохранения, желание отдохнуть) боролись с волей мыслей (сознанием необходимости изучения этого способа).

Перечисленные понятия обычно подразделяют на две пары (см. рис. 10.1.1):

- *психика и интеллект* как ее составляющая;
- *сознание и разум* как его составляющая,

причем интеллект и разум — рассудочные, мыслительные составляющие соответственно психики и сознания.

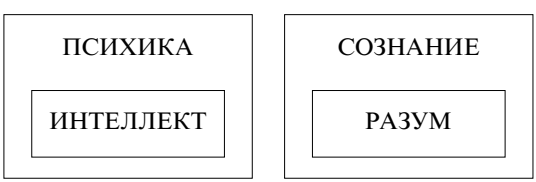


Рис. 10.1.1. Соотношение психики и интеллекта; сознания и разума

Основное отличие второй пары от первой состоит в том, что она образовалась в результате социальной, общественной деятельности людей, и поэтому социальная компонента — неотъемлемая

и существенная черта сознания и разума (классическим примером может служить психика Маугли и психика "нормальных" детей).

Отсюда следует очень важный вывод*: принципиально невозможно моделировать сознание и разум во всей полноте,* так как для этого пришлось бы "моделировать не только человека", но и всю систему его социально-общественных отношений. В то же время моделировать *интеллект как одну из компонент психики отдельных индивидуумов вполне возможно*, хотя и очень сложно.

К этому выводу "примыкает" еще один: *искусственный интеллект — это модель рациональной, мыслительной составляющей психики*. Не моделируются эмоции, ощущения, воля, память чувств и т. п. Машинное сочинение стихов и музыки — это моделирование лишь логической компоненты психической деятельности, сопровождающей эти виды творчества (соблюдение рифмы, размера, законов композиции, гармонии и т. п.). Именно с этим связано неудовлетворительное для большинства людей качество машинных "сочинений".

Учитывая сказанное, можно заключить, что понятие "искусственный интеллект" объединяет *три других* [21, 22]:

— *искусственный бессловесный интеллект* — модель компоненты психики живых существ, отражающая их способность принимать решения, изменять поведение и т. д. на уровне инстинктов, не имеющих словесного выражения (самосохранение, размножение, приспособление и т. п.);

— *искусственный словесный интеллект* — модель рациональной компоненты психической деятельности человека без учета ее социального содержания;

— *искусственный разум* — искусственный словесный интеллект, дополненный социальной компонентой.

В дальнейшем, если не будет специальных оговорок, *под искусственным интеллектом будем понимать искусственный словесный интеллект.*

Приведенные определения основаны на теоретических рассуждениях и в силу этого носят достаточно общий характер.

Существуют по крайней мере *три подхода* к определению этого понятия, носящие гораздо большую *практическую направленность* (рис. 10.1.2).

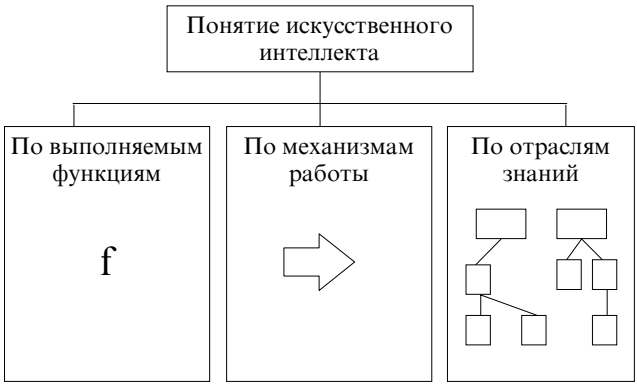


Рис. 10.1.2. Подходы к определению понятия "искусственный интеллект"

Достаточно полным определением понятия "искусственный интеллект" первого типа является следующее [22]: *искусственный интеллект — это область исследований, в рамках которых разрабатываются модели и методы решения задач, традиционно считавшихся интеллектуальными и не поддающимися формализации и автоматизации.*

Применительно к данному определению является справедливым суждение, что интеллектуальной может считаться такая искусственно созданная система, для которой выполняется *тест Тьюринга*, состоящий в следующем: "Испытатель через посредника общается с невидимым для него собеседником — человеком или системой. Интеллектуальной может считаться та система, которую испытатель в процессе такого общения *не может* отличить от человека" [54] (рис. 10.1.3).

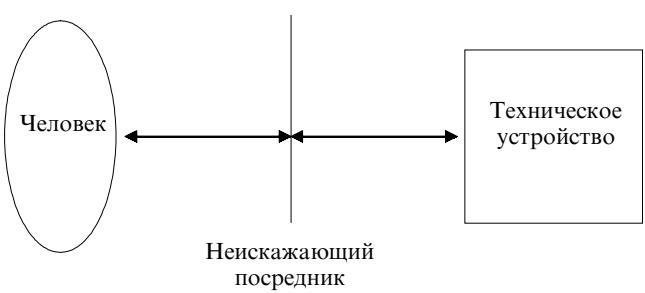


Рис. 10.1.3. Схема проведения теста Тьюринга

В качестве другого определения, достаточно точно отражающего характер второго подхода, может рассматриваться следующее: *искусственный интеллект — это область исследований, в которой изучаются системы, строящие результирующий вывод для задач с неизвестным алгоритмом решения на основе неформализованной исходной информации, использующие технологии символьного программирования и средства вычислительной техники со специальной (не фон неймановской) архитектурой* [22, 54].

Наконец, наиболее цитируемым определением третьего типа является следующее: *искусственный интеллект — это область знаний, которая находит применение при решении задач, связанных с обработкой информации на естественном языке, автоматизацией программирования, управлением роботами, машинным зрением, автоматическим доказательством теорем, разумными машинами извлечения информации и т. д.* [54].

Можно рассмотреть и такое — в определенной степени обобщающее — определение: *искусственный интеллект — научная дисциплина, задачей которой является разработка математических описаний функций человеческого (словесного) интеллекта с целью аппаратурной, программной и технической реализации этих описаний средствами вычислительной техники* [54].

В заключение рассмотрения данного подпункта отметим, что в последние годы многие специалисты согласились, что дискуссия по вопросу об определении самого термина "искусственный интеллект" приобрела схоластический характер, не дает конструктивных результатов теории и практике и может быть бесконечной. Поэтому вместо термина "искусственный интеллект" предлагается использовать другой — *"новая информационная технология решения инженерных задач",* что подчеркивает приоритетную *роль поиска, анализа и синтеза информации* в системах искусственного интеллекта.

10.1.3. Классификация интеллектуальных информационных систем

Рассмотрим классификацию этих систем, имея в виду, что, несмотря на значительное число попыток провести такую класси-

фикацию (см., например, [21, 22, 26, 27, 41]), ни одна из них, на наш взгляд, не является совершенной. На рис. 10.1.4 представлена классификация систем искусственного интеллекта, полученная путем сопоставления и обобщения известных классификаций этих систем.

На рисунке обозначены: СОН — системы общего назначения; СС — специализированные системы.

Наиболее широкое распространение на практике в настоящее время получили *системы искусственного интеллекта, основанные на знаниях*. Понятие "знания" для этих систем имеет принципиальное значение. Под *"знанием" в системах искусственного интеллекта понимается информация о предметной области, представленная определенным образом и используемая в процессе логического вывода*. По своему содержанию данная информация является некоторым набором суждений и умозаключений, описывающих состояние и механизмы (логику) функционирования в выбранной, как правило, весьма ограниченной предметной области. Указанные суждения и умозаключения высказываются экспертом (специалистом) в этой области либо формулируются в результате анализа литературы по данному предметному направлению.

Способы получения и представления знаний в интересах проектирования систем искусственного интеллекта в настоящее время составляют предмет сравнительно нового научного направления — инженерии знаний.

Форма представления знаний имеет отличие от формы представления данных. Обычно (см., например, [53]) *под данными в АИС понимаются факты и идеи, представленные в формализованном виде, позволяющие (лишь) передавать, хранить или обрабатывать эти факты и идеи при помощи некоторого процесса.* В отличие от данных знания предполагают сосредоточение не только фактов и идей в указанном выше смысле (так называемых первичных данных), но и дополнительных данных, которые описывают (интерпретируют) первичные данные с точки зрения следующих составляющих: того, что собой представляют эти данные, какие между ними имеются связи, какие действия с ними и каким образом могут выполняться и т. п.

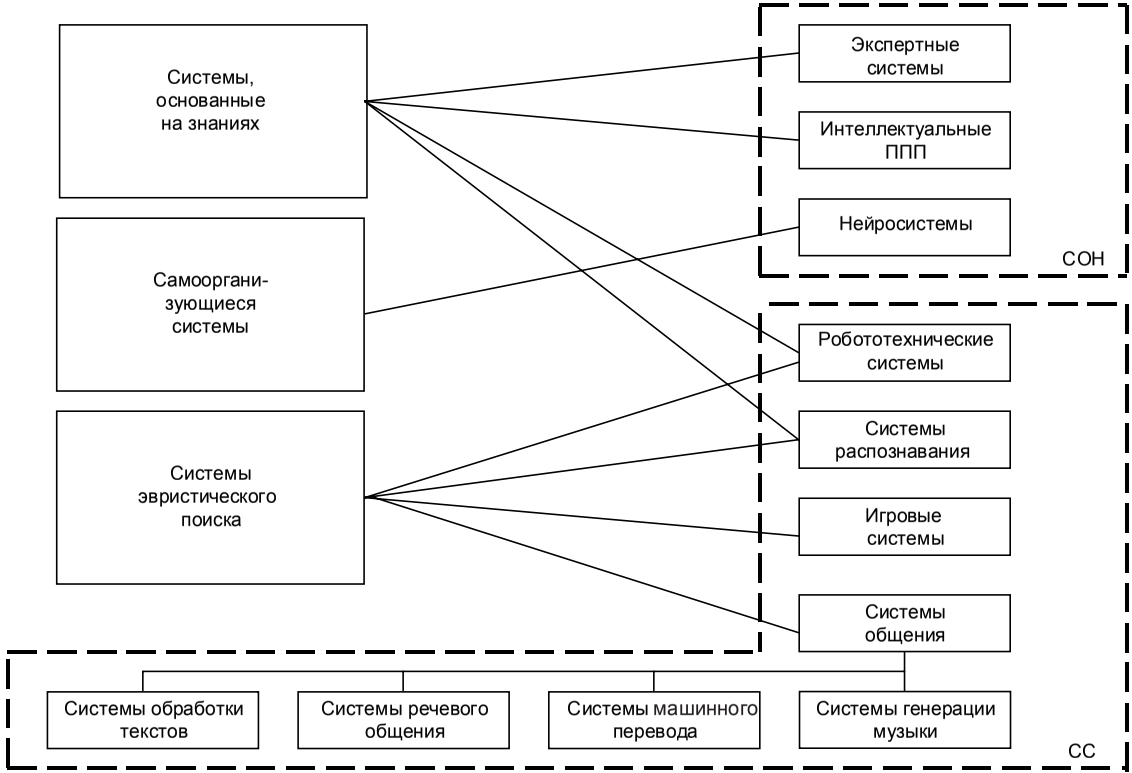


Рис. 10.1.4. Классификация систем искусственного интеллекта

В системах, основанных на знаниях, предполагается, что *исходные знания способны в соответствии с запросами пользователей к системе порождать новые знания*. При этом *сама процедура порождения новых знаний называется логическим выводом (или просто выводом)*. Термин "логический" здесь неслучаен с двух точек зрения. Системы, основанные на знаниях, моделируют мыслительную деятельность людей лишь на логическом (а не на физиологическом) уровне и, кроме того, основным математическим аппаратом, лежащим в основе систем этого типа, является аппарат математической логики.

К системам искусственного интеллекта, полностью основанным на знаниях, относятся два класса систем: *экспертные системы* и *интеллектуальные пакеты прикладных программ* (ИППП). Основные идеи этого направления частично (или даже в значительной части) реализуются и в других системах искусственного интеллекта, в частности, робототехнических системах распознавания и др. (см. рис. 10.1.4).

Наиболее последовательно идеи, на которых базируются системы искусственного интеллекта, основанные на знаниях, воплощены в экспертных системах, которые достаточно подробно рассмотрены в п. 10.3.

Под ИППП понимаются *инструментальные пакеты прикладных программ, в которых механизм сборки отдельных подпрограмм (решения частных задач) в общую программу решения требуемой задачи осуществляется автоматически, на основе механизма логического вывода.*

В *самоорганизующихся системах* реализуется попытка осуществить моделирование интеллектуальной деятельности человека (или более простых живых существ) не на логическом, а *на физиологическом уровне работы головного мозга*. В данном случае мозг человека моделируется *сетью идеальных нейронов*. В соответствии с доказанной фон Нейманом [37] теоремой при воздействии на такую сеть некоторых раздражителей она начинает вырабатывать адекватную реакцию, т. е. способна к самообучению путем самоорганизации. Несмотря на значительную теоретическую перспективность данного (исторически первого) направления в области

искусственного интеллекта, практически значимых результатов этот путь пока не дал. Последнее объясняется технической нереализуемостью на современном уровне достаточного числа взаимосвязанных нейронов в искусственно создаваемой сети.

В то же время данное направление позволило получить весомые результаты в области исследования возможностей создания компьютеров сверхвысокого быстродействия и тем самым повышать возможности систем искусственного интеллекта, создаваемых на других принципах. Кроме того, реальные результаты получены в создании нейросистем *распознавания образов*.

Основная идея, лежащая в основе создания нейросетей, базируется на теореме Мак-Каллока и Питтса [37], которая утверждает: любую вычислимую функцию можно реализовать с помощью *сети идеальных нейронов*. Эксперименты показывают, что реализация этих функций таким путем может осуществляться значительно быстрее, чем на традиционном компьютере. Компьютеры новой архитектуры, воплощающие данную идею, получили название *нейрокомпьютеры*.

