Министерство образования Республики Беларусь

Учреждение образования
«Полоцкий государственный университет»

ИНФОРМАТИКА И КОМПЬЮТЕРНАЯ ГРАФИКА

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС
(в 2-х частях)

для студентов специальности

48 01 03 «Химическая технология природных энергоносителей и углеродных материалов»;

ЧАСТЬ 1. КУРС ЛЕКЦИЙ

Составление и общая редакция
А.В. Спиридонова

Новополоцк 2011

УДК 004 (075.8)
ББК 32.97 я 73
О 75

РЕЦЕНЗЕНТЫ:

А.В. Дубровский, канд. техн. наук, начальник отдела
подготовки кадров завода «Полимир» ОАО «Нафтвн» г. Новополоцка;
А.А. Ермак, канд. техн. наук, доцент кафедры
химической технологии топлива и углеродных материалов

Рекомендован к изданию методической комиссией технологического факультета

О 75 **Информатика и компьютерная графика:** Учеб.-метод. комплекс для студ. спец. 48 01 03 «Химическая технология природных энергоносителей и углеродных материалов» / Сост. и общ. ред. А.В. Спиридонова. – Новополоцк: ПГУ, 2011. – 340 с.
ISBN 985-418-312-2

Приведены темы изучаемого курса, объем в часах лекционных и лабораторных занятий. Представлены методические указания и задания к лабораторным работам, сборник тестов по темам курса, вопросы к экзамену, рекомендации по организации контроля изучения дисциплины.

Предназначен для преподавателей и студентов вузов химико-технологических специальностей.

УДК 004 (075.8)
ББК 32.97 я 73

ISBN 985-418-312-2

СОДЕРЖАНИЕ

РАБОЧАЯ ПРОГРАММА

2.1. НАИМЕНОВАНИЕ ТЕМ ЛЕКЦИОННЫХ ЗАНЯТИЙ, ИХ СОДЕРЖАНИЕ

№ пп	НАИМЕНОВАНИЕ И СОДЕРЖАНИЕ ТЕМ
1	2
1	ВВЕДЕНИЕ В ИНФОРМАТИКУ И ВЫЧИСЛИТЕЛЬНУЮ ТЕХНИКУ. Определение и основные понятия информатики. Предмет и основная задача информатики. Виды и свойства информации. Восприятие, сбор, передача, обработка и накопление информации. Классификация ЭВМ. Представление информации в ЭВМ. Устройство персонального компьютера (базовая конфигурация). Носители информации. Периферийные устройства персонального компьютера. Программное обеспечение средств вычислительной техники. Назначение, классификация и основные функции операционных систем. Программы: технического обслуживания, архиваторы и антивирусы.
2	ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ. Понятие и основные свойства алгоритма. Языки программирования высокого уровня. Введение в программирование на языке паскаль. Константы, переменные и понятие типа данных. Операторы языка и их аналогии в алгоритмических структурах. Программирование линейных алгоритмов. Программирование разветвляющихся алгоритмов. Программирование безусловного перехода, алгоритмов множественного выбора и циклических алгоритмов. Одномерные массивы. Описание и вызов процедур и функций. Рекурсивные подпрограммы. Работа с двумерными массивами. Стандартный модуль Graph.
3	МЕТОДЫ И АЛГОРИТМЫ ЧИСЛЕННОГО РЕШЕНИЯ УРАВНЕНИЙ. Основы теории погрешностей. Приближенное решение нелинейных уравнений. Методы решения систем линейных уравнений. Приближенное вычисление определенных интегралов. Приближенное решение дифференциальных уравнений. Интерполирование функций. Регрессия.
4	ТАБЛИЧНЫЙ ПРОЦЕССОР MICROSOFT EXCEL. Работа с книгами Microsoft Excel. Форматирование данных. Организация вычислений. Диаграммы. Управление данными. Матричные операции и решение систем линейных алгебраических уравнений средствами MS EXCEL. Решение некоторых классов математических задач в MS EXCEL.
5	МАТЕМАТИЧЕСКИЙ ПАКЕТ MCAD. Детали интерфейса. Алфавит входного языка системы MathCAD. Простые вычисления арифметических выражений и их редактирования. Работа с графикой. Работа с векторами и матрицами. Решение уравнений. Выполнение символьных вычислений. Нахождение экстремумов функций. Аппроксимация функций. Вычисление определенных интегралов. Решение дифференциальных уравнений. Решение уравнений в частных производных.

6	КОМПЬЮТЕРНЫЕ СЕТИ. Назначение и основные понятия. Краткая история развития Internet. Службы Internet. Поиск информации в Internet.
7	КОМПЬЮТЕРНАЯ ГРАФИКА. СРЕДСТВА ПОДГОТОВКИ ПРЕЗЕНТАЦИЙ

2.2 ЛАБОРАТОРНЫЕ ЗАНЯТИЯ

№ пп	НАИМЕНОВАНИЕ И СОДЕРЖАНИЕ ТЕМ
1	ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ.
2	МЕТОДЫ И АЛГОРИТМЫ ЧИСЛЕННОГО РЕШЕНИЯ УРАВНЕНИЙ.
3	ТАБЛИЧНЫЙ ПРОЦЕССОР MICROSOFT EXCEL.
4	МАТЕМАТИЧЕСКИЙ ПАКЕТ MCAD.
5	КОМПЬЮТЕРНЫЕ СЕТИ.
6	КОМПЬЮТЕРНАЯ ГРАФИКА. СРЕДСТВА ПОДГОТОВКИ ПРЕЗЕНТАЦИЙ

1. ВВЕДЕНИЕ В ИНФОРМАТИКУ И ВЫЧИСЛИТЕЛЬНУЮ ТЕХНИКУ

1.1. Определение и основные понятия информатики. Предмет и основная задача информатики

Информатика – это техническая наука, систематизирующая приемы создания, хранения, воспроизведения, обработки и передачи информации средствами вычислительной техники (ВТ), а также принципы функционирования этих средств и методы управления ими.

Слово информатика происходит от объединения терминов *Information* (информация) и *Automatique* (автоматика), что выражает ее суть как науки об автоматической обработке информации.

Предмет информатики составляют следующие понятия:

– аппаратное обеспечение и программное обеспечение средств ВТ;

– средства взаимодействия аппаратного и программного обеспечения;

– средства взаимодействия человека с аппаратными и программными средствами.

Методы и средства взаимодействия человека с аппаратными и программными средствами называют пользовательским *интерфейсом*. *Основной задачей* информатики является систематизация приемов и методов работы с аппаратными и программными средствами ВТ. К основным понятиям информатики относятся следующие понятия:

1. *Информация* – это отражение реального мира с помощью знаков и сигналов. В узком смысле под информацией понимают те явления, которые человек получает из окружающего мира. Понятие «информация» тесно связано с понятием «информационные системы».

2. *Информационные системы* выполняют технологические функции по сбору, накоплению, хранению и обработке информации.

3. *Информационные технологии* – целенаправленный процесс преобразования информации, использующий совокупность средств и методов сбора, обработки, хранения и передачи информации.

4. *Информационные ресурсы* – информация, используемая на производстве, в технике, управлении обществом, специально организованная и обработанная на ЭВМ.

5. *Инфосфера* – совокупное информационное пространство.

6. *Информатизация общества* – повсеместное внедрение комплекса мер, направленных на обеспечение полного и своевременного использования достоверной информации, и зависит от степени освоения и развития новых информационных технологий.

1.2. Виды и свойства информации

Все многообразие окружающей нас информации можно сгруппировать по различным признакам. По признаку «область возникновения» информация делится на:

– *элементарную* – отражает процессы и явления неодушевленной природы;

– *биологическую* – отражает процессы растительного и животного мира;

– *социальную* – отражает процессы человеческого общества.

По способу передачи и восприятия различают информацию:

– *визуальную* – передается видимыми образами и символами;

– *аудиальную* – передается звуками;

– *тактильную* – передается ощущениями;

– *органолептическую* – передается запахами и вкусом;

– *машинную* – выдаваемую и воспринимаемую средствами ВТ.

Информацию, создаваемую и используемую человеком, по общественному назначению делят на виды (рис. 1.1).

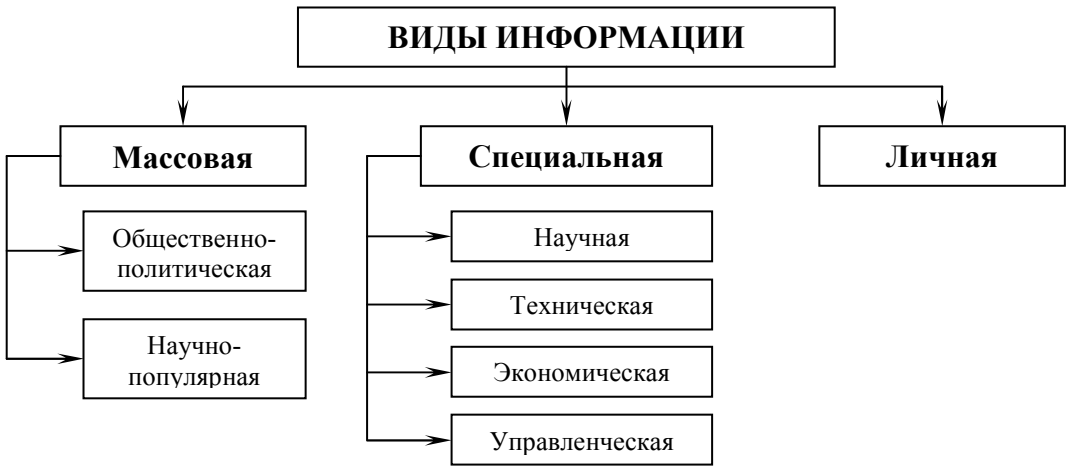


Рис. 1.1. Виды информации

В информатике рассматривают две формы представления информации:

– *аналоговая* (непрерывная) – температура тела; мелодия, извлекаемая на скрипке, когда смычок не отрывается от струн и не останавливается; движение автомобиля;

– *дискретная* (прерывистая) – времена года, точка и тире в азбуке Морзе.

Информация обладает рядом свойств:

– *адекватность* – т. е. степень соответствия информации, полученной потребителем, тому, что автор вложил в ее содержание;

– *достоверность* – соответствие информации объективной реальности (как текущей, так и прошедшей) окружающего мира;

– *полнота* – т. е. достаточность информации для принятия решения. С понятием полноты информации сталкиваются все, кому приходится выполнять служебные задания. Если исходные данные неполны, принять верное решение непросто;

– *избыточность* – это качество позволяет человеку меньше напрягать свое внимание и меньше утомляться;

– *объективность и субъективность* – понятие объективности информации является относительным. Так, например, принято считать, что в результате наблюдения фотоснимка объекта образуется более объективная информация, чем в результате наблюдения рисунка того же объекта, выполненного человеком;

– *доступность* – это мера возможности получить ту или иную информацию;

– *актуальность* – это степень соответствия информации текущему моменту времени.

1.3. Восприятие, сбор, передача, обработка и накопление информации

Восприятие информации – процесс преобразования сведений, поступающих в техническую систему или живой организм из внешнего мира, в форму, пригодную для дальнейшего использования. Благодаря восприятию информации обеспечивается связь системы с внешней средой (в качестве которой могут выступать человек, наблюдаемый объект, явление или процесс и т. д.). Восприятие информации необходимо для любой информационной системы.

Сбор информации – это процесс получения информации из внешнего мира и приведение ее к стандарту для данной информационной системы. Обмен информацией между воспринимающей ее системой и окружающей средой осуществляется посредством сигналов. *Сигнал* можно определить как средство перенесения информации в пространстве и времени. В качестве носителя сигнала могут выступать звук, свет, электрический ток, магнитное поле и т.п. Сбор информации, как правило, сопровождается ее регист-

рацией, т. е. фиксацией информации на материальном носителе (документе или машинном носителе).