Третье направление разработки систем искусственного интеллекта связано с реализацией *эвристического подхода* к построению таких систем. Главной особенностью, характерной для данного направления, является полный *отказ от следования принципу аналогии при моделировании механизма интеллектуальной деятельности* (ни на логическом, ни на физиологическом уровне). Методологической основой систем эвристического поиска служит то утверждение, что любая интеллектуальная деятельность начинается с некоторых данных и завершается получением определенных результатов также в виде данных. Если техническое устройство позволяет по аналогичным исходным данным получить эквивалентные результаты, то оно может быть отнесено к классу интеллектуальных (см. первое определение искусственного интеллекта). При этом механизм переработки исходных данных в результаты не оговаривается и, вообще говоря, может быть совершенно иным по сравнению с реальным. *Системы этого типа выполняют функции, которые традиционно производятся человеком, однако реализуют их другими способами.*

Широкое распространение данное направление получило при решении различных *игровых задач* (шахматы, шашки и т. д.). Однако подходы, присущие этому направлению, нашли применение и в других системах искусственного интеллекта, в частности *системах общения* (особенно в части речевого общения), *системах распознавания, робототехнических системах* и др. В то же время следует заметить: специфика эвристического подхода такова, что *рецепты создания программ для решения интеллектуальных задач в одной области практики,* как правило*, неприменимы в другой области*, а возникающая необходимость изменения характера учета факторов при решении прикладных задач вызывает существенную перестройку программы в целом.

При разработке интеллектуальных робототехнических систем основная задача состоит в решении теоретических и практических вопросов *организации целесообразного поведения подвижных роботов, снабженных сенсорными и эффекторными (исполнительными) механизмами* [54]. Принципиальное отличие робототехнических систем от систем искусственного интеллекта других типов заключается в том, что эти системы *не только воспринимают информацию из окружающего мира* и вырабатывают на ее основе определенные оценочные выводы, но и, сообразуясь с этими выводами, *вносят изменения в окружающий (анализируемый ими) мир.*

К настоящему времени в практике находят применение робототехнические системы с относительно простыми сенсорными и эффекторными механизмами, которые способны выполнять действия только в простых средах с заранее зафиксированными свойствами.

Основа проблемы распознавания образов или в более широком контексте — машинное зрение заключается в придании системе способности разрешения задач *преобразования огромного количества сенсорных данных* (например, присутствующих в телевизионном изображении) к *относительно краткому и осмысленному описанию наблюдаемой проблемной ситуации.* Содержанием такого описания, как правило, является тот минимальный (самый характерный) набор данных, которые отличают изучаемую ситуацию от стандартной. Основная сложность такого описания связана с ответом на следующие вопросы: какие объекты имеют место

в наблюдаемом кадре; какие из них являются ключевыми для выявленной ситуации; что надо принять за стандартную ситуацию для выявленных ключевых объектов; в чем отличие рассматриваемой ситуации от стандартной; откуда первоначально получать наборы стандартных ситуаций. Трудности, с которыми сталкивается практика при решении каждой из перечисленных задач, указывают на то, что, как и в случае робототехнических систем, данное направление находит реализацию только в самых простых случаях.

В дальнейшем будем рассматривать *системы, основанные на знаниях*, как получившие наибольшее практическое развитие и распространение в различных отраслях профессиональной деятельности, в том числе и в экономике, что обусловливает необходимость более подробного рассмотрения *методов представления знаний* в памяти ЭВМ.

Вопросы для самопроверки

1. Вспомните основные этапы развития теории искусственного интеллекта.
2. Каково соотношение понятий “психика — интеллект”, “сознание — разум”?
3. Дайте определение понятиям “искусственный бессловесный интеллект”, “искусственный словесный интеллект”, “искусственный разум”.
4. Вспомните классификацию систем искусственного интеллекта.
5. Какие СИИ относятся к классу систем, основанных на знаниях?
6. В чем смысл теоремы Мак-Каллока и Питтса?
7. Приведите примеры систем эвристического поиска.

10.2. Методы моделирования знаний

Выше уже частично рассматривались такие понятия, как “знания” и “системы, основанные на знаниях”, и отмечалась их особая

значимость в теории искусственного интеллекта. Сделаем еще одно весьма важное замечание: в настоящее время в области разработки систем искусственного интеллекта сложилась следующая аксиома: *никакой самый сложный и изощренный алгоритм извлечения информации (так называемый механизм логического вывода) из интеллектуальной системы не может компенсировать "информационную бедность" ее базы знаний.*

10.2.1. Знания и их свойства

Несмотря на широкое распространение и использование понятия "знания" в различных научных дисциплинах и на практике, строгого определения данного термина нет.

Довольно часто используют так называемый *прагматический подход*: говорят, что *знания — это формализованная информация, на которую ссылаются и/или которую используют в процессе логического вывода*. Однако такое определение ограниченно: оно фиксирует сознание на уже существующих методах представления знаний и, соответственно, механизмах вывода, не давая возможности представить себе другие ("новые").

Возможен и другой подход: попытаться на основе определения уже рассмотренного понятия "*данные"* (см. п. 9.1) выявить их свойства и особенности, сформировать дополнительные требования к ним и уже затем перейти к понятию "знания".

Напомним, что *данными называют формализованную информацию, пригодную для последующей обработки, хранения и передачи средствами автоматизации профессиональной деятельности.*

Какие же свойства "превращают" данные в знания? На рис. 10.2.1 представлены шесть основных свойств знаний (часть из них присуща и данным).

Кратко охарактеризуем эти свойства.

1. *Внутренняя интерпретация* (интерпретируемость). Это свойство предполагает, что в ЭВМ хранятся не только "собственно (сами) данные", но и "данные о данных", что позволяет содержательно их интерпретировать (см. рис. 10.2.2). Имея такую информацию, можно ответить на вопросы типа "Где находится НПО "Энергия"?" или "Какие предприятия выпускают космическую

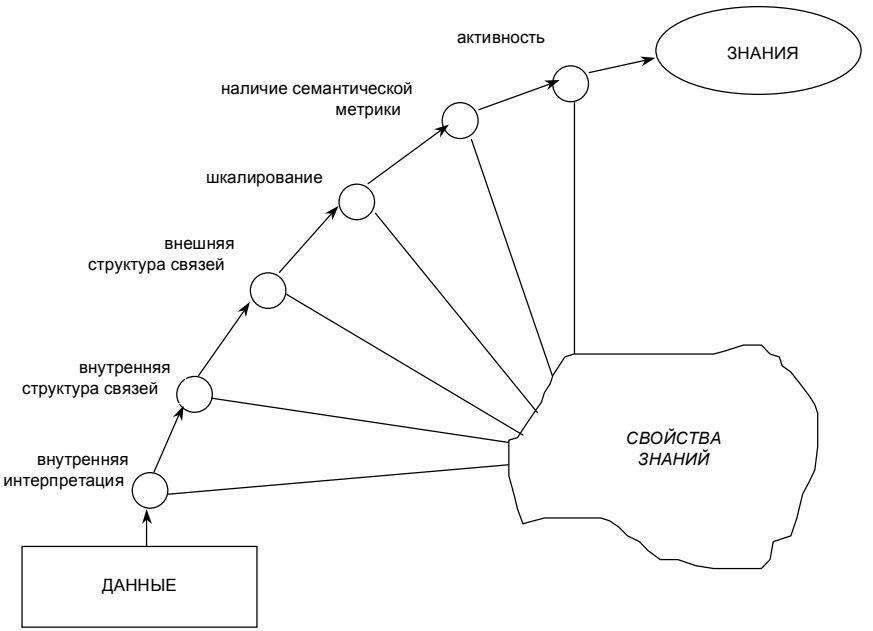


Рис. 10.2.1. Свойства знаний

технику?". При этом в первой строке таблицы на рис. 10.2.2 находятся "данные о данных" (метаданные), а в остальных — сами данные.

Предприятие	Место нахождения	Что выпускает
Завод им. Хруничева	Москва	Космическую технику
НПО "Энергия"	Королев	Космическую технику
НПО "Комета"	Москва	Конструкторскую документацию

Рис. 10.2.2. Иллюстрация свойства внутренней интерпретации

2. Наличие *внутренней структуры связей*.

Предполагается, что в качестве информационных единиц используются не отдельные данные, а их упорядоченные определенными отношениями (родовидовыми, причинно-следственными и др.)

структуры (эти отношения называют классифицирующими). Пример: факультет — курс — учебная группа — студент.

3. Наличие *внешней структуры связей*.

Внутренняя структура связей позволяет описывать отдельный объект (понятие). Однако объекты (понятия) способны находиться и в других отношениях (вступать в ситуативную связь). Пример: объекты "курс Государственного университета управления им. С. Орджоникидзе" и "урожай овощей в совхозе "Зареченский" могут находиться в ситуативной связи "принимает участие в уборке".

4. Возможность *шкалирования*.

Предполагает введение соотношений между различными информационными единицами (т. е. их измерение в какой-либо шкале — порядковой, классификационной, метрической и т. п.) и упорядочение информационных единиц путем измерения интенсивности отношений и свойств.

Пример: "97/ЭИ. 6-01 учебная группа занимает первое место на курсе по успеваемости".

5. Наличие *семантической метрики*.

Шкалирование позволяет соотнести информационные единицы, но прежде всего для понятий, имеющих "количественное" толкование (характеристики). На практике довольно часто встречаются понятия, к которым неприменимы количественные шкалы, но существует потребность в установлении их близости (например, понятия "искусственный интеллект" и "искусственный разум"). *Семантики* классифицируются следующим образом:

• *значение*, т. е. объективное содержание;

• *контекстуальный смысл*, определяемый связями данного понятия с другими, соседствующими в данной ситуации;

• *личностный смысл*, т. е. объективное значение, отраженное через систему взглядов эксперта;

• *прагматический смысл*, определяемый текущим знанием о конкретной ситуации (например, фраза "информация получена" может иметь как негативную, так и позитивную оценку — в зависимости от того, нужно это было или нет) [22].

6. Наличие *активности*.

Данное свойство *принципиально отличает* понятие "знание" от понятия "данные". Например, знания человека, как правило,

активны, поскольку ему свойственна познавательная активность (обнаружение противоречий в знаниях становится побудительной причиной их преодоления и появления новых знаний; стимулом активности является неполнота знаний; выражается в необходимости их пополнения). В отличие от данных знания позволяют выводить (получать) новые знания. Будучи активными, знания позволяют человеку решать не только типовые, но и принципиально новые, нетрадиционные задачи.

Кроме перечисленных, знаниям присущи такие свойства, как *омонимия* (слово "коса" может иметь три смысла, связанных с определениями: девичья; песчаная; острая) и *синонимия* (знания "преподаватель читает лекцию" и "студенты слушают лекцию" во многих случаях являются синонимами) и др.

Классифицировать знания можно по самым различным основаниям.

По способу существования различают *факты* (хорошо известные обстоятельства) и *эвристики* (знания из опыта экспертов).

По *способу использования* в экспертных системах — *фактические знания* (факты) — знания типа "А — это А"; *правила* — знания для принятия решений ("Если ... — то ..."); *метазнания* (знания о знаниях — указывают системе способы использования знаний и определяют их свойства). Классическими примерами метазнаний являются народные пословицы и поговорки, каждая из которых характеризует знания (рекомендации по деятельности) в широком классе конкретных ситуаций (например, пословица "Семь раз отмерь, один — отрежь" применима не только в среде хирургов или портных).

По *формам представления* знания подразделяют на *декларативные* (факты в виде наборов структурированных данных) и *процедуральные* (алгоритмы в виде процедур обработки фактов).

По *способу приобретения* знания подразделяются на *научные* (полученные в ходе систематического обучения и/или изучения) и *житейские, бытовые* (полученные в "ходе жизни").

Дадим еще ряд определений, часто встречающихся в литературе [21].

Интенсиональные знания — знания, характеризующие некоторый класс объектов или относящиеся к нему.

Экстенсиональные знания — знания, относящиеся к конкретному объекту из какого-либо класса (факты, сведения, утверждения и т. д.).

Заметим: отношения интенсиональных и экстенсиональных знаний — это родовидовые отношения. Например, понятие "технологическая операция" — это интенсионал, а понятие "пайка" — это экстенсионал, так как пайка — одна из технологических операций. Очевидно, что эти *понятия относительны*. Так, понятие "пайка", в свою очередь, можно считать интенсионалом по отношению к понятиям "пайка серебром" и "пайка оловом". Как правило, такого рода знания относятся к декларативным.

Физические знания — знания о реальном мире.

Ментальные знания — знания об отношениях объектов.

Мир задачи — совокупность знаний, используемых в задаче.

Мир пользователя — совокупность знаний пользователя.

Мир программы — совокупность знаний, используемых в программе.

Морфологические и синтаксические знания — знания о правилах построения структуры описываемого явления или объекта (например, правила написания букв, слов, предложений и др.).

Семантические знания — знания о смысле и значении описываемых явлений и объектов.

Прагматические знания — знания о практическом смысле описываемых объектов и явлений в конкретной ситуации. (Например, редкая монета для нумизмата и филателиста имеет различную прагматическую ценность.)

Предметные знания — знания о предметной области, объектах из этой области, их отношениях, действиях над ними и др.

10.2.2. Классификация методов моделирования знаний

Для того чтобы манипулировать всевозможными знаниями из реального мира с помощью компьютера, необходимо осуществить их *моделирование*.

При проектировании модели представления знаний следует учесть *два требования*:

- *однородность представления*;
- *простота понимания*.

Выполнение этих требований позволяет упростить механизм логического вывода и процессы приобретения знаний и управления ими, однако, как правило, создателям интеллектуальной системы приходится идти на некоторый компромисс в стремлении обеспечить одинаковое понимание знаний и экспертами, и инженерами знаний, и пользователями.

Классификация методов моделирования знаний с точки зрения подхода к их представлению в ЭВМ показана на рис. 10.2.3.

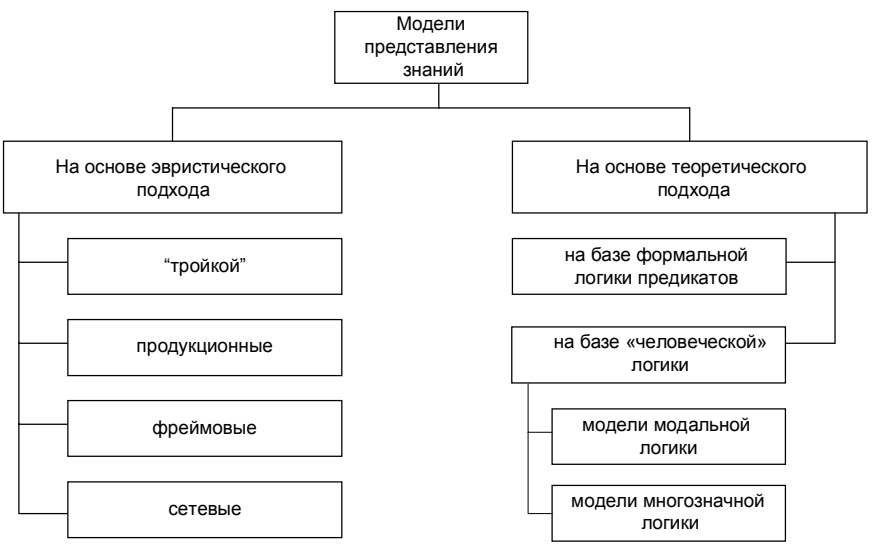


Рис. 10.2.3. Классификация моделей представления знаний

Дадим общую характеристику основных методов представления знаний.

Модели на основе эвристического подхода

1. *Представление знаний тройкой “объект — атрибут — значение”.* Один из первых методов моделирования знаний. Как пра-

вило, используется для представления *фактических знаний в простейших системах.*

Примеры:

Объект	Атрибут	Значение
Студент	Успеваемость	Отличник
Дом	Цвет	Белый
Пациент	Температура	Нормальная

Очевидно, что для моделирования знаний даже об одном объекте (например, о "студенте" или "доме") из предметной области *необходимо хранить значительное число* таких "троек".

2. *Продукционная модель* (модель правил; модель продукций — от англ. production — изготовление, *выработка*). В настоящее время наиболее проработанная и распространенная модель представления знаний, в частности — в экспертных системах.

Модель предусматривает разработку системы продукционных правил (правил продукций), имеющих типовой вид:

ЕСЛИ A_1 И A_2 И ... A_n, ТО B_1 ИЛИ B_2 ИЛИ ... ИЛИ B_m,

где A_i и B_j — некоторые высказывания, к которым применены логические операции И и ИЛИ. Если высказывания в левой части правила (ее часто называют *антецедент* — условие, причина) истинно, истинно и высказывание в правой части (*консеквент* — следствие).

Полнота базы знаний (базы правил) определяет возможности системы по удовлетворению потребностей пользователей. Логический вывод в продукционных системах основан на построении прямой и обратной цепочек заключений, образуемых в результате последовательного просмотра левых и правых частей соответствующих правил, вплоть до получения окончательного заключения.

Пусть в некоторой области памяти (БЗн) хранятся следующие правила (суждения):

• правило 1 — *ЕСЛИ* в стране происходит падение курса национальной валюты,

ТО материальное положение населения ухудшается;

• правило 2 — *ЕСЛИ* объемы производства в стране падают,

ТО курс национальной валюты снижается;

• правило 3 — *ЕСЛИ* материальное положение населения ухудшается,

ТО в стране возрастает уровень смертности.

Если на вход системы поступит новый факт "В стране значительный уровень падения объемов производства", то из правил можно построить цепочку рассуждений и сформулировать два заключения:

факт 1 — правило 2 — правило 1 — заключение 1 — правило 3 — заключение 2,

где заключение 1 (промежуточный вывод) — "Материальное положение населения ухудшается"; заключение 2 (окончательный вывод) — "В стране возрастает уровень смертности".

Отметим, что в современных экспертных системах в базе знаний могут храниться тысячи правил, а коммерческая стоимость одного невыводимого (нового, дополнительного) правила весьма высока.

Главными достоинствами продукционных систем являются простота пополнения и изъятия правил; простота реализации механизма логического вывода и наглядность объяснений результатов работы системы.

Основной недостаток подобных систем — трудность обеспечения непротиворечивости правил при их большом числе, что требует создания специальных правил (так называемых метаправил) разрешения возникающих в ходе логического вывода противоречий. Кроме того, время формирования итогового заключения может быть достаточно большим.

3. *Фреймовая модель*. Сравнительно новая модель представления знаний. Само понятие "фрейм" (англ. frame — рама, рамка, скелет, сгусток, сруб и т. д.) было введено в 1975 г. Марком Минским (M. Minsky, США).

Фрейм — это минимальная структура информации, необходимая для представления знаний о стереотипных классах объектов, явлений, ситуаций, процессов и др. С помощью фреймов можно моделировать знания о самых разнообразных объектах интересующей исследователя предметной области — важно лишь, что-

бы эти объекты составляли класс концептуальных (повторяющихся, стереотипных) объектов, процессов и т. п. Примерами стереотипных жизненных ситуаций могут служить собрание, совещание; сдача экзамена или зачета; защита курсовой работы и др. Примеры стереотипных бытовых ситуаций: отъезд в отпуск; встреча гостей; выбор телевизора; ремонт и др. Примеры стереотипных понятий: алгоритм; действие; методика и др. На рис. 10.2.4 представлен фрейм технологической операции “соединять” [21].

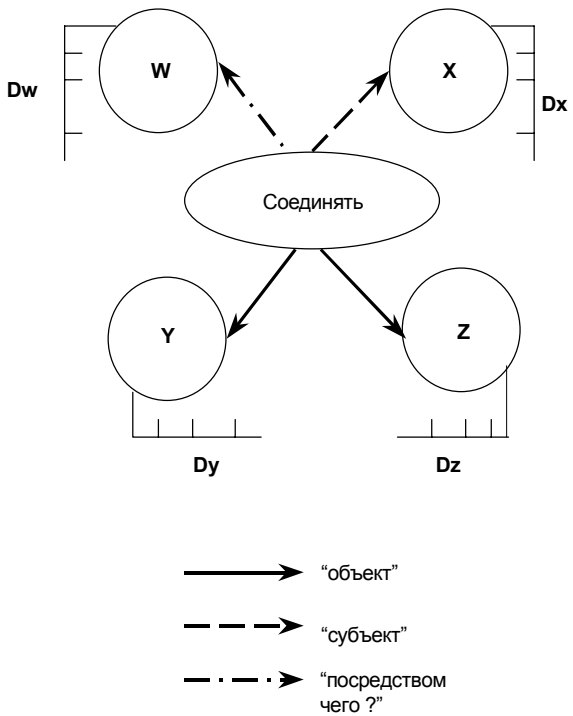


Рис. 10.2.4. Фрейм ситуации “соединять”

Данный фрейм описывает ситуацию “Субъект X соединяет объект Y с объектом Z способом W”. На рисунке обозначены:

вершины X, Y, Z, W — *слоты* (англ. slot — прорез; щель; пустота — составляющие фрейма);

дуги — отношения;

D_X, D_Y, D_Z, D_W — так называемые *шанции* — области возможных значений соответствующих слотов.

Наполняя слоты конкретным содержанием, можно получить фрейм конкретной ситуации, например: “Радиомонтажник соединяет микросхему с конденсатором способом пайки”. Заполнение слотов шанциями называют *активизацией* фрейма.

С помощью фреймов можно моделировать как процедурные, так и декларативные знания. На рис. 10.2.4 представлен пример представления процедурных знаний.

На рис. 10.2.5 приведен пример фрейма “технологическая операция”, иллюстрирующий представление декларативных знаний для решения задачи проектирования технологического процесса.

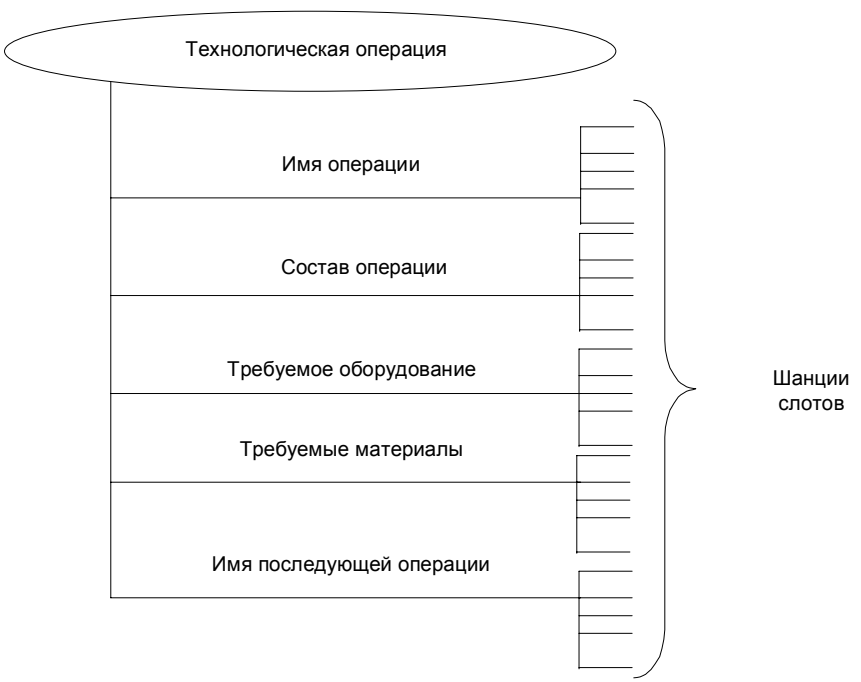


Рис. 10.2.5. Фрейм понятия “технологическая операция”

По содержательному смыслу фрейма выделяют [21]:

- фреймы-понятия;
- фреймы-меню;
- фреймы с иерархически вложенной структурой.

Фрейм-понятие — это фрейм типа *И*. Например, фрейм "операция" содержит объединенные связкой И имена слотов "что делать", "что это дает", "как делать", "кто делает", "где делать" и т. д., а фрейм "предмет" — слоты с именами "назначение", "форма", "вес", "цвет" и т. д.

Фрейм-меню — это фрейм типа *ИЛИ*. Он служит для организации процедурных знаний с помощью оператора "выбрать". Например, фрейм "что делать" может состоять из объединенных связкой *ИЛИ* слотов "решить уравнение", "подставить данные", "уточнить задачу" и т. д., причем каждый из этих слотов может иметь несколько значений.

Фрейм с иерархически вложенной структурой предполагает, что в нем в качестве значений слотов можно использовать имена других фреймов, слотов и т. д., т. е. использовать иерархическую структуру, в которой комбинируются другие виды фреймов (в итоге получают так называемые *фреймы-сценарии*).

Значения слотов могут содержать ссылки на так называемые *присоединенные процедуры*. Различают два вида присоединенных процедур:

— процедуры-*демоны*;

— процедуры-*слуги*.

Процедуры-*демоны* присоединяются к слоту и активизируются при изменении информации в этом слоте (выполняют вспомогательные операции — например, автоматически корректируют информацию во всех других структурах, где используется значение данного слота) (см. рис. 10.2.6).

Процедуры-*слуги* активизируются при выполнении некоторых условий относительно содержимого слотов (часто по запросу). Данные процедуры определяются пользователем при создании фрейма. Например, во фрейме "Учебная аудитория" можно предусмотреть слоты "Длина" и "Ширина", а по их значениям вычислять значение слота "площадь".

Процедуры-демоны

1.	Процедура “Если — добавлено” (IF — ADDED)	Выполняется, когда новая информация помещается в слот
2.	Процедура “Если — удалено” (IF — REMOVED)	Выполняется, когда информация удаляется из слота
3.	Процедура “Если — нужно” (IF — NEEDED)	Выполняется, когда запрашивается информация из пустого слота

Рис. 10.2.6. Типы присоединенных процедур

Фреймы позволяют использовать многие свойства знаний и достаточно широко употребляются. Их достоинства и недостатки схожи с достоинствами и недостатками семантических сетей, которые будут рассмотрены ниже.

4. *Модель семантической сети* (модель Куилиана).

Семантическая сеть — это *направленный граф с поименованными вершинами и дугами*, причем *узлы* обозначают конкретные *объект*ы, а *дуги — отношения между ними* [21]. Как следует из определения, данная модель представления знаний является более общей по отношению к фреймовой модели (иными словами, фреймовая модель — частный случай семантической сети). Семантическую сеть можно построить для любой предметной области и для самых разнообразных объектов и отношений.

В семантических сетях используют три типа вершин:

- вершины-*понятия* (обычно это существительные);
- вершины-*события* (обычно это глаголы);
- вершины-*свойства* (прилагательные, наречия, определения).

Дуги сети (семантические отношения) подразделяют на четыре класса:

- *лингвистические* (падежные, глагольные, атрибутивные);
- *логические* (*И, ИЛИ, НЕ*);
- *теоретико-множественные* (множество — подмножество, отношения целого и части, родовидовые отношения);
- *квантифицированные* (определяемые кванторами общности $\forall$ и существования $\exists$).

(Напомним, что *кванторы* — это *логические операторы*, переводящие одну высказывательную форму в другую и позволяющие указывать объем тех значений предметных переменных, для которых данная высказывательная форма *истинна*.)

Приведем два примера.

На рис. 10.2.7 представлена семантическая сеть для предложения (ситуации) “Студент Табуреткин добросовестно изучает новый план счетов на 2005 г. перед сдачей экзамена по дисциплине “Бухгалтерский учет”.

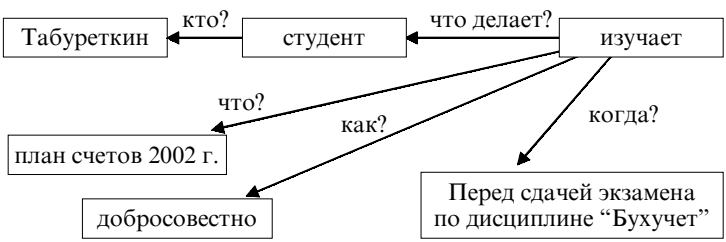


Рис. 10.2.7. Семантическая сеть для предложения (ситуации)

Рис. 10.2.8 содержит фрагмент семантической сети для понятия “автомобиль” (обозначения: IS-A — “есть, является”; HAS-PART — “имеет часть”).