Передача информации осуществляется различными способами: с помощью курьера, пересылка по почте, доставка транспортными средствами, дистанционная передача по каналам связи. Дистанционная передача по каналам связи сокращает время передачи данных. Для ее осуществления необходимы специальные технические средства. Некоторые технические средства сбора и регистрации, собирая автоматически информацию с датчиков, установленных на рабочих местах, передают ее в ЭВМ. Поступление информации по каналам связи в центр обработки в основном осуществляется двумя способами: на машинном носителе и непосредственно в ЭВМ при помощи специальных программных и аппаратных средств.

В современных развитых информационных системах *машинная обработка информации* предполагает последовательно-параллельное во времени решение вычислительных задач. Это возможно при наличии определенной организации вычислительного процесса. Вычислительная задача по мере необходимости обращается с запросами в вычислительную систему. Организация процесса предполагает определение последовательности решения задач и реализацию вычислений. Последовательность решения задается, исходя из их информационной взаимосвязи, когда результаты решения одной задачи используются, как исходные данные для решения другой.

Технология электронной обработки информации – человеко-машинный процесс исполнения взаимосвязанных операций, протекающих в установленной последовательности с целью преобразования исходной (первичной) информации в результатную. Операция представляет собой комплекс совершаемых технологических действий, в результате которых информация преобразуется. Технологические операции разнообразны по сложности, назначению, технике реализации, выполняются на различном оборудовании многими исполнителями.

Хранение и накопление информации вызвано многократным ее использованием, применением постоянной информации, необходимостью комплектации первичных данных до их обработки.

В современном понимании *компьютер – это универсальное электронное устройство, предназначенное для автоматизации создания, хранения, обработки, транспортировки и воспроизведения данных*.

Совокупность устройств, предназначенных для автоматической или автоматизированной обработки данных, называют *вычислительной техникой*. Конкретный набор взаимодействующих между собой устройств и программ, предназначенный для обслуживания одного рабочего участка, назы-

вают *вычислительной системой*. Центральным устройством большинства вычислительных систем является *компьютер*.

1.4. Классификация ЭВМ

1.4.1. 1.4.1. Классификация по назначению

Большие ЭВМ. Это самые мощные компьютеры. Их применяют для обслуживания очень крупных организаций и целых отраслей народного хозяйства. За рубежом компьютеры этого класса называют *мэйнфреймами* (*mainframe*). В России за ними закрепился термин *большие ЭВМ*. Штат обслуживания большой ЭВМ составляет до многих десятков человек. На базе таких суперкомпьютеров создают *вычислительные центры*, включающие в себя несколько отделов или групп.

Мини-ЭВМ. От больших ЭВМ компьютеры этой группы отличаются уменьшенными размерами и, соответственно, меньшей производительностью и стоимостью. Такие компьютеры используются крупными предприятиями, научными учреждениями, банками и некоторыми высшими учебными заведениями, сочетающими учебную деятельность с научной. Для организации работы с мини-ЭВМ тоже требуется специальный вычислительный центр, хотя и не такой многочисленный, как для больших ЭВМ.

Микро-ЭВМ. Компьютеры данного класса доступны многим предприятиям. Организации, использующие микроЭВМ, обычно не создают вычислительные центры. Для обслуживания такого компьютера им достаточно небольшой вычислительной лаборатории в составе нескольких человек. Несмотря на относительно невысокую производительность по сравнению с большими ЭВМ, микро-ЭВМ находят применение и в крупных вычислительных центрах. Там им поручают вспомогательные операции, для которых нет смысла использовать дорогие суперкомпьютеры.

Персональные компьютеры (ПК). Начиная с 1999 г. в области ПК начал действовать международный сертификационный стандарт – *спецификация PC99*. Он регламентирует принципы классификации ПК и оговаривает минимальные и рекомендуемые требования к каждой из категорий. Новый стандарт установил следующие категории персональных компьютеров:

- Consumer PC (массовый ПК);
- Office PC (деловой ПК);
- Mobile PC (портативный ПК);
- Workstation PC (рабочая станция);
- Entertainment PC (развлекательный ПК).

Согласно спецификации *PC99* большинство ПК, присутствующих в настоящее время на рынке, попадают в категорию *массовых*. Для *деловых ПК* минимизированы требования к средствам воспроизведения графики, а к средствам работы со звуковыми данными вообще не предъявляются. Для *портативных ПК* обязательным является наличие средств для создания соединений удаленного доступа, то есть средств компьютерной связи. В категории *рабочих станций* повышены требования к устройствам хранения данных, а в категории *развлекательных ПК* – к средствам воспроизведения графики и звука.

1.4.2. 1.4.2. Классификация по уровню специализации

По уровню специализации компьютеры делятся на *универсальные* и *специализированные*. На базе универсальных компьютеров можно собирать вычислительные системы произвольного состава (состав компьютерной системы называется *конфигурацией*). Так, например, один и тот же ПК можно использовать для работы с текстами, музыкой, графикой, фото- и видеоматериалами.

Специализированные компьютеры предназначены для решения конкретного круга задач. К таким компьютерам относятся, например, бортовые компьютеры автомобилей, судов, самолетов, космических аппаратов.

1.4.3. 1.4.3. Классификация по типоразмерам

По типоразмерам ПК можно классифицировать следующим образом: *настольные* (*desktop*), *портативные* (*notebook*), *карманные* (*palmtop*).

Настольные модели распространены наиболее широко. Они являются принадлежностью рабочего места. Эти модели отличаются простотой изменения конфигурации за счет несложного подключения дополнительных внешних устройств или установки дополнительных внутренних компонентов. Достаточные размеры корпуса в настольном исполнении позволяют выполнять большинство подобных работ без привлечения специалистов, а это позволяет настраивать компьютерную систему оптимально для решения именно тех задач, для которых она была приобретена.

Портативные модели зачастую используют бизнесмены, коммерсанты, руководители предприятий и организаций, проводящие много времени в командировках и разъездах. С портативным компьютером можно работать при отсутствии рабочего места.

Карманные модели выполняют функции «интеллектуальных записных книжек». Они позволяют хранить оперативные данные и получать к ним быстрый доступ.

1.4.4. 1.4.4. Классификация по совместимости

Аппаратная совместимость. В области ПК сегодня наиболее широко распространены две аппаратные платформы: *IBM PC* и *Apple McIntosh*. Кроме них существуют и другие платформы, распространенность которых ограничивается отдельными регионами или отдельными отраслями. Принадлежность компьютеров к одной аппаратной платформе повышает совместимость между ними, а принадлежность к разным платформам – понижает.

Кроме аппаратной совместимости существуют и другие виды совместимости: *совместимость на уровне операционной системы* (ОС), *программная совместимость*, *совместимость на уровне данных*.

1.5. Представление информации в ЭВМ

1.5.1. 1.5.1. Кодирование информации

Кодирование данных двоичным кодом. Своя система кодирования существует и в ВТ – она называется *двоичным кодированием* и основана на представлении данных последовательностью всего двух знаков: 0 и 1. Эти знаки называются *двоичными цифрами*, по-английски – *binary digit*, или, сокращенно, *bit* (*бит*).

Кодирование целых и действительных чисел. Для кодирования целых чисел от 0 до 255 достаточно иметь 8 разрядов двоичного кода (8 бит). Комбинация из 8 бит называется *байтом*.

0000 0000 = 0
0000 0001 = 1
.....
1111 1110 = 254
1111 1111 = 255

Шестнадцать бит позволяют закодировать целые числа от 0 до 65535, и 24 бита – уже более 16,5 миллионов разных значений. Для действительных чисел используют 80-разрядное кодирование.

1.5.2. 1.5.2. Системы счисления

Система счисления – это совокупность цифровых знаков и правил их записи, применяемая для однозначной записи чисел. Все системы счисления подразделяются на *позиционные* и *непозиционные*.

Непозиционной называется такая система счисления, в которой значение цифры не зависит от ее положения в ряду цифр, изображающих число.

Примером является римская система счисления. Цифры в римской системе обозначаются различными знаками: 1 – I; 3 – III; 5 – V; 10 – X; 50 – L; 100 – C; 500 – D; 1000 – M. Так, XC – 90; CX – 110; MCMLXXXVIII – 1988. Выполнять арифметические действия в непозиционных системах неудобно. Поэтому в настоящее время эти системы не используются для расчетов.

Позиционной называется такая система счисления, в которой значение цифры зависит от ее положения в ряду цифр, изображающих число, т. е. *веса*. В десятичной системе счисления вес каждой последующей цифры в 10 раз больше веса предыдущей. Например, цифра 2 в числе 1235 имеет значение 200, так как она расположена в третьей справа позиции числа.

В ЭВМ применяют ПСС с недесятичным основанием: двоичную, восьмеричную, шестнадцатеричную и др. В табл. 1.1 показано соответствие записи чисел в десятичной, двоичной, восьмеричной и шестнадцатеричной.

Пример 1. Число 321_{10} запишем в двоичной системе счисления. Для этого необходимо разложить число в виде суммы по степеням 2:

$$321_{10} = 1 \cdot 2^8 + 1 \cdot 2^6 + 1 \cdot 2^0$$

Пример 2. Число 10100101_2 перевести в десятичную систему счисления:

$$10100101_2 = 1 \cdot 2^0 + 1 \cdot 2^2 + 1 \cdot 2^5 + 1 \cdot 2^7 = 165_{10}.$$

Таблица 1.1

Соответствие записи чисел в различных системах счисления

X(10)	X(2)	X(8)	X(16)
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

1.5.3. 1.5.3. Единицы измерения информации

Наименьшей единицей измерения информации является *байт*. Поскольку одним байтом, как правило, кодируется один символ текстовой информации. Более крупная единица измерения – *килобайт* (Кбайт). 1 Кбайт = 2^{10} байт = 1024 байт. В килобайтах измеряют сравнительно небольшие объемы данных.

Более крупные единицы:

1 Мбайт (Мегабайт) = 1024 Кбайт = 2^{20}байт;

1 Гбайт (Гигабайт) = 1024 Мбайт = 2^{30}байт;

1 Тбайт (Терабайт) = 1024 Гбайт = 2^{40}байт.

1.6. Устройство персонального компьютера (базовая конфигурация)

Персональный компьютер – универсальная техническая система. Его *конфигурацию* (состав оборудования) можно гибко изменять по мере необходимости. Существует также понятие *базовой конфигурации*. В таком комплекте компьютер обычно поставляется. Понятие базовой конфигурации может меняться. В настоящее время в базовой конфигурации рассматривают четыре устройства: системный блок, монитор, клавиатура, манипулятор «мышь».

1.6.1. 1.6.1. Системный блок

Системный блок – это самая главная часть компьютера, внутри которой установлены наиболее важные компоненты. Устройства, находящиеся внутри системного блока, называют *внутренними*, а устройства, подключаемые к нему снаружи, – *внешними*. Внешние дополнительные устройства, предназначенные для ввода, вывода и длительного хранения данных, также называют *периферийными*. По внешнему виду системные блоки различаются формой корпуса. Корпуса ПК выпускают в горизонтальном (*desktop*) и вертикальном (*tower*) исполнении.

В системном блоке размещаются следующие элементы:

– электронные схемы, управляющие работой компьютера (микропроцессор, оперативная память, контроллеры устройств и т. д.);

– дисководы (приводы) для гибких магнитных дисков, компакт-дисков (CD-ROM);

– жесткий магнитный диск (*винчестер*);

– блок питания, преобразующий переменное сетевое напряжение в ряд постоянных, необходимых для работы электронных схем компьютера;

– прочие устройства, которые могут быть установлены в системном блоке.