Из приведенных примеров понятно, почему многие специалисты по ИИ считают *фрейм частным случаем семантической сети со строго структурированными знаниями.*

Основное достоинство методов моделирования знаний с помощью семантических сетей и фреймов — универсальность, удобство представления как декларативных, так и процедуральных знаний. Имеют место и два недостатка:

— *громоздкость, сложность* построения и *изменения*;

— *потребность* в разнообразных *процедурах обработки*, связанная с разнообразием типов дуг и вершин.

В рамках реализации *теоретического подхода* применяют *логические модели*, прежде всего использующие представления знаний в системе *логики предикатов*. Преимущества такого подхода

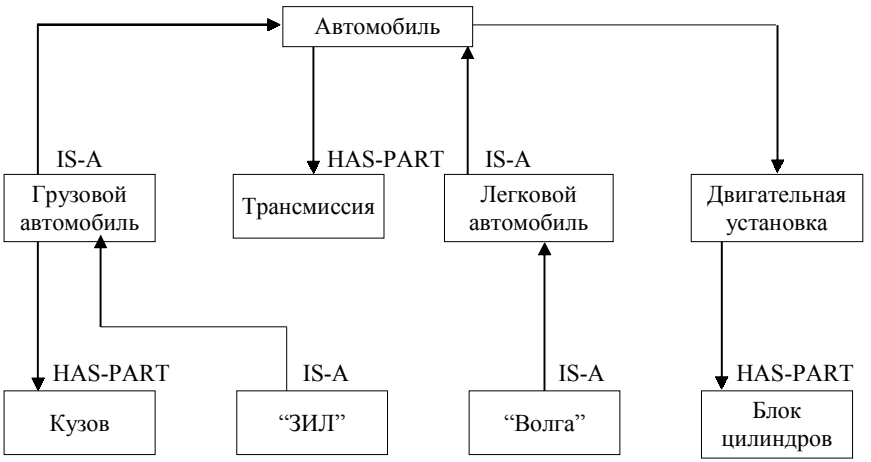


Рис. 10.2.8. Фрагмент семантической сети понятия “автомобиль”

очевидны: единственность теоретического обоснования и возможность реализации системы путем введения формально точных определений и правил получения выводов. Однако в полной мере претворить в жизнь данный подход даже для “простых” задач оказалось весьма сложно. Поэтому появились *попытки перейти от формальной логики к так называемой человеческой логике* (модальной логике, многозначной логике и др.), модели которой в большей или меньшей степени учитывают “человеческий фактор”, т. е. являются в определенном смысле компромиссными в плане использования и теоретического, и эвристического подходов.

Очень коротко остановимся на ставшей классической предикатной модели представления знаний. Первые попытки использовать такую модель относятся к 50-м гг. прошлого века. Дадим несколько определений.

Пусть имеется некоторое *множество объектов*, называемое *предметной областью*. Выражение $P(x_1, x_2, ..., x_n)$, где x_i, $i = 1, ..., n$ — так называемые предметные переменные, а P принимает значения 0 или 1, называется *логической функцией* или *предикатом*.

Предикат $P(x_1, x_2, ..., x_n)$ задает отношение между элементами $x_1, x_2, ..., x_n$ и обозначает высказывание, что “$x_1, x_2, ..., x_n$ находятся

между собой в отношении P(x). Например, если A — множество целых чисел, а P(a) — высказывание “a — положительное число”, то P(a) = 1 при a > 0 и P(a) = 0 при a ≤ 0.

Из подобного рода элементарных высказываний с помощью *логических связок* образуют *более сложные высказывания*, которые могут принимать те же значения — “истина” и “ложь”. В качестве связок используются *конъюнкция*, *дизъюнкция*, *импликация*, *отрицание*, *эквивалентность*.

Предикат от n переменных называют n-*местным*.

Одноместные (унарные) предикаты отражают свойства *определенного объекта* или *класса объектов*. *Многоместные* предикаты позволяют записывать *отношения*, которые существуют *между группой элементов*.

Если a — тоже предикат, то P(a) — предикат 2-го порядка, и т. д. до n-го порядка.

Приведем примеры различных предикатов.

1. Унарный предикат (высказывание) “Река впадает в Каспийское море” имеет значение 1, если “Река” = “Волга”, и значение 0, если “Река” = “Днепр”.

2. Двухместный предикат “x_1 не меньше x_2” может иметь значение 1 или 0 в зависимости от значений x_1 и x_2. Если значение предиката *тождественно равно 1* при любых значениях предметных переменных, он называется *тавтологией*.

В аппарат исчисления предикатов входят также *символы функций* (обычно обозначаемые латинскими буквами f, g, h и т. д.), задаваемых на множестве предметных переменных, и *кванторы общности* ∀ и *существования* ∃.

3. Представление с помощью предиката знаний, заключенных в теореме Пифагора: P {g [f(x), f(y)], f(z)}, где предикат P — “быть равным”, функция g(x, y) = x + y; функция $f(x) = x^2$.

Иногда используется такая форма записи:

РАВНЫ [СУММА (КВАДРАТ(x), КВАДРАТ(y)), КВАДРАТ (z)].

Предикат P равен 1, если x, y, z — соответственно длины катетов и гипотенузы прямоугольного треугольника.

Как уже отмечалось, предикаты удобны для описания *декларативных* знаний (фактов, событий и т. п.). Их *главные достоин-*

ства — возможность реализации строгого вывода знаний (исчисления предикатов) и сравнительная *компактность модели*. К сожалению, предикаты мало пригодны для записи *процедуральных* знаний. Кроме того, опыт показал, что человеческое знание по своей структуре много сложнее структуры языков предикатного типа, поэтому требуется специальные навыки *подгонки* структуры реального знания под структуру модели (как правило, значительно обедняющей исходные знания).

Вопросы для самопроверки

1. Дайте определение понятия “знания”.
2. Назовите свойства знаний.
3. Поясните свойства знаний: наличие внешней структуры связей; шкалирование. Приведите примеры.
4. Как вы понимаете смысл свойства “наличие семантической метрики”?
5. В чем смысл свойств омонимии и синонимии знаний?
6. Какое свойство принципиально отличает знания от данных?
7. Вспомните основные модели знаний.
8. Приведите примеры: продукционных моделей знаний; семантических сетей.
9. Что понимают под термином “фрейм”?

10.3. Этапы проектирования экспертных систем

10.3.1. Структура и назначение экспертных систем

В настоящее время среди всех систем искусственного интеллекта (ИИ) наибольшее распространение (по некоторым оценкам, до 90%) получили *экспертные системы* (ЭС) различных типов. Объяснение этому находится в самой истории развития технологии искусственного интеллекта. В 1960-х гг. специалисты в области ИИ пытались *моделировать сложный процесс мышления*, отыс-

кивая *общие методы решения широкого класса задач* и реализуя их в *универсальных программах*. Как уже отмечалось, большая часть таких попыток была неудачной.

Дальнейшие исследования в 1970-е гг. были сконцентрированы на разработке *двух групп методов*:

• методов *представления задач* (в стремлении сформулировать решаемую проблему так, чтобы ее было легче решить);

• методов *поиска (вывода) ответа* (в стремлении создать достаточно хитроумные способы управления ходом решения задачи, обеспечивающие приемлемый расход машинных ресурсов).

Однако и эта стратегия не принесла реальных успехов.

Только в конце 1970-х гг. был сделан *принципиальный вывод: эффективность программы при решении интеллектуальных задач в большей степени зависит от знаний*, которыми она обладает, а не только от используемых *формализмов и схем вывода*. Чтобы сделать *систему интеллектуальной*, ее нужно снабдить *множеством высококачественных знаний о некоторой предметной области*. Это послужило основой новой концепции развития систем ИИ — создания *специализированных программных систем, каждая из которых является как бы экспертом в некоторой узкой предметной области*. Такие программы в дальнейшем и стали называть ЭС.

Огромный интерес к ЭС обусловлен тремя основными обстоятельствами [22]:

• ЭС ориентированы на решение *широкого круга задач в ранее не формализуемых областях*, которые считались малодоступными для использования ЭВМ;

• ЭС предназначены *для решения задач в диалоговом режиме* со специалистами (конечными пользователями), от которых не требуется знания программирования. Это резко расширяет сферу использования вычислительной техники, которая в данном случае выступает как *инструмент подкрепления (поддержки)* памяти специалиста и усиления его способностей к логическому выводу;

• специалист, использующий ЭС для решения своих задач, *может достигать, а иногда и превосходить по результатам возможности экспертов в данной области* знаний, что позволяет *резко повысить квалификацию рядовых специалистов* за счет аккуму-

ляции знаний в ЭС, в том числе знаний экспертов высшей квалификации.

Свое название ЭС получили *по двум причинам*:

• информацию (знания) для них поставляют *эксперты*;

• ЭС выдает *решения, аналогичные* тем, которые формулируют *эксперты*.

Понятие "эксперт" заслуживает отдельного обсуждения.

По Д. Уотермену, *эксперт* (англ. domain expert — знаток, специалист в области, сфере деятельности) — *человек*, который за годы обучения и практики научился *чрезвычайно эффективно* решать *задачи*, относящиеся к *конкретной предметной области* [54]. Главным в этом определении является требование к эксперту, которое предъявляется и к ЭС: *эффективность решения конкретных задач из узкой предметной области*.

В соответствии с определением П. Джонса [54], "*эксперт* — это *человек*, который благодаря обучению и опыту *может делать то, что мы все, остальные люди, делать не умеем*; эксперты работают не просто профессионально, но к тому же *уверенно и эффективно*. Эксперты обладают *огромными познаниями* и пользуются различными *приемами и уловками* для применения своих знаний к проблемам и заданиям; они также умеют *быстро переворошить массу несущественной информации*, чтобы добраться до главного, и хорошо *умеют распознавать в проблемах, с которыми сталкиваются, примеры тех типовых проблем, с которыми они уже знакомы*. В основе поведения экспертов лежит *совокупность практически применимых знаний, которую мы будем называть компетентностью*. Поэтому разумно предположить, что эксперты — это те люди, к которым надо обратиться, когда *мы желаем проявить компетентность, делающую возможным такое поведение, как у них*".

Отметим, что в обоих определениях подчеркиваются источники знаний экспертов — обучение и практика (опыт).

Таким образом, можно дать следующее определение: под *ЭС* понимается *программная система*, выполняющая действия, *аналогичные* тем, которые выполняет *эксперт* в некоторой прикладной предметной области, делая определенные *заключения* в ходе выдачи *советов и консультаций*.

Каково же назначение ЭС? В табл. 10.3.1 приведены основные области их применения (в порядке уменьшения числа ЭС, используемых в данной области).

Таблица 10.3.1

Основные области применения ЭС

№ п\п	Область применения ЭС
1	Проектирование экспертных систем
2	Медицинский диагноз и консультации по лечению
3	Консультации и оказание помощи пользователю по решению задач в различных предметных областях
4	Автоматическое программирование, проверка и анализ программного обеспечения
5	Проектирование сверхбольших интегральных схем. Обучение в различных предметных областях
6	Техническая диагностика и выработка рекомендаций по ремонту оборудования
7	Планирование в различных предметных областях. Анализ данных в различных предметных областях (в том числе и статистический). Интерпретация геологических данных и выработка рекомендаций по обнаружению полезных ископаемых
8	Интерпретация данных и планирование эксперимента в ходе научных исследований в области биологии. Решение задач, связанных с космическими исследованиями
9	Обеспечение научных исследований в химии, выработка рекомендаций по синтезу соединений
10	Управление проектированием, технологическими процессами и промышленным производством. Анализ и синтез электронных схем. Формирование математических понятий, преобразование математических выражений
11	Анализ рисков в политике и экономике

Структура типовой ЭС представлена на рис. 10.3.1.

На рисунке обозначены: СОЗ — система, основанная на знаниях; ЛП — лингвистический процессор; РП (БД) — рабочая память (база данных); БЗн — база знаний; МЛВ — механизм (машина) логического вывода; КПЗн — компонент приобретения знаний; КОб — компонент объяснений.

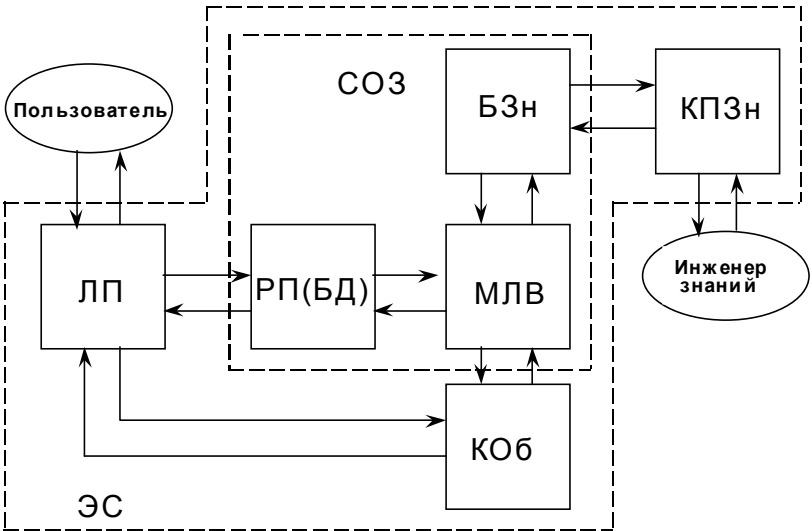


Рис. 10.3.1. Структура экспертной системы

Дадим краткую характеристику структурных элементов ЭС.

СОЗ представляет собой программную систему, состоящую из трех элементов: БЗн, МЛВ и РП (БД).

БЗн — часть ЭС (СОЗ), предназначенная для *генерации и поддержания динамической модели знаний о предметной области* (в качестве возможных моделей знаний могут использоваться рассмотренные выше — в п. 10.2.2 — продукционные, сетевые или фреймовые модели).

МЛВ — часть ЭС (СОЗ), реализующая *анализ* поступающей в ЭС и имеющейся в ней *информации и формирование (вывод) на ее основе новых заключений (суждений) в ответ на запрос к системе*.

РП (БД) — часть ЭС (СОЗ), предназначенная для *информационного обеспечения работы МЛВ*, прежде всего в части хранения и обработки поступивших (новых) фактов (суждений) и промежуточных результатов логического вывода (подробнее см. п. 10.4).

ЛП предназначен для обеспечения *комфортного интерфейса между конечным пользователем и ЭС*. В нем реализуются *процедуры*

морфологического, синтаксического и семантического контроля поступающих в систему запросов и приведение их к виду, "понятному" ЭВМ. При выдаче ответной информации осуществляется обратная операция — заключение "переводится" на *ограниченный естественный язык*, понятный конечному пользователю. Отметим, что в первых ЭС ЛП отсутствовал, так как общение с машиной осуществлялось на (строго) формальном языке. В дальнейшем (особенно при переходе к ЭВМ пятого поколения) значимость ЛП в составе ЭС будет возрастать.

КПЗн предназначен для *обеспечения работы инженера знаний по поддержанию модели знаний, адекватной реальной предметной области* (генерации БЗн, ее тестирования, пополнения новыми знаниями, исключения неверных (ставших таковыми) знаний и т. п.).

Наличие КОб, обеспечивающего по запросу пользователя *выдачу информации о ходе и исходе логического вывода*, принципиально отличает ЭС от всех других программных систем. Дело в том, что в большинстве случаев конечному пользователю недостаточно сообщить лишь *конечное заключение* ЭС, которое он должен (может) использовать в своей профессиональной деятельности. Гораздо большее доверие вызывает у него конечный *вывод, подтвержденный понятными промежуточными рассуждениями*. Кроме того, с помощью КОб можно организовать процесс обучения конечных пользователей работе с ЭС. В обучающих ЭС КОб играет еще более важную роль.

Важным классом СОЗ является класс интеллектуальных пакетов прикладных программ (ППП). Структура такого пакета приведена на рис. 10.3.2.

Интеллектуальные ППП дают возможность конечному пользователю решать прикладные задачи по их описаниям и исходным данным *без программирования — генерация ("сборка") программы "под задачу" осуществляется автоматически механизмом логического вывода.* БЗн в интеллектуальном ППП может строиться по любому из известных эвристических методов (часто используются семантические *сети* и *фреймы*), лишь бы настраиваемая МЛВ программа была эффективна для решения поставленной задачи.

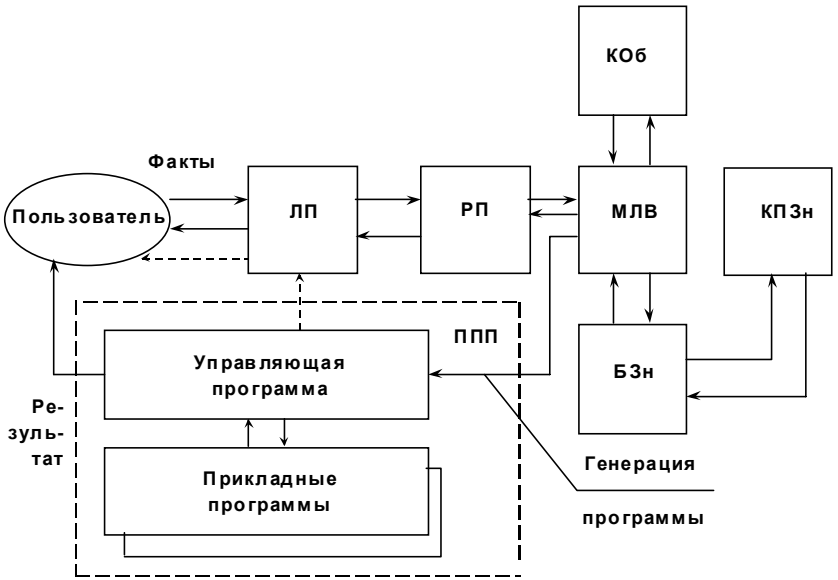


Рис. 10.3.2. Структура интеллектуального пакета прикладных программ

10.3.2. Классификация, этапы и средства разработки экспертных систем

Существует *множество признаков*, по которым можно (весьма условно) классифицировать ЭС [49]. По *степени сложности* различают *поверхностные* и *глубинные* ЭС; по *степени связанности* правил продукционные ЭС подразделяют на *связные* и *малосвязные*, по *типу предметной области* выделяют *статические, динамические и ЭС реального времени* и т. п. Процесс создания ЭС занимает немало времени, поэтому определенный интерес представляет классификация ЭС *по стадиям разработки* — применительно к продукционным ЭС изображена на рис. 10.3.3 (заметим, что аналогичные стадии в своем жизненном цикле имеют практически все — достаточно сложные — программные системы).

Масштабы разработки ЭС предопределили создание *специальных инструментальных (аппаратных и программных) средств*,

Рис. 10.3.3. Классификация экспертных систем по стадиям разработки

систематизированное представление которых составляет содержание рис. 10.3.4.

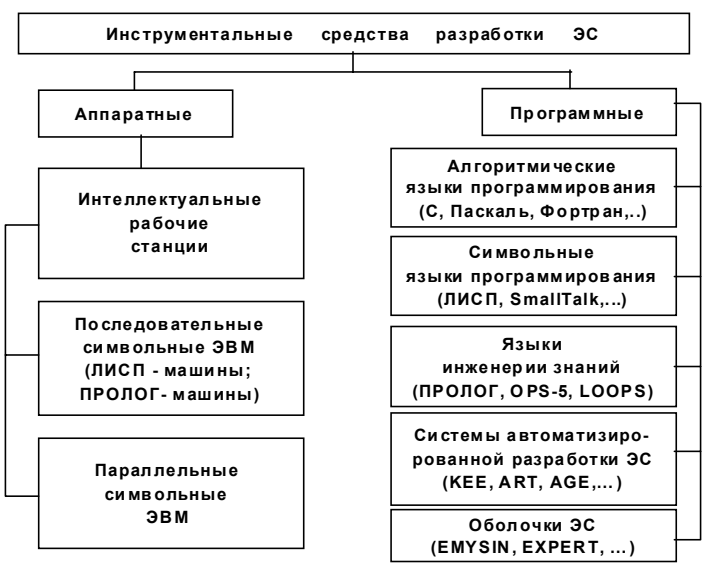


Рис. 10.3.4. Инструментальные средства разработки экспертных систем

Следует отметить, что *первоначально* разработка ЭС осуществлялась на *традиционных алгоритмических языках* программирования с реализацией на *универсальных ЭВМ*. В дальнейшем были созданы как *специализированные аппаратные и программные средства*, так и *средства автоматизации программирования*. Появились и *оболочки ЭС*, которые по задумке авторов должны были существенно упростить (и удешевить) разработку систем. Однако *в полной мере эти надежды не оправдались* (как показало дальнейшее развитие прикладных программных средств не только в области искусственного интеллекта, и не могли оправдаться). Это связано с *принципиальной сложностью использования конкретной ЭС* (даже весьма эффективной в своей предметной области) для решения *совершенно иных задач*, а именно таким путем создавались первые оболочки ЭС. Еще *более проблематичными* представляются попытки создания так называемых *универсальных оболочек*, пригодных для применения "во всех" предметных областях.

При создании *ЭС наибольшую трудность представляет разработка совершенной базы знаний*, т. е. моделирование знаний экспертов о некоторой предметной области. Разработка любой модели — в том числе и модели знаний — представляет собой полностью не формализуемый процесс, содержащий *элементы творчества и строго формальных действий*. Условное соотношение "искусства" и "науки" при создании ЭС представлено на рис. 10.3.5.

Разработка ЭС включает нескольких этапов [38], основное содержание которых применительно к продукционным системам отражено на рис. 10.3.6.

Процедуры уточнения, перепроектирования и *переформулирования не являются обязательными*, они характерны для разработки достаточно сложных ЭС и, как правило, *предполагают проведение нескольких итераций*. Отметим, что перечисленные этапы работ (идентификация — концептуализация — формализация — реализация — тестирование), как и стадии разработки (см. рис. 10.3.3), являются обязательными при создании любой программной системы.

Очевидно, что *разработка ЭС является коллективным трудом*, в котором принимают участие различные специалисты. Цен-

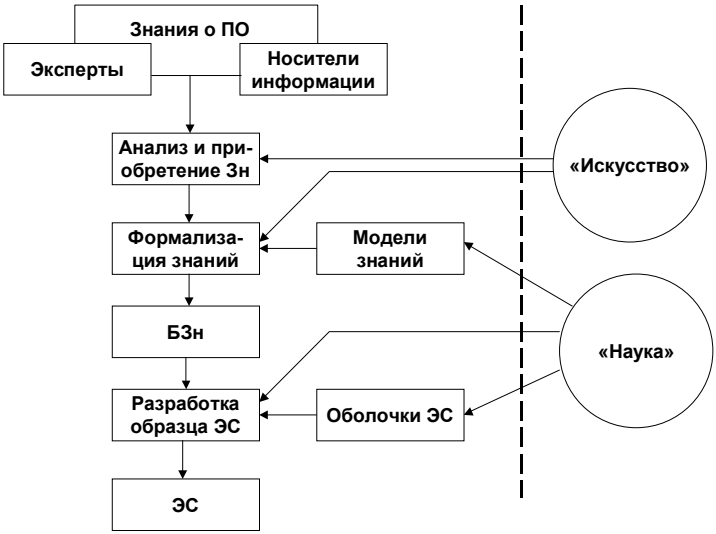


Рис. 10.3.5. Соотношение формальных и неформальных процедур при разработке ЭС

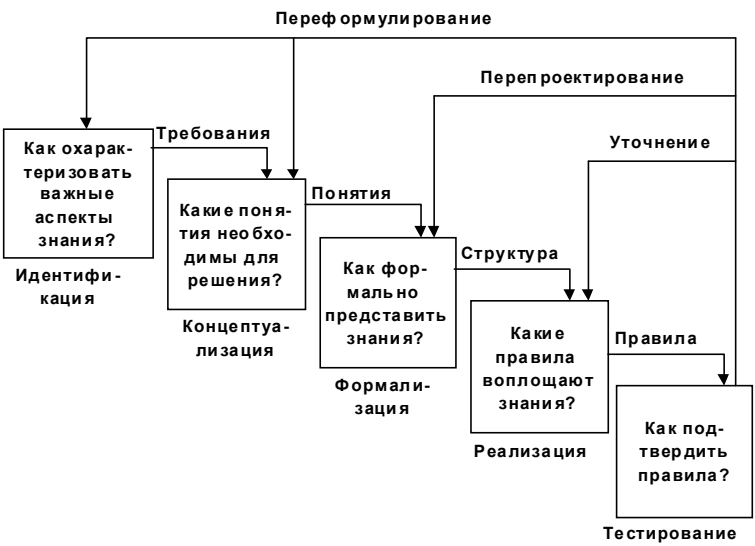


Рис. 10.3.6. Этапы разработки экспертной системы

тральное место в схеме взаимодействия участников создания ЭС занимает *инженер знаний* (англ. knowledge engineer). Именно он организует все важнейшие работы и осуществляет их координацию. Ему принадлежит право *выбора типовых* или — при необходимости и наличии соответствующих ресурсов — *заказа новых инструментальных средств разработки* ЭС. Он работает с предметными экспертами, генерирует, тестирует, уточняет и пополняет базу знаний и т. д. Направления взаимодействия создателей ЭС (этот процесс иногда называют *игрой* [38]) представлены на рис. 10.3.7.

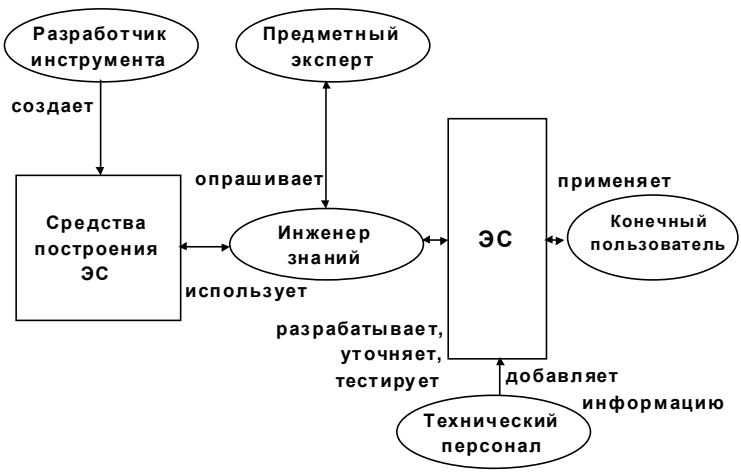


Рис. 10.3.7. Схема взаимодействия создателей экспертной системы

Как явствует из вышеизложенного, разработка ЭС — *сложный, дорогостоящий* и *длительный* процесс. Последнее обстоятельство иллюстрируется рис. 10.3.8, на котором приведены условные затраты времени на создание систем для решения проблем различной сложности [38].