Своеобразной базой ПК является *материнская плата*, на основе которой можно получить множество вариантов ЭВМ, наилучшим образом приспособленных для того или иного рода работ. На материнской плате располагаются различные устройства и микросхемы.

Микропроцессор – основная микросхема, выполняющая большинство математических и логических операций. В компьютерах IBM PC и совместимых чаще всего используются микропроцессоры фирм Intel и AMD. Важнейшим показателем микропроцессора является его *тактовая частота*, т. е. количество элементарных операций в секунду. Чем выше тактовая частота, тем выше производительность процессора.

Оперативная память (ОЗУ) – набор микросхем, предназначенных для временного хранения данных, когда компьютер включен. Из ОЗУ процессор берет программы и исходные данные для обработки, в нее же записывает результат. При выключении компьютера ОЗУ очищается, и записанная в него информация уничтожается.

Шины – наборы проводников, по которым происходит обмен сигналами между внутренними устройствами компьютера.

Микросхема ПЗУ (постоянное запоминающее устройство) – микросхема, предназначенная для длительного хранения данных, в т. ч. и когда компьютер выключен. Комплект программ, находящихся в ПЗУ, образует базовую систему ввода-вывода (*BIOS – Basic Input-Output System*). Основное назначение программ этого пакета состоит в том, чтобы проверить состав и работоспособность компьютерной системы и обеспечить взаимодействие с клавиатурой, монитором, жестким диском и дисководом гибких дисков. Программы, входящие в BIOS, позволяют наблюдать на экране диагностические сообщения, сопровождающие запуск компьютера, а также вмешиваться в ход запуска с помощью клавиатуры.

Звуковая карта (плата). Как правило, звуковые карты обеспечивают возможность как цифро-аналогового, так и обратного преобразования звуковых колебаний, а также позволяют создавать некоторые звуковые эффекты.

В разъемы системной шины материнской платы встраиваются контроллеры различных устройств (монитора, CD-ROM и т. д.). Разъемы для внешних устройств выводятся на заднюю панель системного блока.

На лицевой панели обычно выведены отсеки для приводов внешних запоминающих устройств, кнопки пуска и перезагрузки, светодиодные индикаторы питания и жесткого диска, а также (не всегда) цифровой индикатор тактовой частоты установленного процессора.

1.6.2. 1.6.2. Мониторы

Монитор – это устройство вывода графической и текстовой информации, основанное на преобразовании электрического видеосигнала в видимое изображение. Монитор подключается к компьютеру через особую плату, находящуюся внутри компьютера и называемую *видеокартой*. В настоящее время существует два вида мониторов: с электронно-лучевой трубкой (ЭЛТ (CRT)-мониторы) и жидкокристаллические (LCD-мониторы). Приведем основные характеристики мониторов.

Размер экрана – расстояние между противоположными углами экрана по диагонали. Обычно определяется в дюймах (14, 15, 17, 19, 20, 21). Для сложных чертежей и графического дизайна используются в основном мониторы с размером экрана более 17 дюймов.

Шаг сетки – минимальный размер одной точки экрана (пиксель – pixel). Обычно он составляет 0,25 мм. Чем мельче зерно, тем лучше качество изображения.

Разрешение – количество точек на экране, которые могут уместиться при образовании изображения (640×480, 800×600, 1024×768).

Частота кадровой развертки – частота управляющих сигналов, указывающих на необходимость перехода к новому кадру. Она важна не только для компьютера, но и для пользователя. При малой частоте смены кадров пользователь видит своеобразное мерцание экрана, что негативно влияет на зрение. Только при частоте кадровой развертки, равной или превышающей 75-80 Гц, это мерцание пропадает.

1.6.3. 1.6.3. Клавиатура

Клавиатура – это клавишное устройство управления ПК. Служит для ввода данных, а также команд управления. Комбинация монитора и клавиатуры обеспечивает простейший интерфейс пользователя. Стандартная клавиатура имеет более 100 клавиш, функционально распределенных по следующим группам:

– *алфавитно-цифровые*. Для ввода знаковой информации и команд;

– *функциональные (F1-F12)*. Их назначение может быть различно в различных программах;

– *модификации кодов (Shift, Ctrl, Alt, Caps Lock)*. Изменяют стандартное назначение клавиш других групп;

– *клавиши управления курсором*. Управляют позицией ввода данных;

– *клавиши дополнительной панели*. Дублируют действия клавиш других групп.

1.6.4. 1.6.4. Мышь

Мышь – устройство управления манипуляторного типа. Существуют мыши с двумя или тремя кнопками и, возможно, дополнительными органами управления. Перемещение мыши по плоской поверхности синхронизировано с перемещением указателя мыши на экране монитора.

Комбинация монитора и мыши обеспечивает графический интерфейс пользователя. Пользователь наблюдает на экране графические объекты и элементы управления. С помощью мыши он изменяет свойства объектов и приводит в действие элементы управления компьютерной системой, а с помощью монитора получает от нее отклик в наглядном виде.

В зависимости от принципа устройства, мыши делятся на *механические*, *оптомеханические* и *оптические*. В механических перемещение шарика внутри отслеживается механическими датчиками (колесиками), в оптомеханических также перемещается шарик, но его положение отслеживается оптическими датчиками, а в оптических – движущихся частей нет вообще.

1.7. Носители информации

1.7.1. 1.7.1. Жесткий диск

Жесткий диск (винчестер, Hard Disc Drive, HD) – основное устройство для долговременного хранения больших объемов данных и программ. Жесткий диск представляет собой группу соосных дисков, имеющих магнитное покрытие и вращающихся с высокой скоростью. Таким образом, этот диск имеет не две поверхности, как должно быть у обычного плоского диска, а $2n$ поверхностей, где n – число отдельных дисков в группе. Над каждой поверхностью располагается головка, предназначенная для чтения-записи данных.

К основным параметрам жестких дисков относятся *емкость* и *производительность*. Емкость дисков зависит от технологии их изготовления. В настоящее время технологический уровень приближается к 40 – 80 Гбайт на пластину. Что касается производительности, то она характеризуется *скоростью внутренней передачи данных* и *средним временем доступа*. Сегодня все жесткие диски имеют очень высокий показатель скорости внутренней передачи данных. Параметр среднего времени доступа определяет интервал времени, необходимый для поиска нужных данных, и зависит от скорости вращения диска. Для дисков, вращающихся с частотой 7200 об/мин – 7 – 8 мкс. Изделия более высокого уровня обеспечивают среднее время доступа к данным 5 – 6 мкс.

1.7.2. 1.7.2. Гибкие магнитные диски

Гибкие магнитные диски (дискеты, флоппи-диски) используются, как правило, для оперативного переноса небольших объемов данных. Дискеты бывают 5- и 3-дюймовые. 5-дюймовые дискеты в последнее время практически не используются. 3-дюймовые дискеты имеют емкость 1,44 Мбайт.

1.7.3. 1.7.3. CD-ROM и DVD-ROM

CD-ROM – постоянное запоминающее устройство на основе компакт-диска. Принцип действия этого устройства состоит в считывании числовых данных с помощью лазерного луча, отражающегося от поверхности диска. Для CD-ROM характерна высокая плотность записи данных. Стандартный компакт-диск может хранить ~700 Мбайт данных. Обычный компакт-диск *CD-R* применим только в качестве архивного носителя, так как перезаписать на него информацию невозможно. В настоящее время широкое распространение получили *перезаписываемые компакт-диски* (*CD-RW*). Несмотря на более высокую стоимость устройства записи (*CD-ReWriter*) и самого диска, возможность осуществлять перезапись информации (до 1000 перезаписей) может быть значительным преимуществом.

1.8. Периферийные устройства персонального компьютера

1.8.1. 1.8.1. Принтеры

Принтеры – это устройства для вывода графической и текстовой информации на бумагу или прозрачный носитель.

Матричные принтеры. *Матричные принтеры* – печатают специальными иголками, ударяющими по красящей ленте. В результате образуется комбинация точек, изображающая символ. Матричные принтеры медленные и шумны, обеспечивают наихудшее качество печати, но цена отпечатанной ими страницы минимальна.

Струйные принтеры. Изображение в струйных принтерах формируется из пятен, образующихся при попадании капель специальных чернил на бумагу. Струйные принтеры довольно быстрые и бесшумные, обеспечивают неплохое качество печати на бумаге определенного сорта. Стоимость отпечатанной страницы у них выше, чем у матричных принтеров. Важнейшей особенностью струйной печати является возможность создания высококачественного цветного изображения.

Лазерные принтеры. Эта группа принтеров обладает наивысшим качеством печати, близким к типографскому. Лазерные принтеры перено-

сят изображение на бумагу при помощи светочувствительного барабана, к освещенным лазером участкам которого прилипает тонер. Затем в результате соприкосновения бумажного листа с барабаном происходит перенос тонера на бумагу. Лазерные принтеры отличаются высокой скоростью печати, бесшумны.

1.8.2. 1.8.2. Устройства ввода графических данных

Для ввода графической информации в компьютер используют *сканеры*, *дигитайзеры*, *цифровые фотокамеры*. *Сканеры* можно также использовать для ввода знаковой информации. В этом случае исходный материал вводится в графическом виде, после чего обрабатывается специальными программными средствами – *программами распознавания образов*.

Основные виды сканеров:

– *планшетные*. Предназначены для ввода графической информации с прозрачного и непрозрачного листового материала;

– *барабанные*. Предназначены для сканирования исходных изображений, имеющих высокое качество, но недостаточные линейные размеры (фотонегативов, слайдов и т. п.);

– *сканеры форм*. Предназначены для ввода данных со стандартных форм, заполненных механически или вручную. Необходимость в этом возникает при проведении переписи населения, обработке результатов голосований и анализе анкетных данных;

– *штрих-сканеры*. Предназначены для ввода данных, закодированных в виде штрих-кода. Такие устройства имеют применение в розничной торговой сети.

Дигитайзеры. Эти устройства предназначены для ввода художественной графической информации. В основе принципа действия этих устройств лежит фиксация перемещения специального пера относительно планшета. Такие устройства удобны для художников и иллюстраторов, так как позволяют им создавать экранные изображения привычными приемами и инструментами (карандаш, перо, кисть).

Цифровые фотокамеры. Основным параметром цифровых фотоаппаратов является разрешающая способность. Наилучшие потребительские модели в настоящее время обеспечивают разрешение 1920×1600 точек и более. У профессиональных моделей эти параметры еще выше.

1.8.3. 1.8.3. Модемы

Модемы – это устройства, предназначенные для обмена информацией между удаленными компьютерами по каналам связи (проводным, оптоволоконным, кабельным, радиочастотным линиям). Модем необходим для подключения к сети Internet, а также для обеспечения внутрикорпоративной связи, создания локальных сетей и т. д. По своему исполнению модемы бывают *внешними* (*External*), т. е. подключаемыми к системному блоку снаружи, и *внутренними* (*Internal*), т. е. встроенными в системный блок.

К основным потребительским параметрам модемов относится производительность (бит/с). От производительности зависит объем данных, передаваемых в единицу времени. Максимальная скорость передачи данных у модемов может быть от 2400 до 115200 бит/с.

1.9. Программное обеспечение средств вычислительной техники

Программы – это упорядоченные последовательности команд. Конечная цель любой компьютерной программы – управление аппаратными средствами. Программное и аппаратное обеспечение ПК работают в неразрывной связи и непрерывном взаимодействии. Состав программного обеспечения ПК называют *программной конфигурацией*. Все программное обеспечение (ПО) подразделяется на несколько уровней: базовый, системный, служебный, прикладной.

1.9.1. 1.9.1. Базовый уровень

Программы этого уровня отвечают за взаимодействие с *базовыми аппаратными средствами*. Базовые программные средства, как правило, входят в состав базового оборудования и хранятся в специальных микросхемах, называемых *постоянными запоминающими устройствами* (ПЗУ). Программы и данные записываются в эти микросхемы на этапе производства и не могут быть изменены в процессе эксплуатации.

1.9.2. 1.9.2. Системный уровень

Системный уровень – переходный. Программы этого уровня обеспечивают взаимодействие прочих программ компьютерной системы с программами базового уровня и непосредственно с аппаратным обеспечением. К программам системного уровня относятся *драйверы устройств* (отвечают за взаимодействие других программ с устройствами компьютера) и программные средства для обеспечения пользовательского интерфейса.

Совокупность программного обеспечения системного уровня образует *ядро операционной системы компьютера*. То есть, наличие ядра ОС – непременное условие для возможности практической работы человека с вычислительной системой.

1.9.3. 1.9.3. Служебный уровень

Назначение служебных программ (*утилит*) состоит в автоматизации работ по проверке, наладке и настройке компьютерной системы. Эти программы взаимодействуют как с программами базового уровня, так и с программами системного уровня. Во многих случаях они используются для расширения или улучшения функций системных программ.

1.9.4. 1.9.4. Прикладной уровень

Программное обеспечение прикладного уровня представляет собой комплекс прикладных программ, с помощью которых на данном рабочем месте выполняются конкретные.

1.9.5. 1.9.5. Классификация прикладных программных средств

Текстовые редакторы. Основная функция этого класса прикладных программ заключается во вводе и редактировании текстов. С этого класса прикладных программ обычно начинают знакомство с программным обеспечением и на нем отрабатывают первичные навыки взаимодействия с компьютерной системой.

Текстовые процессоры. В отличие от текстовых редакторов позволяют не только вводить и редактировать тексты, но и *форматировать* их. Под форматированием понимают оформление документов путем применения нескольких шрифтовых наборов, использования методов выравнивания текста, встраивания в текстовый документ объектов иной природы (рисунков), а также управления взаимодействием графики и текста.

Графические редакторы. Программы этого класса предназначены для создания и/или обработки графических изображений. Различают *растровые редакторы* (когда графический объект представлен в виде комбинации точек), *векторные редакторы* (когда элементарным объектом изображения является линия) и программные средства для создания и обработки трехмерной графики (*3D-редакторы*).

Системы управления базами данных (СУБД). Базами данных называют огромные массивы данных, организованных в табличные структуры. Основные функции СУБД: создание пустой (незаполненной) структуры базы

данных; предоставление средств ее заполнения, обеспечение возможности доступа к данным, а также предоставление средств поиска и фильтрации.

Электронные таблицы. Электронные таблицы предоставляют комплексные средства для хранения различных типов данных (акцент на числовые данные) и их обработки (акцент на преобразование данных, а не на хранение).

Браузеры. Эти программные средства предназначены для просмотра электронных документов, выполненных в формате *HTML* (документы этого формата используются в качестве Web-документов).

Бухгалтерские системы. Это специализированные программные средства, сочетающие в себе функции текстовых и табличных редакторов, электронных таблиц и систем управления данными. Предназначены для автоматизации подготовки первичных бухгалтерских документов предприятия и их учета, для ведения счетов плана бухгалтерского учета, а также для автоматической подготовки регулярных отчетов по итогам производственной, хозяйственной и финансовой деятельности в форме, принятой для предоставления в налоговые органы, внебюджетные фонды и органы статистического учета.

Финансовые аналитические системы. Программы этого класса используются в банковских и биржевых структурах. Они позволяют контролировать и прогнозировать ситуацию на финансовых, товарных и сырьевых рынках, производить анализ текущих событий, готовить сводки и отчеты.

1.10. Назначение, классификация и основные функции операционных систем

Операционная система – это комплекс программ, которые загружаются при включении компьютера. Операционная система производит диалог с пользователем, осуществляет управление компьютером, его ресурсами (процессоры, память, дисковые накопители, сетевые коммуникационные средства, принтеры и другие устройства), запускает другие (прикладные) программы на выполнение, кроме того, ОС обеспечивает пользователю и прикладным программам удобный *интерфейс* (способ общения) с устройствами компьютера.

Классифицировать ОС для ПК можно следующим образом:

– однозадачные и многозадачные;

– однопользовательские и многопользовательские.

Однозадачные ОС позволяют запустить одну программу в основном режиме. *Многозадачные ОС* позволяют запустить одновременно несколько

программ, которые будут работать параллельно, не мешая друг другу. Большинство современных графических ОС – многозадачные.

Однопользовательские ОС позволяют работать на компьютере только одному человеку. В многопользовательской системе работу можно организовать так, что каждый пользователь будет иметь доступ к информации общего доступа, введя пароль, к личной информации, доступной только ему. Отличием многопользовательских систем является наличие средств защиты информации пользователей от несанкционированного доступа.

1.10.1. 1.10.1. Файловая система

Файловая система – это часть ОС, предназначенная для организации работы с хранящимися на диске данными и обеспечения совместного использования файлов несколькими пользователями и процессами. Файловая система ОС определяет структуру хранения файлов и папок на диске, правила задания имен файлов, допустимые атрибуты, права доступа и др.

Разные ОС могут применять разные файловые системы. Например, операционные системы MS-DOS и Windows 95 могут работать только с файловой системой *FAT 16*. Предельный размер диска для нее равен 2 Гбайт. Если физический диск имеет больший размер, его делят на несколько логических дисков.

Операционные системы Windows 98 и Me работают не только с файловой системой *FAT 16*, но и *FAT 32*. В системе FAT 32 нет практических ограничений на размер жесткого диска. Системы FAT 32 и FAT 16 совместимы сверху вниз, то есть файлы, записанные в системе *FAT 16*, читаются на компьютерах, работающих в системе *FAT 32*, но не наоборот.

Операционная система Windows 2000 и выше может работать в файловых системах *FAT 16*, *FAT 32* и *NTFS*. *NTFS* – специфическая файловая система. Она обеспечивает повышенную скорость работы, но несовместима с ОС Windows 9x.

1.10.2. 1.10.2. Файловая структура

Файл – это область на диске или другом носителе информации, обладающая уникальным собственным именем. Собственное имя файла состоит из двух частей: *собственно имени* и *расширения* (или *типа*). В имени разрешается использовать пробелы и несколько точек. Расширением имени считаются все символы, идущие после последней точки. Использование расширения в имени файла обязательным не является.

Тип файла (*расширение*) определяет его предназначение и способ использования: например, обращение к программному файлу запускает про-

грамму. Помимо готовых к выполнению программ в файлах могут храниться тексты программ, документы, и любые другие данные. Примеры распространенных типов файлов:

– *.exe* – исполнимые;

– *.txt* – текстовые (читаемые);

– *.doc* – документы Microsoft Word;

– *.hlp* – файлы помощи и др.

При выборе имени создаваемого файла необходимо учитывать следующие требования:

– имя файла в папке должно быть уникальным;

– имя файла может иметь длину до 255 символов – такое имя файла называется *длинным*;

– не допускается использование в именах следующих символов \ / * ? : “ | < >;

– не разрешается использовать в качестве имен файлов имена, зарезервированные в ОС под имена устройств: *prn* – принтер; *con* – консоль (при вводе – клавиатура, при выводе – монитор); *nul* – пустое устройство, все операции, указанные для него игнорируются; *lpt1-lpt3* – устройства, подсоединяемые к параллельным портам компьютера; *com1-com3* – устройства, подсоединяемые к последовательным портам;

– при сравнении имен не различается написание прописными и строчными буквами, хотя в списках и полях имена файлов выводятся с учетом того, в каком регистре они были введены.

Путь доступа к файлу начинается с имени устройства и включает все имена папок, через которые он проходит. В качестве разделителя используется символ «\». Собственное имя файла вместе с путем доступа к нему считается *полным именем файла* (рис. 1.2):

Рис. 1.2. Пример записи полного имени файла

Кроме имени и расширения файла ОС хранит для каждого файла дату его создания (изменения) и несколько атрибутов файла. *Атрибуты* – это дополнительные параметры, определяющие свойства файлов:

– *только для чтения* (*Read only*);

– *скрытый* (*Hidden*);

– *системный* (*System*);

– *архивный* (*Archive).*

1.11. Обмен данными в Windows

Буфер обмена – это область оперативной памяти, резервируемая системой Windows для организации обмена данными между приложениями. В любой момент времени в ней можно хранить только один объект. При попытке поместить туда другой объект предыдущий объект перестает существовать. Поэтому буфер обмена не используют для длительного хранения чего-либо. Поместив объект в буфер, немедленно выполняют вставку из буфера в нужное место.

Принцип работы с буфером обмена:

1) открыть папку-источник. Выделить щелчком нужный объект;

2) скопировать или вырезать объект в буфер. В первом случае объект остается в папке-источнике. Во втором случае он удаляется из папки источника, но может некоторое время храниться в буфере;

3) открыть папку-приемник и поместить в нее объект из буфера обмена.

Три указанные операции (*Копировать*, *Вырезать* и *Вставить*) можно выполнять разными способами:

– использовать команды меню *Правка*;

– пользоваться кнопками панели инструментов:  – *вырезать*,  – *копировать*,  – *вставить*;

– использовать комбинации клавиш клавиатуры: `Ctrl` + `C` – *копировать в буфер*; `Ctrl` + `X` – *вырезать в буфер*; `Ctrl` + `V` – *вставить из буфера*.

Эти приемы работают во всех приложениях Windows. Через буфер обмена можно переносить фрагменты текстов из одного документа в другой. Можно также переносить иллюстрации, звукозаписи, видеофрагменты, файлы, папки и вообще любые объекты.

1.12. Программы-архиваторы

Назначение *программ-архиваторов* – экономить место на диске за счет сжатия (упаковки) одного или нескольких исходных файлов в ар-

хивный файл. Программы-архиваторы используются для хранения в упакованном виде больших объемов информации, которая понадобится только в будущем; переноса информации между компьютерами с помощью дискет или электронной почты; создания в сжатом виде резервных копий файлов. В результате работы программ-архиваторов создаются архивные файлы (архивы).

Типовые функции программ-архиваторов состоят в:

– помещении исходных файлов в архив;
– извлечении файлов, удалении файлов из архива;
– просмотре оглавления архива;
– верификации (проверки) архива.

Среди современных программ-архиваторов выделяют: WinRAR (разработка Е. Рошаль), WinZip фирмы Niko Mak Computing и др. Мы рассмотрим использование архиватора WinRAR, отличающегося большой степенью сжатия, работой с длинными именами файлов, удобным интерфейсом. Пользовательский интерфейс WinRAR содержит строку меню, панель инструментов и рабочую область, в которой показаны все файлы текущей папки, (рис. 1.3).