Существует ряд подходов к оценке того, *когда же разработка ЭС является рациональной* [21, 22, 26]. На наш взгляд, наиболее конструктивен подход Д. Уотермена, который основан на проверке *возможности, оправданности и разумности построения системы.*

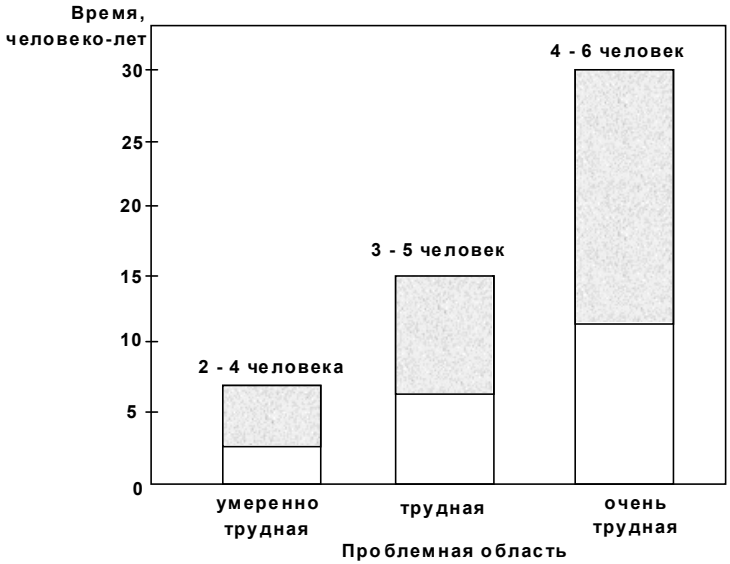


Рис. 10.3.8. Затраты времени на создание экспертной системы

При этом предлагается считать, что *разработка ЭС возможна* при совместном выполнении следующих основных условий:

- задача не требует *общедоступных* знаний;
- решение задачи требует только *интеллектуальных* действий;
- *существуют* подлинные (компетентные) *эксперты*;
- эксперты *способны описать свои методы* (приемы, уловки и т. п.) решения задачи;
- эксперты *единодушны* в своих решениях (или, по крайней мере, их мнения "хорошо" *согласованы*);
- задача понятна и "не слишком" трудна.

Разработка ЭС оправдана, если выполняется хотя бы одно из следующих основных условий:

- получение решения задачи *высокорентабельно*;
- человеческий *опыт решения задачи* по различным причинам *утрачен*;
- *число экспертов* в рассматриваемой предметной области *мало*;

• *опыт* решения задачи *востребован во многих местах*;

• *опыт* нужно применять *во враждебных человеку условиях*.

Наконец, *разработка ЭС разумна*, если совместно выполняются следующие основные условия:

• задача требует *эвристических* решений;

• задача требует *оперирования символами*;

• задача "*не слишком" проста*;

• задача представляет *практический интерес*;

• задача имеет *размерность*, допускающую *реализацию*.

При всей условности и субъективности проверки наличия перечисленных обстоятельств можно по-новому взглянуть на причины столь широкой представительности перечня областей применения ЭС.

В заключение напомним о принципиальной важности совершенства базы знаний для эффективности ЭС. Другим важнейшим составным элементом любой системы, основанной на знаниях, в том числе и ЭС, является *механизм логического вывода* (МЛВ). Обсуждению основ функционирования МЛВ для различных моделей представления знаний посвящен следующий пункт.

Вопросы для самопроверки

1. К какому классу систем искусственного интеллекта относятся экспертные системы (ЭС)?

2. Почему эти системы получили такое название?

3. Дайте определение термину "эксперт".

4. Изобразите структуру экспертной системы.

5. Поясните назначение основных элементов ЭС: лингвистического процессора; базы знаний; рабочей памяти; механизма логического вывода.

6. Какие элементы ЭС не являются обязательными?

7. Каким образом пополняется база знаний?

8. Вспомните классификацию ЭС.

9. Назовите этапы создания ЭС.

10. Какие специалисты участвуют в создании ЭС?

10.4. Основы построения и использования механизмов логического вывода

Механизм логического вывода — неотъемлемая *часть СОЗ (ЭС), реализующая функции вывода (формирования) умозаключений (новых суждений)* на основе информации из *базы знаний* (*БЗн*) и *рабочей памяти* (*РП*).

Как следует из определения, для работы МЛВ необходима как "долговременная" информация, содержащаяся в БЗн в выбранном при разработке ЭС виде, так и "текущая", оперативная информация, поступающая в РП после обработки в лингвистическом процессоре запроса пользователя. Таким образом, БЗн отражает основные (долговременные) закономерности, присущие предметной области. Запрос пользователя, как правило, связан с появлением каких-либо новых фактов и/или с возникновением потребности в их толковании.

Перед рассмотрением конкретных МЛВ подчеркнем *несколько важных обстоятельств*:

- *единого* МЛВ для *произвольных СОЗ (ЭС) не существует*;
- *МЛВ полностью определяется моделью представления знаний*, принятой в данной системе;
- существующие *МЛВ не являются строго фиксированными* ("узаконенными") для каждого типа СОЗ (ЭС).

10.4.1. Механизм логического вывода в продукционных системах

Из всех известных механизмов вывода данный механизм является *наиболее формализованным* (предопределенным).

Различают два типа логического вывода:

- *прямой* вывод (прямая цепочка рассуждений);
- *обратный* вывод (обратная цепочка рассуждений).

Сущность *прямого* логического вывода в продукционных ЭС состоит в построении *цепочки выводов* (продукций или правил), связывающих *начальные факты с результатом вывода.*

В терминах "факты — правила" формирование цепочки вывода заключается *в многократном повторении элементарных шагов "сопоставить — выполнить".*

Рассмотрим следующий ***пример*** [54]. В базе знаний некоторой ЭС содержатся три правила, а в рабочей памяти до начала вывода — пять фактов: B, C, H, G, E (см. рис. 10.4.1а). Пусть на вход системы (в РП) поступил факт A. Механизм вывода "просматривает" левые части правил с целью нахождения таких из них, которые позволяют извлечь новые факты (процедура "сопоставить"). В нашем примере на основе третьего правила выводится факт D. Происходит элементарный шаг "выполнить" — данный факт заносится в РП (рис. 10.4.1б). Процедуру "сопоставить" по фактам C и D выявляет факт F. После шага "выполнить" в РП попадает этот факт (рис. 10.4.1в). По фактам F и B выводится факт Z, и дальнейший "просмотр" правил БЗн новой информации не дает.

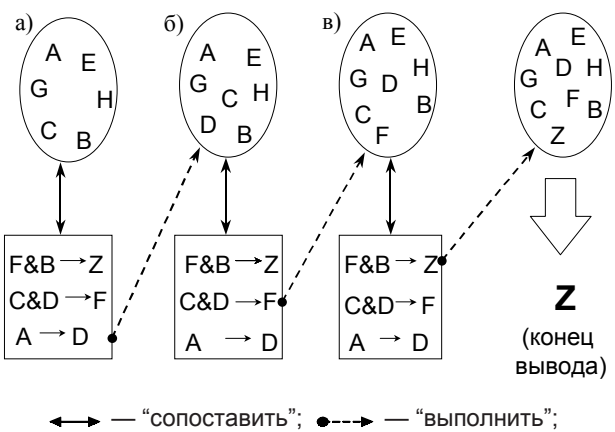


Рис. 10.4.1. Иллюстрация прямой цепочки рассуждений

Таким образом, прямая цепочка рассуждений состоит из следующих фактов: A — D — F — Z. Иными словами, из факта A на основе имеющихся в БЗн правил "выведен" факт Z.

Отметим, что, несмотря на очевидную простоту прямого вывода для пользователя ЭС, от которого требуется лишь сообщить системе о вновь поступивших или интересующих его фактах, для БЗн со *значительным числом правил* могут возникнуть две проблемы:

• когда *завершить вывод*;

• как обеспечить *непротиворечивость правил*.

Последнее обстоятельство требует формирования и хранения в ЭС так называемых *метаправил* — правил "работы с правилами". Кроме того, прямая цепочка рассуждений иногда требует значительного времени.

Механизм обратного вывода имеет совершенно иной алгоритм. Его идея заключается *в проверке справедливости некоторой гипотезы* (некоторого суждения, факта), которая *выдвигается пользователем и проверяется ЭС*. При этом осуществляется проверка истинности не *левых*, а *правых* частей продукций, а вопрос формулируется так: "Что нужно, чтобы правая часть данного правила была справедлива, и есть ли необходимые суждения в рабочей памяти?" На рис. 10.4.2 показана работа механизма обратного вывода для того же примера, что для прямой цепочки рассуждений (в предположении, что факт A занесен в РП).

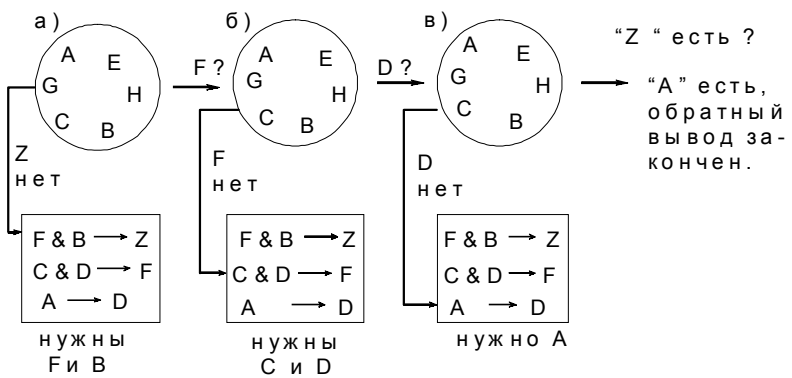


Рис. 10.4.2. Иллюстрация обратной цепочки рассуждений

При реализации данного механизма пополнения РП новыми (выведенными) фактами не производится, а лишь проверяется наличие необходимых суждений на очередном шаге алгоритма. Как следует из рис. 10.4.2а), поскольку непосредственно факта Z в РП нет, производится анализ правых частей правил до поиска такого

правила, которое обосновывает справедливость суждения Z. Чтобы факт Z был истинным, необходимы факты F и B. Факт B есть, факта F нет (рис. 10.4.2б). Чтобы факт B был истинным, необходимы факты C и D. Факт C есть, факта D — нет (рис. 10.4.2в). Наконец, чтобы был справедлив факт D, нужно наличие факта A, и так как он в РП имеется, обратный вывод закончен. Окончательный результат — на основании имеющихся в ЭС правил и фактов гипотеза о справедливости факта Z подтверждается.

Очевидно, что обратная цепочка рассуждений предъявляет к квалификации пользователя ЭС определенные требования — он должен *уметь формулировать "правдоподобные" гипотезы*. В противном случае легко представить весьма непродуктивную работу ЭС, проверяющей и отвергающей одну гипотезу за другой (в качестве примера аналогичной ситуации представим себе врача, ставящего один диагноз за другим и пользующего пациента лекарствами от самых разных болезней). Платой за выполнение данного требования служит, как правило, сокращение времени реакции ЭС на запрос пользователя.

Для обеспечения уверенности пользователя в получаемых ЭС суждениях *после обратного вывода часто прибегают к прямой цепочке рассуждений*. Совпадение результатов работы обоих механизмов служит гарантией получения истинного вывода.

10.4.2. Понятие о механизме логического вывода в сетевых системах

Механизм логического вывода в сетевых системах основан на использовании двух ведущих принципов:

- *наследования свойств;*
- *сопоставления по совпадению*.

Первый принцип базируется на учете важнейших связей, отражаемых в семантической сети. К таким связям относятся:

- связь *"есть", "является"* (англ. IS-A);
- связи *"имеет часть", "является частью"* (англ. HAS-PART, PART-OF).

Последовательно переходя от одного узла сети к другому по направлению соответствующих связей, можно выявить (извлечь)

новую информацию, характеризующую тот или иной узел. На рис. 10.4.3а) показан малый фрагмент некоторой семантической сети и обозначена так называемая *ветвь наследования свойств*. Из этого фрагмента можно вывести заключения типа “Иван — человек”, “у Ивана есть голова”, “мужчина имеет голову” и т. п.

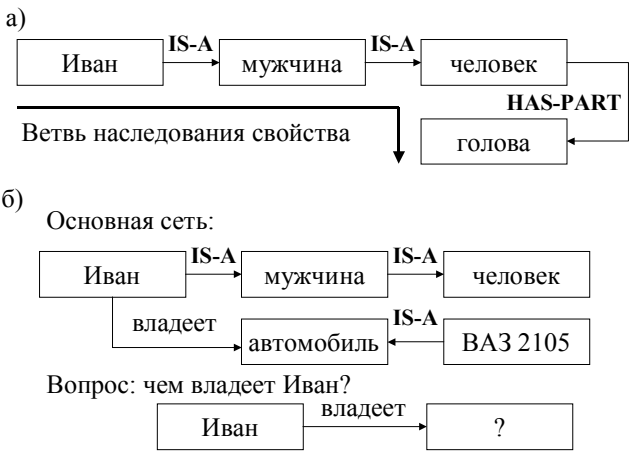


Рис. 10.4.3. Иллюстрация механизма логического вывода в семантической сети

Принцип *сопоставления по совпадению* основан на представлении вопроса к системе в виде *фрагмента семантической сети* с использованием *тех же названий сущностей (узлов) и связей*, что и в основной сети, и реализации процедуры “наложения” вопроса на сеть и поиска такого его положения, которое соответствует ответу на вопрос. На рис. 10.4.4 б) помимо уже известной связи “есть”, представлено *отношение владения* (связь “владеет”). Вопрос “Чем владеет Иван?” формализуется с помощью узла “Иван” и отношения “владеет”. Далее в простейшем случае осуществляется перебор узлов сети, имеющих имя “Иван” (если они имеются) и поиск такого из них, который имеет связь “владеет”. Далее может быть задействован принцип наследования свойств. Ответами на поставленный в примере вопрос будут суждения “Иван владеет автомобилем” и “Иван владеет (автомобилем) ВАЗ 2105”. Понятно, что

в практике использования ЭС такого типа приходится реализовывать значительно *более сложную процедуру поиска, включающую элементы семантического анализа.*

10.4.3. Понятие о механизме логического вывода во фреймовых системах

Как уже отмечалось в п. 10.2.2, обычно фреймовая модель знаний имеет сложную иерархическую структуру, отражающую реальные объекты (понятия) и отношения (связи) некоторой предметной области. *Механизм логического вывода в таких ЭС основан на обмене значениями между одноименными слотами различных фреймов и выполнении присоединенных процедур "если — добавлено", "если — удалено" и "если — нужно".* Условная схема таких действий для простейшего варианта представлена на рис. 10.4.4.