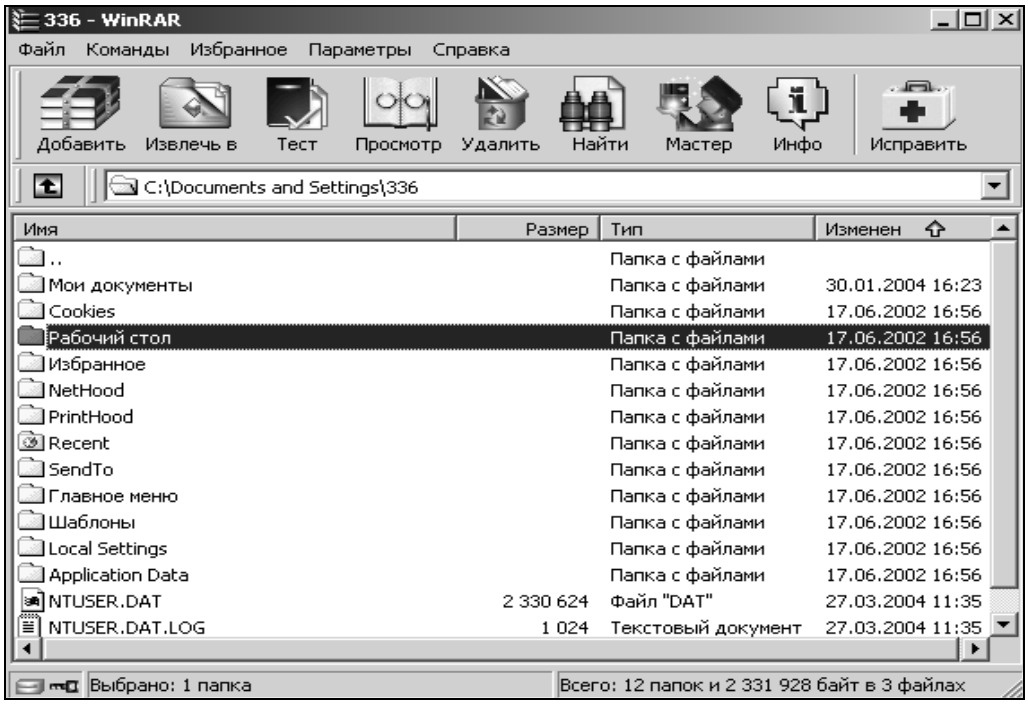


Рис. 1.3. Пользовательский интерфейс архиватора WinRAR

Строка меню архиватора включает пункты *Файл*, *Команды*, *История*, *Избранное*, *Параметры* и *Справка*, содержащих сгруппированные по функциональному назначению команды архиватора.

Команды меню *Файл* выполняют операции над файлами, содержащимися в архиве или помещаемыми в архив.

Команда *Пароль* применяется при установке пароля на вновь создаваемый архивный файл. При выборе этой команды пользователю следует в появившемся окне набрать и подтвердить пароль. Впоследствии без знания этого пароля невозможно будет получить доступ к содержимому хранящихся в архиве файлов.

Следующая группа команд используется для выделения нескольких файлов. Так, команда *Выделить все* автоматически выделяет все файлы текущей папки. Команда *Выделить группу* активизирует маску ввода шаблона файлов, удовлетворяющих некоторому критерию. Аналогично, команда *Снять выделение* вызывает маску шаблона для отмены выделения файлов.

Меню *Команды* содержит команды обработки содержимого архива.

Для создания архива или добавления данных в существующий архив используется команда *Добавить файлы в архив* или кнопка  . При этом в архив помещаются предварительно выделенные файлы. При выборе этой команды на экране появляется диалоговое окно *Имя и параметры архива* (рис. 1.4).

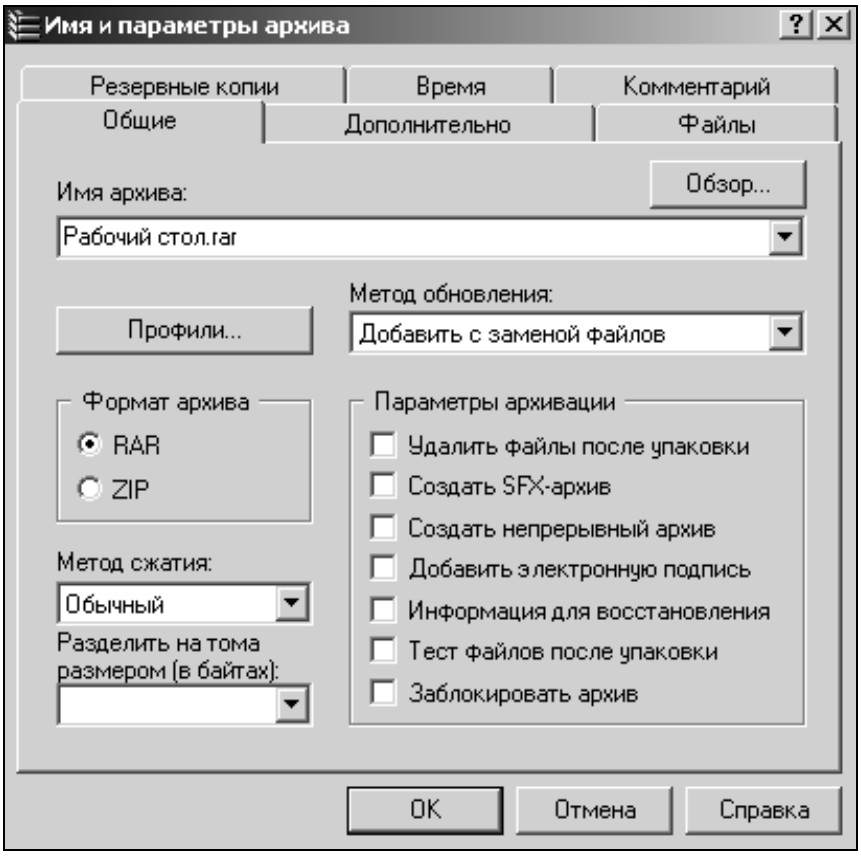


Рис. 1.4. Диалоговое окно *Имя и параметры архива*

В поле *Архив* этого окна указывается местонахождение помещаемых в архив файлов. По умолчанию архив создается в текущей папке. Для указания другой папки следует использовать кнопку Обзор.

Содержимое поля *Метод сжатия* определяет степень сжатия создаваемого архива. По умолчанию установлен вариант *Обычный*. Если установить наилучшую степень сжатия, то архивный файл будет занимать меньший объем, однако время его создания возрастет.

В поле *Размер словаря* задается длина фрагмента, в рамках которого алгоритм сжатия ищет повторения для кодирования и сжатия.

Параметр *Размер тома* применяется при создании многотомных архивов, его установка позволяет создавать архив в виде нескольких файлов, размер которых не превосходит заданного размера тома. Это особенно актуально при необходимости последующего переноса архива на дискетах, если архив превышает емкость имеющихся носителей, или при пересылке по электронной почте.

В поле со списком *Метод обновления* определяются варианты помещения файлов в архив. Устанавливаемый по умолчанию режим *Добавить с заменой файлов* определяет архивирование всех выделенных пользователем файлов. Указание режима *Обновить существующие файлы*, позволяет добавить в архив лишь те файлы, старые версии которых уже находятся в архиве. Режим *Добавить с обновлением файлов* помещает в архив файлы, копии которых в архиве отсутствуют.

Группа команд *Параметры архивации* позволяет выбрать алгоритм сжатия данных. По умолчанию программа настроена на базовый вариант. Однако, например, для мультимедийных данных целесообразно использовать вариант *Мультимедиа-сжатие*.

Когда настройка завершена, следует нажать кнопку OK. Появившееся после этого окно иллюстрирует текущее состояние процедуры архивации (степень обработки очередного файла набора и общее состояние выполнения всей процедуры).

На этом процедура создания архива или добавления данных в существующий архив заканчивается.

Команда *Восстановить архив* используется при нарушении целостности архива, возникающем, например, в результате его длительного хранения.

Архиватор WinRAR позволяет удалять ненужные файлы, как это делается в программе *Проводник*. Для этого используется команда *Удалить файлы* нажатие клавиши Delete на клавиатуре или кнопки 

.

Остальные команды этого меню относятся только к файлам, содержащимся в архиве, и становятся доступными, если в рабочей области WinRAR выделен файл архива.

Команда *Извлечь файлы из архива* обеспечивает распаковку предварительно выделенных пользователем файлов из данного архива. Если необходимо разархивировать файл не в текущую папку, то следует воспользоваться командой *Извлечь в другую папку* и указать путь к этой папке-получателю. Тестирование отдельных файлов в архиве на предмет возможных повреждений их структуры производится с помощью команды *Протестировать файлы в архиве*. Эту команду следует применять для проверки целостности файлов при их длительном хранении, особенно на ненадежных магнитных носителях.

Архив можно снабдить комментарием, воспользовавшись командой *Добавить архивный комментарий* или соответствующей кнопкой 

на панели инструментов. Выбор команды *Добавить информацию для восстановления* вызывает специальную процедуру, которая вносит в текущий архив дополнительные данные, повышающие его устойчивость к сбоям. Однако это приводит к некоторому увеличению объема архива.

Архиватор WinRAR позволяет создавать самораспаковывающиеся архивы, разворачивающиеся при запуске их на исполнение. Для этого требуется выполнить команду *Преобразовать архив в SFX*. Эта команда становится доступной, если в рабочей области WinRAR выделен файл архива.

Выбор команды *Информации об архиве* позволяет получить сведения об архиве, выделенном в данный момент в рабочей области (размер и количество файлов в архиве; коэффициент сжатия архива; наличие комментариев; наличие пароля; ОС, для которой этот архив создан).

Команды меню *История* обеспечивают доступ к последним обрабатываемым архивам, с которыми работал пользователь.

Группа команд меню *Параметры* предназначена для настройки основных параметров архиватора WinRAR и регистрации пользователей через Internet. С помощью этого меню пользователь имеет возможность определить интерфейс архиватора; задать значения по умолчанию основных параметров архиватора (метода сжатия, размера словаря); определить папку, в которую следует помещать файл архива и др.

Команды меню *Помощь* описывают возможности работы архиватора, поясняют технику работы с ним и содержат информацию о разработчике и процедуре приобретения архиватора.

1.13. Компьютерные вирусы и антивирусные программы

Компьютерный вирус – специальная программа, способная самопроизвольно присоединяться к другим программам («заражать» их) и при запуске последних выполнять различные нежелательные действия: порчу файлов и папок, искажение результатов вычислений, засорение или стирание памяти, создание помех в работе компьютера. Наличие вирусов проявляется в следующих ситуациях:

– некоторые программы перестают работать или начинают работать некорректно;

– на экран выводятся посторонние сообщения, сигналы и другие эффекты;

– работа компьютера существенно замедляется;

– структура некоторых файлов оказывается испорченной и т. д.

1.13.1. 1.13.1. Классификация компьютерных вирусов

Имеется несколько признаков классификации существующих компьютерных вирусов:

– по среде обитания;

– по области поражения;

– по особенностям алгоритма;

– по способу заражения;

– по деструктивным возможностям.

Рассмотрим приведенную классификацию более детально.

Классификация вирусов по среде обитания. Различают файловые, загрузочные, макро- и сетевые вирусы.

Файловые вирусы – наиболее распространенный тип вирусов. Эти вирусы внедряются в выполняемые файлы, создают файлы-спутники (companion-вирусы) или используют особенности организации файловой системы (link-вирусы).

Загрузочные вирусы записывают себя в загрузочный сектор диска (boot-сектор) или в сектор системного загрузчика жесткого диска (Master Boot Record). Начинают работу при загрузке компьютера и обычно становятся резидентными (постоянно хранящимися во время работы в оперативной памяти). Как правило, эти вирусы состоят из двух частей, поскольку загрузочная запись имеет небольшой размер и в ней трудно разместить целиком программу вируса.

Макровирусы заражают файлы широко используемых пакетов обработки данных. Эти вирусы представляют собой программы, написанные на встроенных в эти пакеты языках программирования. Наибольшее распро-

странение получили макровирусы для приложений Microsoft Office. Для своего размножения такие вирусы используют возможности встроенного языка Visual Basic for Applications (VBA).