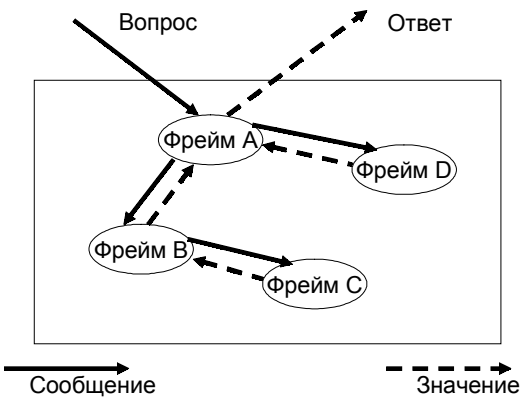


Рис. 10.4.4. Иллюстрация механизма вывода во фреймовой модели

Запрос к ЭС в виде сообщения поступает в старший по иерархии фрейм (на рисунке — фрейм A). Если ответа на запрос нет ни в одном из слотов этого фрейма или их совокупности, соответствующие сообщения (запросы) передаются во все фреймы, где имеется слот (слоты), имена которых содержатся в запросе или

необходимы для поиска ответа на него (фреймы B и D). Если в них содержится искомый ответ, значение соответствующего слота передается в старший по иерархии фрейм (из фрейма D во фрейм A). Если для этого нужна дополнительная информация, предварительно передается сообщение (из фрейма B во фрейм C) и получается значение (из фрейма C во фрейм B). Значения, передаваемые в ответ на сообщения, либо непосредственно содержатся в соответствующих слотах фреймов, либо определяются как результат выполнения присоединенных процедур.

В современных фреймовых системах, как правило, для пользователя реализована возможность *формулировать запросы на языке, близком к реальному*. Интерфейсная программа (лингвистический процессор) должна "уметь" по результатам анализа запроса определять, в какой (какие) слот (слоты) необходимо поместить значение (значения) для инициализации автоматической процедуры поиска ответа.

Рассмотрим более конкретный пример, иллюстрирующий работу фреймовой ЭС, используемой в подразделении, организующем научно-исследовательскую работу (НИР) в некотором учреждении. На рис. 10.4.5 представлена иерархия справочной информации об отчете по НИР (о понятии, узле "отчет по НИР").

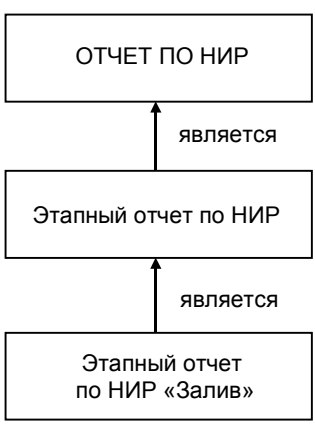


Рис. 10.4.5. Иерархия понятия "Отчет по НИР"

Рис. 10.4.6 содержит структуры понятий “Отчет по НИР” и “Этапный отчет по НИР”, а рис. 10.4.7 — структуру понятия “Этапный отчет по НИР “Залив” со значениями некоторых слотов и присоединенными процедурами.

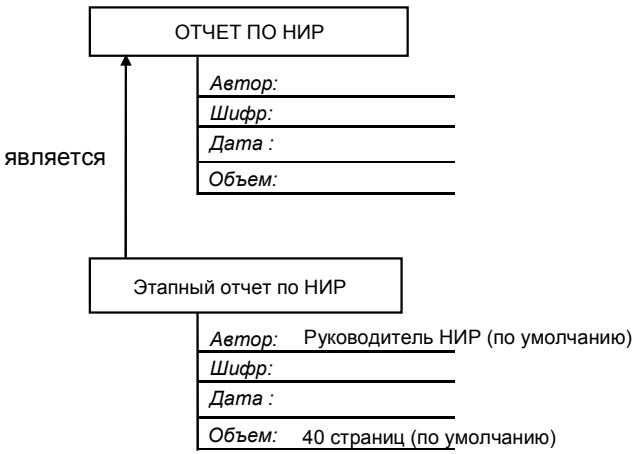


Рис. 10.4.6. Структуры понятий “Отчет по НИР” и “Этапный отчет по НИР”

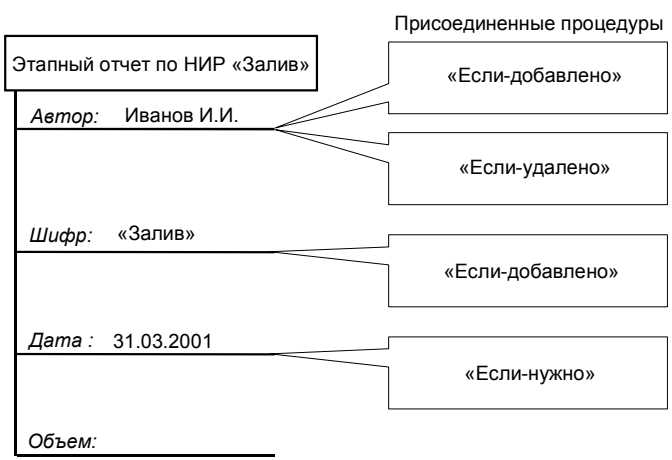


Рис. 10.4.7. Структура понятия “этапный отчет по НИР “Залив”

Фреймовая система функционирует следующим образом. Пусть в ЭС поступил запрос от полномочного пользователя: "Необходима информация о ходе выполнения НИР "Залив" (напомним, что, как правило, язык исходного запроса близок к естественному). Информация проходит через лингвистический процессор, анализируется и в виде значения "Залив" вносится в слот *Шифр* узла "Этапный отчет по НИР "Залив". Далее начинают работать присоединенные процедуры:

• процедура "если — добавлено", связанная со слотом *Шифр*, выполняется, поскольку в слот было введено некоторое значение. Эта процедура осуществляет поиск сведений о руководителе НИР "Залив" (в нашем примере — И. И. Иванов) и вписывает это имя в слот *Автор* узла "Этапный отчет по НИР "Залив";

• процедура "если — добавлено", связанная со слотом *Автор*, выполняется, так как в слот было вписано значение. Эта процедура начинает составлять сообщение, чтобы отправить его Иванову И. И., но обнаруживает, что отсутствует значение слота *Дата*;

• процедура "если — добавлено", просматривая слот *Дата* и найдя его пустым, активизирует процедуру "если — нужно", связанную с этим слотом. Процедура найдет текущую дату, используя календарь ЭС, выберет ближайшую к ней (но большую) дату представления отчета (в нашем примере — 31.03.2003) и впишет ее в слот *Дата*;

• процедура "если — добавлено", связанная со слотом *Автор*, найдет, что отсутствует еще одно значение, необходимое для формирования выходного сообщения, а именно — значение слота *Объем*. Данный слот (узла "Этапный отчет по НИР "Залив") не имеет присоединенных процедур, поэтому приходится брать значение по умолчанию из одноименного слота общей концепции "Этапного отчета по НИР" (в нашем примере — 40 страниц).

Теперь ЭС может сформировать выходное сообщение типа: "Этапный отчет по НИР "Залив" должен быть представлен Ивановым И. И. к 31 марта 2003 г. Предполагаемый объем отчета — 40 страниц" и/или "Иванов И. И.! Представьте этапный отчет по НИР "Залив" объемом не более 40 страниц к 31 марта 2003 г.".

Если в какой-либо момент значение слота *Автор* (в нашем примере — Иванов И. И.) будет удалено, то сработает процедура "если — удалено" и система автоматически отправит Иванову И. И. уведомление о том, что отчет не требуется.

10.4.4. Механизм логического вывода в диагностических системах байесовского типа

Диагностические ЭС широко применяются в различных областях человеческой деятельности (медицине, технике, экономике и др.). Как правило, в них используются *продукционные модели* знаний о предметной области. Однако, если имеется возможность использования в правилах статистических данных о понятиях и связях между ними, весьма целесообразно применить известную *теорему Байеса* для пересчета апостериорных вероятностей по результатам проверки наличия тех или иных симптомов.

Применительно к техническим диагностическим системам используется следующая схема формализации:

- объект имеет множество возможных *неисправностей* $S_1, S_2, ..., S_m$;
- каждой неисправности приписывается *априорная вероятность*

$$P(S_1), P(S_2), ..., P(S_m); \sum_{i=1}^{m} P(S_i) = 1;$$

- каждая *неисправность проявляется через симптомы* $C_1, C_2, ..., C_n$,

причем каждая неисправность характеризуется "своими" симптомами из "общего" списка;

- известны *условные вероятности* проявления симптомов при каждой неисправности

$P(C_j / S_i)$, $i = 1, 2, ..., m$; $j = 1, 2, ..., n$.

Тогда можно определить апостериорные вероятности наличия неисправности при данном симптоме:

$$P(S_i / C_1) = \frac{P(C_1 / S_i) \cdot P(S_i)}{P(C_1)} = \frac{P(C_1 / S_i) \cdot P(S_i)}{\sum_{i=1}^{m} P(C_1 / S_i) \cdot P(S_i)},$$

причем при расчете апостериорной вероятности учитывается, *наблюдался при испытании данный симптом или нет.*

Зная перечисленные вероятности, легко реализовать процедуру проверки наиболее вероятных симптомов, причем проверка очередного симптома должна сопровождаться пересчетом значений *всех* апостериорных вероятностей. Для получения априорных и условных вероятностей необходимо обработать статистические данные (при их наличии) или получить и обработать экспертную информацию.

На рис. 10.4.8 представлена иллюстрация описанного подхода. На рис. 10.4.8а) показаны исходные априорные вероятности наличия неисправностей. Как правило, задается некоторый уровень вероятности $P_{тр}$, превышение которого свидетельствует о необходимости проверки именно тех неисправностей, для которых и наблюдается превышение (в нашем примере — S_i). Далее проверяется наличие того симптома, для которого вероятность его проявления при i-й неисправности наибольшая (например, симптома C_1 на рис. 10.4.8б).

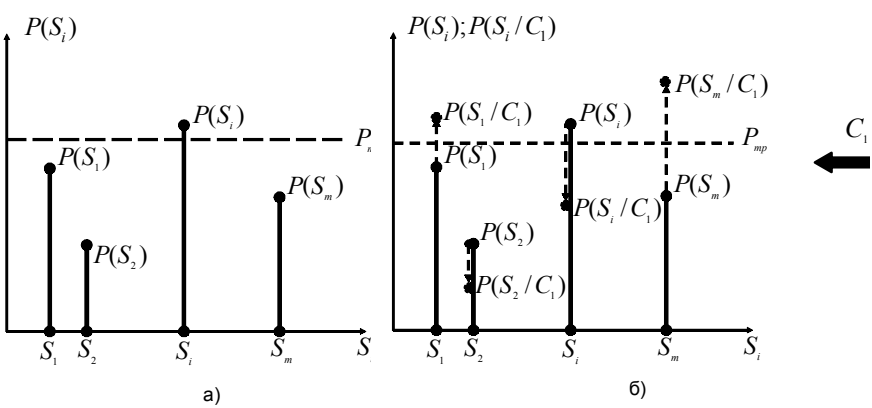
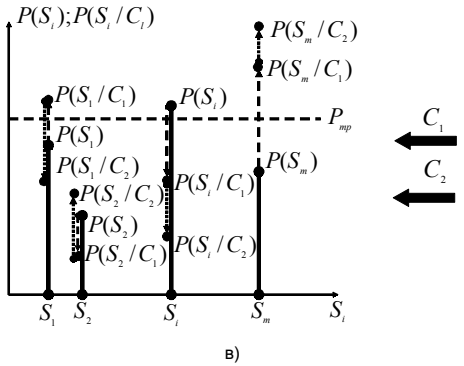


Рис. 10.4.8. Иллюстрация алгоритма работы диагностической ЭС



в)

Рис. 10.4.8. Иллюстрация алгоритма работы диагностической ЭС

По результатам проверки пересчитываются все апостериорные вероятности и выявляются те из них, которые превышают заданный уровень. По ним определяется очередной проверяемый симптом (на рис. 10.4.8в) — симптом C_2) и т. д. Заметим, что в результате пересчета апостериорная вероятность той или иной неисправности может как увеличиться, так и уменьшиться. После нескольких шагов данный алгоритм приводит к тому, что ЭС некоторые неисправности, апостериорные вероятности которых стали очень малыми, отбрасывает (перестает учитывать), а другие предлагает исправить.

Рассмотрим конкретный пример — фрагмент ЭС диагностического типа, предназначенной для поиска неисправности в автомобиле при следующих исходных данных:

- автомобиль может иметь четыре неисправности:
  - S_1 — неисправна аккумуляторная батарея;
  - S_2 — отсутствует топливо;
  - S_3 — “отсырело” зажигание;
  - S_4 — замаслены свечи;
- симптомами неисправностей являются:
  - C_1 — фары не горят;
  - C_2 — указатель топлива на нуле;
  - C_3 — автомобиль не заводится;
  - C_4 — стартер проворачивается;
  - C_5 — двигатель работает неустойчиво, “чихает”;

• значения априорных вероятностей:

$P(S_1) = 0{,}4$; $P(S_2) = 0{,}2$; $P(S_3) = 0{,}3$; $P(S_4) = 0{,}1$;

• значения условных вероятностей проявлений симптомов при наличии неисправностей приведены на рис. 10.4.9. Знаком "+" обозначены вероятности $P(C_j / S_i)$, знаком "–" — вероятности $P(\overline{C}_j / S_i)$.