Сетевые вирусы используют для своего распространения протоколы или команды компьютерных сетей и электронной почты. Основным принципом работы сетевого вируса является возможность самостоятельно передать свой код на удаленный сервер или рабочую станцию. Полноценные сетевые вирусы при этом должны обладать возможностью запустить на удаленном компьютере свой код на выполнение.

На практике существуют разнообразные сочетания вирусов – например, файлово-загрузочные вирусы, заражающие как файлы, так и загрузочные сектора дисков, или сетевые макровирусы, которые заражают редактируемые документы и рассылают свои копии по электронной почте.

Классификация вирусов по области поражения. Как правило, каждый вирус заражает файлы одной или нескольких ОС: MS-DOS, Windows, Win95/NT, OS/2, Unix. Макровирусы заражают файлы форматов MS Word, MS Excel и других приложений MS Office. Многие загрузочные вирусы также ориентированы на конкретные форматы расположения системных данных в загрузочных секторах дисков.

Классификация вирусов по особенностям алгоритма. Выделяют резидентные вирусы, стелс-вирусы (stealth – *англ*. невидимка), полиморфные и др.

Резидентные вирусы способны оставлять свои копии (или части) в оперативной памяти, перехватывать обработку событий (например, обращения к файлам или дискам) и вызывать при этом процедуры заражения объектов (файлов и секторов). Эти вирусы активны в памяти не только в момент работы зараженной программы, но и после. Резидентные копии таких вирусов жизнеспособны до перезагрузки ОС, даже если на диске уничтожены все зараженные файлы. От таких вирусов сложно избавиться простым восстановлением копий файлов с дистрибутивных или резервных дисков. Это объясняется тем, что резидентная копия вируса остается активной в оперативной памяти и заражает вновь создаваемые файлы. Если резидентный вирус является также загрузочным и активизируется при загрузке ОС, то даже форматирование диска при наличии в памяти этого вируса его не удаляет. Это объясняет то, что многие резидентные вирусы заражают диск повторно после того, как он отформатирован.

Нерезидентные вирусы, напротив, активны довольно непродолжительное время – только в момент запуска зараженной программы. Для сво-

его распространения они выбирают на диске незараженные файлы и записываются в них. После окончания работы зараженной программы вирус становится неактивным вплоть до очередного запуска какой-либо зараженной программы. Зараженные нерезидентными вирусами файлы восстанавливаются значительно проще.

Стелс-алгоритмы позволяют вирусам полностью или частично скрыть свое присутствие. Наиболее распространенным *стелс-алгоритмом* является перехват запросов ОС на чтение/запись зараженных объектов. Стелс-вирусы при этом либо временно «лечат» эти объекты, либо подставляют вместо себя незараженные участки информации.

Полиморфность (*самошифрование*) используется для усложнения процедуры обнаружения вируса. *Полиморфные вирусы* – это трудно выявляемые вирусы, не имеющие постоянного участка кода. В общем случае два образца одного и того же вируса не имеют совпадений. Это достигается шифрованием основного тела вируса и модификациями программы-расшифровщика. Так, например, некоторые макровирусы при создании своих новых копий случайным образом меняют имена своих переменных, вставляют пустые строки или модифицируют свой код иным способом.

Классификация вирусов по способу заражения. Различают так называемые троянские программы, утилиты скрытого администрирования, Intended-вирусы и пр.

Троянские программы получили свое название по аналогии с троянским конем. Назначение этих программ – имитация каких-либо полезных программ, новых версий популярных утилит или дополнений к ним. Очень часто они рассылаются через электронные конференции. При их записи пользователем на свой компьютер троянские программы активизируются и выполняют нежелательные действия.

Разновидностью троянских программ являются *утилиты скрытого администрирования* (backdoor). По своей функциональности и интерфейсу они во многом напоминают системы администрирования компьютеров в сети, разрабатываемые и распространяемые различными фирмами-производителями программных продуктов. При инсталляции эти утилиты самостоятельно устанавливают на компьютере систему скрытого удаленного управления. В результате возникает возможность скрытого управления этим компьютером. Реализуя заложенные алгоритмы, утилиты без ведома пользователя принимают, запускают или отсылают файлы, уничтожают информацию, перезагружают компьютер и пр. Возможно исполь-

зование этих утилит для обнаружения и передачи паролей и иной конфиденциальной информации, запуска вирусов, уничтожения данных.

К *Intended-вирусам* относятся программы, которые не способны размножаться из-за существующих в них ошибок. Например, вирусы при заражении не помещают в начало файла команду передачи управления на код вируса или записывают в нее неверный адрес своего кода. К этому классу также можно отнести вирусы, которые размножаются только один раз. Заразив какой-либо файл, они теряют способность к дальнейшему размножению через него.

Классификация вирусов по деструктивным возможностям. Вирусы разделяют на:

– *неопасные*, влияние которых ограничивается уменьшением свободной памяти на диске, замедлением работы компьютера, графическими и звуковыми эффектами;

– *опасные*, которые потенциально могут привести к нарушениям в структуре файлов и сбоям в работе компьютера;

– *очень опасные*, в алгоритм работы которых специально заложены процедуры уничтожения данных и, согласно одной из неподтвержденных гипотез, возможность обеспечивать быстрый износ движущихся частей механизмов путем ввода в резонанс и разрушения головок чтения/записи некоторых накопителей на жестких дисках.

1.13.2. 1.13.2. Конструкторы вирусов

Конструктор вирусов – это утилита, предназначенная для изготовления новых компьютерных вирусов. Известны конструкторы вирусов для MS-DOS, Windows и макровирусов. Они позволяют генерировать исходные тексты вирусов, объектные модули и/или непосредственно зараженные файлы. Некоторые конструкторы снабжены стандартным оконным интерфейсом, где при помощи системного меню можно выбрать тип вируса, поражаемые объекты, наличие или отсутствие самошифровки, противодействие отладчику, внутренние текстовые строки, сопровождающие работу вируса, эффекты и др.

1.13.3. 1.13.3. Методы борьбы с компьютерными вирусами

Для борьбы с вирусами существуют программы, которые можно классифицировать по основным группам: мониторы, детекторы, доктора, ревизоры и вакцины.

Программы-мониторы. Программы-мониторы (иначе называемые программы-фильтры) располагаются резидентно в оперативной памяти

компьютера, перехватывают и сообщают пользователю об обращениях ОС, которые используются вирусами для размножения и нанесения ущерба. Пользователь имеет возможность разрешить или запретить выполнение этих обращений. К преимуществу таких программ относят возможность обнаружения неизвестных вирусов. Это актуально при наличии самомодифицирующихся вирусов. Использование программ-фильтров позволяет обнаруживать вирусы на ранней стадии заражения компьютера.

Недостатками программ являются невозможность отслеживания вирусов, обращающихся непосредственно к BIOS, а также загрузочных вирусов, активизирующихся до запуска антивируса при загрузке MS-DOS; частая выдача запросов на выполнение операции.

Программы-детекторы. Программы-детекторы проверяют, имеется ли в файлах и на дисках специфическая для данного вируса комбинация байтов. При ее обнаружении выводится соответствующее сообщение. Однако если программа не опознается детекторами как зараженная, то возможно в ней находится новый вирус или модифицированная версия старого, неизвестная программе-детектору.

Программы-доктора. Программы-доктора восстанавливают зараженные программы путем удаления из них тела вируса. Обычно эти программы рассчитаны на конкретные типы вирусов и основаны на сравнении последовательности кодов, содержащихся в теле вируса, с кодами проверяемых программ. Программы-доктора необходимо периодически обновлять с целью получения новых версий, обнаруживающих новые виды вирусов.

Программы-ревизоры. Программы-ревизоры анализируют изменения состояния файлов и системных областей диска. Проверяют состояния загрузочного сектора и таблицы FAT; длину, атрибуты и время создания файлов; контрольную сумму кодов. Пользователю сообщается о выявлении несоответствий.

Программы-вакцины. Программы-вакцины модифицируют программы и диски так, что это не отражается на работе программ, но вирус, от которого производится вакцинация, считает программы или диски уже зараженными.

Существующие антивирусные программы в основном относятся к классу гибридных программ (детекторы-доктора, доктора-ревизоры и др.).

При заражении или при подозрении на заражение компьютера вирусом необходимо:

– оценить ситуацию и не предпринимать действий, приводящих к потере информации. Если вы не обладаете достаточными знаниями и опытом, лучше обратиться к специалистам;

– перезагрузить ОС компьютера. При этом использовать специальную, заранее созданную и защищенную от записи системную дискету. В результате будет предотвращена активизация загрузочных и резидентных вирусов с жесткого диска компьютера;

– запустить имеющиеся антивирусные программы, пока не будут обнаружены и удалены все вирусы. В случае невозможности удалить вирус и при наличии в файле ценной информации произвести архивирование файла и подождать выхода новой версии антивируса. После окончания перезагрузить компьютер.

К антивирусным программам, получившим распространение в России, странах СНГ и за рубежом, относят программы фирм Symantec (Norton Antivirus), Network Associates (Doctor Solomon) и российских фирм – Лаборатории Касперского (AntiViral Toolkit Pro) и ДиалогНаука (ADinf, Dr.Web).

2. ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

Процесс решения задачи на ЭВМ состоит из восьми этапов:

1) постановка задачи;
2) математическое описание задачи;
3) алгоритмизация задачи;
4) программирование;
5) разработка тестовой задачи;
6) перенос программы на машинные носители;
7) отладка программы;
8) получение и анализ результатов.

Рассмотрим более подробно этап алгоритмизации задачи.

2.1. Понятие и основные свойства алгоритма

Алгоритм – это конечная последовательность однозначных предписаний, исполнение которых позволяет с помощью конечного числа шагов получить решение задачи, однозначно определяемое исходными данными.

Свойства алгоритма:

– *дискретность*. Это свойство состоит в том, что алгоритм должен представлять процесс решения задачи как последовательность простых шагов. Т. е. преобразование исходных данных в результат осуществляется во времени дискретно;

– *определенность*. Каждая команда алгоритма должна быть четкой, однозначной и не оставлять места для произвола;

– *результативность*. Алгоритм должен приводить к решению поставленной задачи за конечное число шагов;

– *массовость*. Алгоритм решения задачи разрабатывается не для одной конкретной задачи, а для целого класса однотипных задач, различающихся лишь исходными данными.

Алгоритм может быть предназначен для выполнения его человеком или автоматическим устройством, называемым *формальным исполнителем*.

Объекты, над которыми исполнитель может совершать действия, образуют так называемую *среду исполнителя*.

Существуют следующие способы записи алгоритма:

– *словесно-формульное описание* (на естественном языке с использованием математических формул). Данный способ записи алгоритма состоит из перечня действий (шагов), каждый из которых имеет порядковый номер. Словесное описание алгоритмов применяют при решении несложных задач,

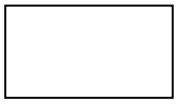
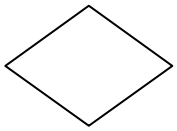
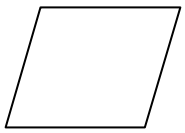
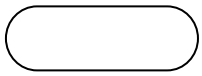
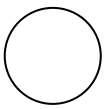
но оно малопригодно для представления сложных алгоритмов из-за отсутствия наглядности;

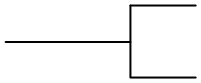
– *графическое описание в виде блок-схемы*. Для обозначения шагов решения в виде схемы алгоритма используются специальные обозначения (символы). Перечень наиболее часто употребляемых символов приведен в табл. 3.1.

– *описание на каком-либо языке программирования* (программа).

Таблица 3.1

Символы для создания блок-схем алгоритмов

Наименование символа	**Обозначение символа**	**Функция символа**
1	**2**	**3**
Процесс		Выполнение операции или группы операций, в результате которых изменяется значение, форма представления или расположения данных
Решение		Выбор направления выполнения алгоритма или программы в зависимости от некоторых переменных условий
Ввод-вывод		Преобразование данных в форму, пригодную для обработки (ввод) или отображения результатов обработки (вывод)
Пуск-останов		Начало, конец, прерывание процесса обработки данных или выполнения программы
Предопределительный процесс		Использование ранее созданных или отдельно описанных алгоритмов и программ
Соединительный		Указание связи между прерванными линиями потока, связывающими символы

Наименование символа	Обозначение символа	Функция символа
1	2	3
Модификация		Организация циклических вычислений
Комментарий		Связь между элементами схемы и пояснением

Программа – это набор машинных команд, который следует выполнить компьютеру для реализации того или иного алгоритма. *Программа* на языке программирования представляет собой совокупность операторов, записанных в соответствии с принятым синтаксисом.

Язык программирования – это набор символов и терминов, который в соответствии с правилами синтаксиса описывает алгоритм решения задачи.

Программирование (programming) – это процесс создания последовательности действий (операций), проводимый в целях достижения требуемого результата. Процесс программирования состоит из стадий: *формулирования*, *разработки программы*, включая *кодирование* и *тестирование*, и, далее, *создания новых версий*.

Существуют сотни реально использующихся языков программирования, каждый для своей области применения. Уровень языка программирования определяется в зависимости от степени детализации алгоритма. Причем, чем меньше детализация, тем выше уровень языка. Различают три уровня языков программирования:

– машинные;

– машинно-ориентированные (ассемблеры);

– машинно-независимые (языки высокого уровня).

Таким образом, машинные языки и машинно-ориентированные языки – это языки *низкого уровня*, которые требуют указания мелких деталей процесса обработки данных.

Проблемно-ориентированные языки относятся к группе искусственных языков. Проблемно-ориентированные языки предоставляют особые удобства для наглядного описания процедурных шагов процесса обработки данных и передачи данных при решении самого широкого класса задач. Так, все большее развитие получают языки параллельной

обработки, а в технике важную роль выполняет язык описания технических средств.

Средством разработки и развития программного обеспечения является *инструментальное программное обеспечение* (*software tools*). Инструментальное программное обеспечение, называемое также инструментарием, характеризуется набором программ по созданию и модернизации новых программ. Среди них: текстовой редактор, графический редактор, транслятор, загрузчик, а также средства отладки программ.

Инструментальное программное обеспечение обеспечивает выполнение последовательных этапов, включающих выработку требований к создаваемому программному обеспечению, общее проектирование, детальная разработка, создание отдельных модулей, тестирование полученных модулей, объединение модулей в единое целое, выпуск проекта, эксплуатацию и сопровождение созданных программ. В процессе руководства процессом разработки инструментарий опирается на специальную целевую базу данных. Инструментальное программное обеспечение не уничтожается в течение жизненного цикла разрабатываемой программы.

Программисты используют в своей работе ряд характерных методов и моделей. Например, *водопадная модель* процесса разработки программы подразумевает линейную последовательность выполнения работ: *анализ-проектирование-создание*. То есть, каждый последующий этап должен быть завершен до начала следующего. Такая модель имеет ряд недостатков: не учитываются эволюционные изменения, игнорируется аспект организации структуры данных, нет базы для многократного использования.

После написания программы, она подвергается *тестированию*. Цель тестирования состоит в определении ее работоспособности и степени готовности. После тестирования проводят анализ поведения программы. Так как процесс создания большой программы чрезвычайно трудоемок и ряд ошибок выявляются через некоторое время, доказательство правильности работы программы – сложный процесс.

2.2. Языки программирования высокого уровня

Языки высокого уровня во многом имитируют естественные языки, используют многие разговорные слова, общепринятые математические символы. Различают языки следующих групп: алгоритмические, предназначенные для однозначного описания алгоритмов (такие, как *Basic*, *Pascal*, *C*), логические – ориентированные не на разработку алгоритма ре-

шения задачи, а на систематическое и формализованное описание задачи (например, *Prolog*, *Lisp*), объектно-ориентированные, основанные на понятии объекта и действиях над ним (к примеру, *Object Pascal*, *C++*, *Java*).

В мире осуществляется стандартизация языков программирования высокого уровня. Ею занимается специализированная организация «Конференция по языкам информационных систем» (Conference On DAta SYstem Language – CODASYL). Эта организация основана министерством обороны США, она создана в 1959 г., в нее входят специалисты индустрии информатики, представляющие промышленные предприятия, фирмы, занимающиеся разработкой программного обеспечения и средств обработки данных. CODASYL разрабатывает стандарты управления данными, языки программирования. Для эффективного использования высокоуровневых языков создаются специальные интегральные схемы.

Языки программирования подробно изучаются при подготовке профессиональных программистов.

2.3. Основные понятия объектно-ориентированного программирования

В общем случае создаваемое приложение (программа или пакет программ) реализуется в виде набора взаимосвязанных модулей. Каждый модуль имеет заданные входные и выходные параметры и реализует определенную функцию. При создании приложения модули собираются в единое целое. В настоящее время концепция модульных систем не позволяет обеспечить развитие приложений в полном объеме. Одним из путей ее совершенствования является объектно-ориентированный подход. Основными понятиями этого подхода являются *объект*, *свойства*, *методы*.

Объект – это совокупность свойств определенных сущностей (в данном случае *сущность* – это какая-то часть программы или данных, которые обрабатываются программой) и методов их обработки. Объект содержит операторы и обрабатываемые данные и взаимодействует с другими объектам посредством обмена сообщениями. Примерами объектов являются прикладные программы, документы, процессы, события.

Характеристикой объекта является *свойство*. Свойства объекта выделяют его из множества подобных объектов, задают качественную определенность, обусловливают независимость создания и обработки от других объектов.

Методы – это действия, выполняемые над объектом или его свойствами. Метод рассматривается как связанная с объектом программа, посредством которой осуществляется преобразование свойств объекта или его поведения. Крупные объекты создаются путем объединения мелких и наследуют их свойства. Объект имеет скрытую внутреннюю структуру и обладает определенным интерфейсом. В результате, заменяя внутреннюю реализацию объекта другой, оказывается возможным сохранять построенные на его основе крупные объекты. Этот подход принят многими фирмами, например Microsoft, Apple, IBM, Novell, Sun Microsystems.

2.4. Введение в программирование на языке Паскаль (общая структура программ, основные элементы языка).

Program {заголовок программы}
Uses {подключаемые библиотеки}
Label {объявление меток}
Const {объявление констант}
Type (объявление типов}
Var {объявление переменных}
Procedure (**Function**){описание процедуры (функции)}

Раздел основного блока программы:

Begin{начало блока}
операторы
End.{конец блока}

Кроме того, в программе могут использоваться **комментарии** — любой текст, ограниченный (* *) или { }. Комментарий может быть помещен в любом месте программы.

В строке **Uses** производится подключение используемых в данной программе библиотек (стандартных модулей). Паскаль содержит ряд **модулей в своей библиотеке модулей**, в том числе:

System;
Dos и **WinDos**;
Crt;
Printer;
Graph.

В разделе описания меток **Label** содержатся сведения об используемых в программе **метках**. С помощью меток обеспечивается в программе

безусловный переход к другим операторам. Переход по меткам выполняется оператором

Goto <имя метки>;

Метки отделяются от помечаемого оператора знаком двоеточие - “**:**”. Метки могут быть целочисленными (от 0 до 9999) или идентификаторами. Каждая описанная метка обязательно должна появиться в программе.

В разделе описания типов **type** программист может задавать свои типы, сформированные на основе определенных правил с использованием стандартных типов.

Type

Имя типа 1 = структура типа на основе базового типа 1;

… ;

В разделе **Var** описываются переменные**,** используемые в программе:

Var

список 1: имя типа 1;

список 2: имя типа 2;

… ;

В списке переменных их имена отделяются запятой.

В разделе текстов процедур и функций в соответствии с определенными правилами производится описание процедур и функций.

Основной блок программы состоит из ряда операторов и является выполняемой частью программы. Он начинается со слова **Begin** и заканчивается словом **End.**, после которого должна стоять **точка**.

Операторы языка **Паскаль** не привязаны к определенной позиции строки. В одной строке можно указывать несколько операторов. Исполняемые операторы отделяются друг от друга знаком “**;**”. Допускается перенос с одной строки на другую частей операторов (но без разделения ключевых слов).

Алфавит языка Паскаль включает**:**

- латинские буквы;
- цифры;
- специальные символы **(+ - * / = ^ < > ( ) [ ] { } . , : ; ‘ # $** и др.).

Кроме того, в символьных константах и комментариях могут использоваться и другие знаки (например, буквы русского алфавита).

При формировании и описании своих базовых элементов язык Паскаль использует **зарезервированные слова**, которые не могут быть переопределены пользователем**:**

and end not shr

array	file	object	then
asm	for	of	to
abs	function	or	type
begin	goto	procedure	unit
case	if	program	until
const	in	record	uses
div	interrupt	repeat	var
do	label	string	with
downto	mod	set	while
else	nil	shl	xor

Простые типы данных

Два типа T1 и T2 являются эквивалентными или совместимы, если выполняется одно из двух условий:

- TI и T2 представляют собой одно и то же имя типа;
- тип T2 описан с использованием типа T1 с помощью равенства или последовательности равенств. Например:

```
type
  T1 =  integer;
  T2 =  T1;
```

Каждому из перечисленных ранее типов соответствует свой набор операций. Так, для целочисленного и вещественного типов используются операции арифметических действий: ” +” - сложение, “ – “ - вычитание**, “** * “ - умножение, и “ / “ — деление, которые выполняются в соответствии с их математическим приоритетом. Приоритет действий в выражениях может быть изменен путем введения круглых скобок. Кроме этого, для целых значений применима операция целочисленного деления **div,** которая возвращает частное от деления (например, 7 **div** 3 = 2) и операция **mod,** которая возвращает остаток от деления двух целых чисел (например, 7 **mod** 3 = 1). Кроме того, Паскаль предусматривает возможность вычисления для каждого типа данных значений стандартных функций**.**

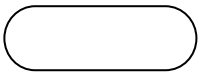
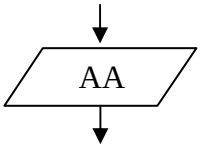
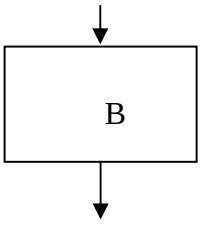
cos(x)	косинус числа
sin(x)	синус числа
abs(x)	абсолютное значения числа
sqr(x)	квадрат числа
sqrt(x)	квадратный корень числа
arctan(x)	арктангенс числа
exp(x)	экспонента числа
ln(x)	натуральный логарифм числа
Round(x)	Округление числа до целого
trunc(x)	Вычисление дробной части
int(x)	Вычисление целой части
odd(x)	нечетность числа

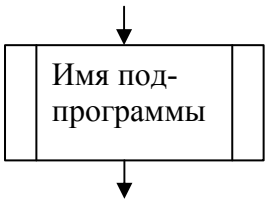
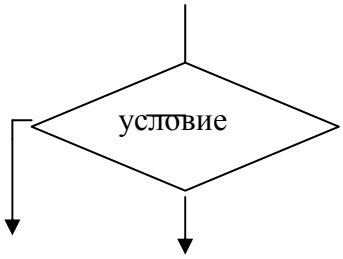
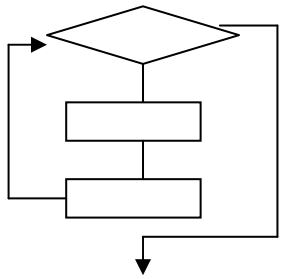
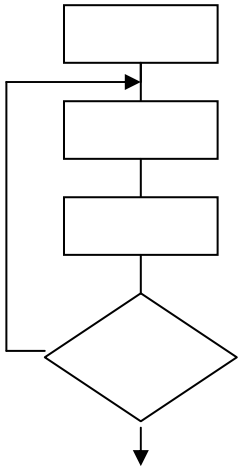
Примечание. Во всех тригонометрических функциях число **X** указывается в радианах.