S_i \ C_j	C_1		C_2		C_3		C_4		C_5	
	+	–	+	–	+	–	+	–	+	–
S_1	1	0	0,9	0,1	1	0	0	1	0,1	0,9
S_2	0,1	0,9	1	0	1	0	0,9	0,1	0,1	0,9
S_3	0,1	0,9	0,1	0,9	0,7	0,3	0,9	0,1	0,1	0,9
S_4	0,1	0,9	0,1	0,9	0,8	0,2	0,9	0,1	1	0

Рис. 10.4.9. Значения условных вероятностей проявления симптомов при наличии неисправностей

Реализация описанного выше алгоритма для последовательности симптомов "фары горят" — "указатель топлива не на нуле" — "стартер проворачивается" — "автомобиль заводится" — "двигатель "чихает" приведет к следующему заключению ЭС: "Просушите зажигание, проверьте свечи". Другая последовательность симптомов "фары не горят" — "автомобиль не заводится" — "стартер не проворачивается" — "указатель топлива не на нуле" –"двигатель не чихает" приведет ЭС к рекомендации "Замените аккумуляторную батарею". Если при проверке симптомов окажется, что "фары горят", "указатель топлива на нуле", "автомобиль не заводится", "стартер проворачивается", "двигатель не чихает", рекомендация ЭС, естественно, такова: "Залейте бензин".

Широкое распространение диагностических ЭС в различных областях деятельности определяется рядом обстоятельств.

Во-первых, возможностью обеспечения близости априорных и условных вероятностей, используемых в алгоритме, к "истинным" значениям.

Как правило, при грамотном учете опыта работы специалистов по устранению соответствующих неисправностей хорошие

оценки названных вероятностей могут быть получены по результатам обработки статистических данных.

Во-вторых, сравнительной простотой обеспечения диалога пользователя с системой на языке, близком к естественному, поскольку промежуточные и итоговые заключения ЭС, основанные на формальных шагах алгоритма работы, легко интерпретируются в понятные всем рекомендации.

В-третьих, возможностью выдачи пользователю (как правило, по запросу) промежуточных результатов диагностики неисправности, т. е. пояснения рекомендаций ЭС, что в подавляющем большинстве случаев облегчает их восприятие.

Наконец, возможностью постоянного учета текущего опыта пользователей и простотой корректировки (при необходимости) модели знаний о предметной области.

В заключение раздела отметим, что в учебнике рассмотрены лишь методологические основы построения и использования систем искусственного интеллекта. Практика совершенствования информационных технологий представляет все *новые направления применения интеллектуальных средств*. Так, например, наряду с интеллектуальными пакетами прикладных программ появились так называемые *интеллектуальные базы данных* [27], в которых используются достижения теории искусственного интеллекта как для организации хранения информации о предметной области, так и для удовлетворения информационных потребностей пользователей. Другим примером может служить разработанная специалистами Института человеческого и машинного познания при университете Западной Флориды (США) *технология хранения и представления пользователям информации*, получившая название *CMap* (англ. Concept Map — карта понятий), являющаяся одним из вариантов применения семантических сетей. С помощью этой технологии можно осваивать большие объемы сложноструктурированного материала, решая различные задачи (в том числе и задачи обучения специалистов). Еще одним примером является известная *концепция "интеллектуального дома (жилища)"*, призванная рационально использовать средства искусственного интеллекта при всестороннем обеспечении управления бытовыми система-

ми (начиная от регулирования подачи электроэнергии и воды и заканчивая включением/выключением микроволновой печи или телевизора в заданное время). Существует множество подобных примеров, подтверждающих главный вывод: *магистральным путем современной автоматизации профессиональной деятельности людей является ее интеллектуализация.*

Вопросы для самопроверки

1. Каково назначение механизма логического вывода (МЛВ)?
2. Поясните сущность прямой цепочки рассуждений для МЛВ продукционных ЭС.
3. Поясните сущность обратной цепочки рассуждений для МЛВ продукционных ЭС.
4. Какие принципы лежат в основе МЛВ ЭС, использующих для моделирования знаний семантическую сеть?
5. Приведите пример использования принципа наследования.
6. В чем смысл работы МЛВ фреймовых ЭС?
7. Сформулируйте постановку задачи для МЛВ диагностических ЭС (ДЭС) байесовского типа.
8. Каким образом можно определить исходные данные (априорные и условные вероятности) для работы этого МЛВ?
9. В чем достоинства таких ДЭС? недостатки?
10. Оцените возможности машинной реализации известных вам МЛВ. Попытайтесь ранжировать МЛВ по этому признаку.

Литература

1. Автоматизированные информационные технологии в экономике / Под ред. Г. А. Титоренко. — М.: ЮНИТИ, 2002.

2. Айвазян С. А., Мхитарян В. С. Прикладная статистика: Основы эконометрики. — М.: ЮНИТИ, 2001.

3. Амосов А. А., Дубинский Ю. А., Копченова Н. В. Вычислительные методы для инженеров. — М.: Высшая школа, 1994.

4. Арсеньев Ю. И., Шелобаев С. И., Давыдкова Т. Ю. Интегрированные интеллектуальные системы принятия решений. — М.: ЮНИТИ, 2002.

5. Бабешко Л. О. Коллокационные модели прогнозирования в финансовой среде. — М.: Экзамен, 2001.

6. Балдин К. В. и др. Теоретические основы моделирования сложных систем. Компьютерные методы и средства обучения моделированию. — М.: Издательство РДЛ, 1995.

7. Балдин К. В. Моделирование жизненного цикла сложных систем. Ч. I и II. — М.: Издательство РДЛ, 2000.

8. Балдин К. В., Уткин В. Б. Теоретические основы автоматизации управленческой деятельности в экономике. — Воронеж: МОДЭК, 2003.

9. Балдин К. В. и др. Экономические и информационно-аналитические основы управления инвестиционными проектами: Монография. — Воронеж: МОДЭК, 2002.

10. Бермант А. Ф., Араманович И. Г. Краткий курс математического анализа. — М.: Наука, 1969.

11. Благотатских В. А., Енгибарян М. А., Ковалевская Е. В. и др. Экономика, разработка и использование программного обеспечения. — М.: Финансы и статистика, 1998.

12. Бугров Я. С., Никольский С. М. Высшая математика: В 3 т. — М.: Дрофа, 2003.

13. Булдык Г. М. Сборник задач и упражнений по высшей математике. — Минск: ООО “Юнипресс”, 2002.

14. Верещагин Н. К., Шень А. Начала теории множеств. — М.: МЦНМО, 1999.

15. Выгодский М. Я. Справочник по высшей математике”. — М.: ДЖАНГАР, Большая медведица, 2001.

16. Гейн К., Сарсон Т. Системный структурный анализ: средства и методы. М.: “Эйтекс”, 1992.

17. Гилберт С. Самоучитель Visual C++ 6 в примерах. — М.: Диасофт, 2002.

18. Гончарова Г. А., Мочалин А. А. Элементы дискретной математики. — М.: ФОРУМ—ИНФРА-М, 2003.

19. Горчаков А. А., Орлова Н. В. Компьютерные экономико-математические модели. — М.: Компьютер, ЮНИТИ, 1995.

20. ГОСТ 19.101-77, 19.002-80, 19.003-80. Виды программ и программных документов. Схемы алгоритмов и программ. Правила выполнения. Обозначения условные графические. — М: Стандарты, 1984.

21. ГОСТ 24.003-84, 24.101-80, 24.103-84, 24.104-85, 24.101-85, 24.205-80, 24.301-80, 24.302-80, 24.303-80, 24.304-82, 34.003-90, 34.602-89. АСУ. Основные положения. Термины и определения. Общие требования. Техническое задание на АСУ. Условные обозначения. — М.: Стандарты, 1988.

22. Грес П. В. Математика для гуманитариев. — М.: ЮРАЙТ, 2000.

23. Гусак А. А. Справочное пособие к решению задач: математический анализ и дифференциальные уравнения. — Минск: ТетраСистемс, 1998.

24. Дейт К. Введение в системы баз данных. — М.: Диалектика, 1998.

25. Додж М., Стинсон К. Эффективная работа с Microsoft Excel 2000. — СПб.: Питер, 2002.

26. Дуброва Т. А. Статистические методы прогнозирования. — М.: ЮНИТИ, 2002.

27. Емеличев В. А., Мельников О. И., Сарванов В. И., Тышкевич Р. И. Лекции по теории графов. — М.: Наука, 1989.

28. Справочник по математике для экономистов / Под ред. В. И. Ермакова — М.: Высшая школа, 1987.

29. Жак С. В. Математические модели менеджмента и маркетинга. — Ростов н/Д: ЛаПО, 1997.

30. Завгородний В. И. Комплексная защита информации в компьютерных системах. — М.: Логос, 2001.

31. Зегжда Д. П., Ивашко А. М. Основы безопасности информационных систем. — М.: Горячая линия — Телеком, 2000.

32. Идельсон А. В., Блюмкина И. А. Аналитическая геометрия. Линейная алгебра. — М.: ИНФРА-М, 2000.

33. Интеллектуальные САПР технологических процессов в радиоэлектронике / А. С. Алиев, Л. С. Восков, В. Н. Ильин и др.; под ред. В. Н. Ильина. — М.: Радио и связь, 1991.

34. Искусственный интеллект. Кн. 1. Системы общения и экспертные системы / Под ред. Э. В. Попова. — М.: Радио и связь, 1990.

35. Калянов Г. Российский рынок CASE-средств // НС WEEK/RE. № 23. 1998.

36. Калянов Г. CASE — структурный системный анализ (автоматизация и применение). — М.: Лори, 1996.

37. Касперский Е. Antivirtual Toolkit Pro: Энциклопедия компьютерных вирусов. — М.: Лаборатория Касперского, 1999. URL: www.avp.ru.

38. Клиот-Дашинский М. И. Алгебра матриц и векторов. — СПб.: Лань, 2001.

39. Компьютерные информационные системы управленческой деятельности / Под ред. Г. А. Титоренко. — М.: Экономическое оборудование, 1993.

40. Корнеев В. В., Гарев А. Ф., Васютин С. В., Райх В. В. Базы данных. Интеллектуальная обработка информации. — М.: Нолидж, 2000.

41. Котов С. А. Нормирование жизненного цикла программной продукции. — М.: ЮНИТИ, 2002.

42. Краснощеков П. С., Петров А. А. Принципы построения моделей. — М.: ФАЗИС: ВЦ РАН, 2000.

43. Кремер Н. Ш. и др. Исследование операций в экономике. — М.: ЮНИТИ, 1997.

44. Кук Д., Бейз Г. Компьютерная математика. — М.: Наука, 1990.

45. Курош А. Г. Курс высшей алгебры. — М.: ГИФМЛ, 1962.

46. Лисичкин В. Т., Соловейчик И. Л. Математика. — М.: Высшая школа, 1991.

47. Ловцов Д. А., Сергеев Н. А. Управление безопасностью эргасистем. — М.: Изд-во РДЛ, 2000.

48. Локальные вычислительные сети: принципы построения, архитектура, коммутационные средства / Под ред. С. В. Назарова. — М.: Финансы и статистика, 1994.

49. Лукацкий А. В. Обнаружение атак. — СПб.: БХВ — Петербург, 2001.

50. Максимов Ю. Д. и др. Курс высшей математики для гуманитарных специальностей. — СПб.: Специальная литература, 1999.

51. Математика для бакалавров технических специальностей: Т. 1 / Общие разделы под общ. ред. Ю. Д. Максимова. — СПб.: Специальная литература, 1999.

52. Марков Л. Н., Размыслович Г. П. Высшая математика: ч. 1. Элементы линейной и векторной алгебры. Основы аналитической геометрии. — Минск: Амалфея, 1999.

53. Медведовский И. Д., Семьянов П. В., Леонов Д. Г. Атака на Internet. — М.: ДМК, 1999.

54. Моделирование производственно-инвестиционной деятельности фирмы /Под ред. Г. В. Виноградова. — М.: ЮНИТИ, 2002.

55. Мыльник В. В. и др. Системы управления. — М.: Экономика и финансы, 2002.

56. Нейлор К. Как построить свою экспертную систему: Пер. с англ. — М.: Энергоатомиздат, 1991.

57. Немыцкий В. и др. Курс математического анализа: В 2 т. — М.: ГИТТЛ, 1957.

58. Общая алгебра / Под общ. ред. Л. А. Скорнякова. Т. 1 — М.: Наука, 1990.

59. Олифер В. Г, Олифер Н. А. Компьютерные сети: Принципы, технологии, протоколы. — СПб.: Питер, 1999.

60. Оре О. Теория графов. — М.: Наука, 1968.

61. Першиков В. И., Савинков В. М. Толковый словарь по информатике. — М.: Финансы и статистика, 1991.

62. Петров А. А., Поспелов И. Г., Шананин А. А. Опыт математического моделирования экономики. — М.: Энергоатомиздат, 1996.

63. Пискунов Н. С. Дифференциальное и интегральное исчисления: В 2 т. — М.: Наука, 1964.

64. Подольский В. А., Суходский А. М. Сборник задач по высшей математике. — М.: Высшая школа, 1974.

65. Попов Э. В. и др. Статистические и динамические экспертные системы. — М.: Финансы и статистика, 2000.

66. Райордан Р. Основы реляционных баз данных. — М.: Русская редакция, 2001.

67. Саймино Д. Сети ИНТРАНЕТ: внутреннее движение: Пер. с англ. — М.: ООО "Бук Медиа Паблишер", 1997.

68. Самарский А. А., Гулин А. В. Численные методы. — М.: Наука, 1989.

69. Саукап Рон. Проектирование реляционных систем баз данных. — М.: Русская редакция, 1998.

70. Свами М., Тхуласираман К. Графы, сети и алгоритмы. — М.: Мир, 1984.

71. Системы управления базами данных и знаниями / Под ред. А. Н. Наумова. — М.: Финансы и статистика, 1998.

72. Спортак М., Паппас Ф. и др. Компьютерные сети и сетевые технологии. — Киев: ООО "ТИД "ДС", 2002.

73. Спортак М. Компьютерные сети. Кн. I и II. — М.: Диасофт, 2002.

74. Судоплатов С. В., Овчинникова Е. В. Элементы дискретной математики. — М.: ИНФРА-М, 2002.

75. Теория и практика обеспечения информационной безопасности / Под ред. П. Д. Зегжды. — М.: Яхтсмен, 1996.

76. Толковый словарь по искусственному интеллекту. — М.: Радио и связь, 1996.