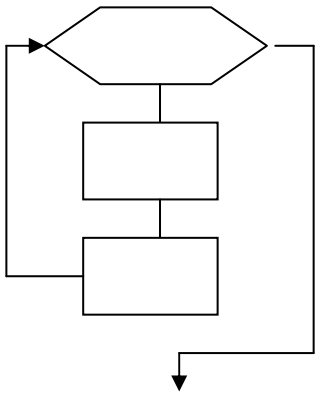
Данные *логического типа* могут принимать два значения: True - “истина” и False – “ложь”. Для логического типа выполняются операции сравнения (<, >, =, <=, >=, <>) и логические операции: **and** — логическое “и”; **or** — логическое “или”; **xor** — логическое исключающее “или”**; not —** логическое. “не”.

Символьный тип предназначен для хранения одного символа (буквы, знака, цифры или кода). Имя типа *Char*. Формат представления символьного типа 1 байт. Диапазон порядковых номеров типа 0 .. 255. **CHR(x)** - эта функция преобразует выражение x типа byte в символ и возвращает его своим значением, например, CHR(65) возвращает символ A.

Операторы языка и их аналогии в алгоритмических структурах.

N	Графический символ	Назначение	Оператор языка
1		Блок начала алгоритма Блок конца алгоритма	BEGIN END.
2		Ввод данных и вывод данных	READ(AA) WRITE(AA)
3		- вычислительный процесс с сохранением результата в переменной B или - присвоение значений переменным	B:=AA+SQR(AA) G:=0; U:=0

N	Графический символ	Назначение	Оператор языка
4		Предопределенный процесс: работа подпрограммы (функции или процедуры)	Вызов функции или вызов процедуры
5	 	Блок принятия решения. Точка разветвления алгоритма может использоваться и в других случаях: а) при разработке циклического алгоритма с предусловием;	IF .. THEN...ELSE WHILE ... DO ...
6		б) при разработке циклического алгоритма с послеусловием	REPEAT... . . UNTIL ...

N	Графический символ	Назначение	Оператор языка
7		Блок модификации для циклических структур со счетчиком	FOR ..TO....DO

2.5. Программирование линейных алгоритмов

Алгоритм линейной структуры (следование) – это алгоритм, в котором все действия выполняются последовательно друг за другом. Такой порядок выполнения действий называется естественным.

В качестве примера составим программу для задачи, в которой требуется вычислить сумму двух заданных чисел, которые необходимо ввести с клавиатуры. Результат необходимо вывести на экран.

```
Program   sum;
var
a, b, s:   real;
begin
write('Введите значения чисел a и  b     ');
readln(a, b);
s := a + b;
writeln ('Сумма ='; s:7:2);
end.
```

Таким образом, на экран выведется **Сумма = 38.70**. Выведенное вещественное число имеет две цифры в дробной части и поле вывода в семь позиций экрана.

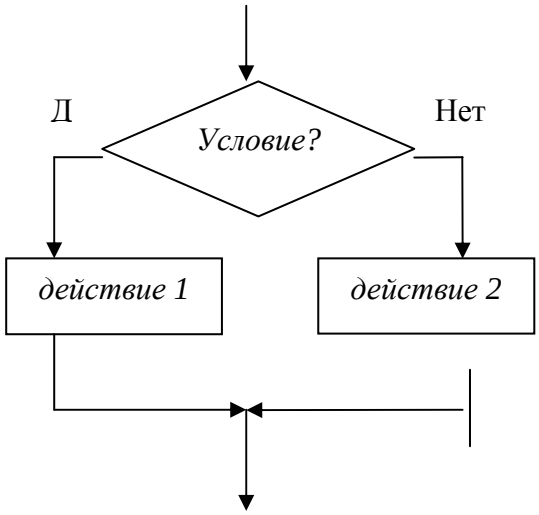
Пример. Пусть требуется преобразовать величину, выраженную в минутах, в соответствующее ей значение, выраженное в часах и минутах.

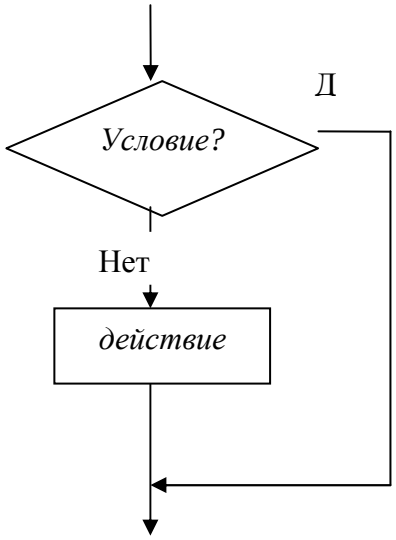
```
Program Preobr_1;
var
min : integer;  { интервал в минутах}
```

```
  h : integer;  { количество часов}
  m : integer;  {количество минут}
begin
write( ‘Введите величину временного интервала в минутах : ’);
readln(min);
h := min div 60; m := min mod 60;
writeln(min, ‘ мин. – это ’,h, ‘  час.’, m, ‘  мин.’);
readln;
end.
```

В приведенной программе для определения количества часов использовалась операция целочисленного деления div, а для определения минут – операция вычисления остатка от целочисленного деления mod.

2.6. Программирование разветвляющихся алгоритмов





На языке Паскаль для записи такого алгоритма используется оператор условных переходов сокращенной формы.

Оператор имеет вид:

if <условие> **then** <оператор 1> **e1se** <оператор 2>**;**

Простые условия записываются в виде равенств или неравенств. Сложные условия составляются из простых условий с помощью логических операций. Например, сложное условие может выглядеть так:

```
If (a>d) and (c>d) then min:=d else …;
If (a>d) or (c>d) then min:=d else …;
```

If (a>2) and (a<5) or (a>10) and (a<15) then s:=s+1else s:=0;

Приведем пример программы с использованием оператора **if**. Пусть необходимо вывести на экран большее из двух данных чисел. Программа для решения этой задачи имеет вид:

```
program pr1;
var   x,y:integer;
begin
writeln('Введите два числа');
readln(x,y);
if x>y then writeln (x)
  else writeln(y);
readln;
end.
```

2.7. Программирование алгоритмов множественного выбора

Такие алгоритмы можно было бы программировать с помощью вложенных операторов условного перехода If Then Else, но в языке Паскаль есть более компактная конструкция – это оператор множественного выбора **Case Of**. Оператор **case** имеет вид:

case <порядковая переменная> **of**
<константа 1> : <оператор 1>;
…
<константа n>: <оператор n>;
else <оператор>
end;

Пример 3. Приведем текст полной программы, переводящей школьные отметки в словесные оценки. Переменная – селектор должна принимать значения целых чисел: 1,2,3,4,5.

```
PROGRAM  Primer_3;
  var BALL : Integer;
BEGIN
  Write ('Введите величину отметки: '); Read (BALL);
  WriteLn;
  Case  BALL  of  { Перевод  отметки в оценку }
    1: WriteLn ('Очень плохо...');
    2: WriteLn ('Плохо...');
```

```
    3: WriteLn ('Удовлетворительно...');
    4: WriteLn ('Хорошо!');
    5: WriteLn ('Отлично!')
END.
```

2.8. Программирование циклических алгоритмов

2.8.1. Оператор for

Данный оператор применяют тогда, когда известно число повторений одного и того же действия. Начальное и конечное значения параметра цикла могут быть представлены константами, переменными или арифметическими выражениями Оператор имеет две формы:

for <параметр> := **a to b do** <тело цикла>;
for <параметр> := **a downto b do** <тело цикла>;

где **a** и **b** — величины целого типа, причем **a** — начальное значение, а **b** — конечное значение переменной-параметра.
Рассмотрим, как выполняется оператор

for <параметр> := **a to b do** <тело цикла>;

Сначала вычисляются значения выражений **a** и **b**. Если **a** окажется меньшим или равным **b**, то <параметр> последовательно принимает значения, равные **a, a+1,…,b—1, b**, и для каждого из этих значений исполняется <тело цикла>. Если же **a > b**, то <тело цикла> не будет выполнено ни разу, и управление будет передано следующему оператору программы.
Оператор цикла с параметром

for <параметр> := **a downto b do** <тело цикла>;

выполняется аналогичным образом, но <параметр> принимает значения, равные **a, a-1, ,..., b+1, b**, т.е сшагом -1.
Если <тело цикла> состоит из нескольких операторов, то операторы тела цикла заключаются в операторные скобки **begin — end**. Например:

```
for  x:=1  to  10  do
begin
      writeln(x);
      for i:=10 to  100 do y:=y+5
end;
```

2.8.2. Оператор while

Данный оператор используется для программирования циклических процессов, в которых число повторений неизвестно, но задается некоторое условие его окончания. Это оператор цикла с предусловием.
Оператор имеет вид:

while <условие> **do** <тело цикла>;

Работа оператора начинается с проверки условия, записанного после слова **while**. Если это условие соблюдается, то выполняется <тело цикла>, а затем вновь проверяется условие и т. д. Как только на очередном шаге окажется, что условие не соблюдается, то выполнение <тела цикла> прекратится. Если <тело цикла> состоит из нескольких операторов, то они объединяются операторными скобками **begin …end**. В <теле цикла> обязательно должен быть оператор, влияющий на соблюдение условия, в противном случае произойдет зацикливание.

2.8.3. Оператор repeat

Для программной реализации циклических процессов с неизвестным числом повторений существует еще; один оператор, оператор цикла с послеусловием, он имеет вид:

repeat
<тело цикла>
until <условие>;

Данный оператор аналогичен предыдущему оператору цикла, но отличается от него тем, что проверка условия производится после очередного выполнения <тела цикла>. Это обеспечивает его выполнение хотя бы один раз. Сначала выполняется последовательность операторов, входящих в <тело цикла>, после чего проверяется выполнение условия, записанного за служебным словом **until**. Если условие не соблюдается, цикл завершается. В противном случае <тело цикла> повторяется еще раз, после чего снова проверяется соблюдение условия. Естественно <тело цикла> должно содержать оператор, влияющий на условие окончания (продолжения), иначе цикл будет бесконечным.

В качестве примера рассмотрим задачу планирования закупки товаров в магазине на определенную сумму, не превышающую заданной величины. Обозначим через c и k соответственно цену и количество товара, через p — заданную предельную сумму, через s -— общую стоимость

покупки. Начальное значение обшей стоимости покупки (s) равно нулю. Значение предельной суммы *(p)* должно вводиться с клавиатуры. Необходимо повторять запрос о цене и количестве выбранного товара, вычислять его стоимость, суммировать ее с общей стоимостью и выводить результат на экран до тех пор, пока s не превысит предельную сумму *p*. В таком случае на экран нужно вывести сообщение о превышении предельного значения. Решение этой задачи можно записать в виде следующей программы:

```
program pr2;
var c, k, p, s; integer;
begin
writeln('Предельная сумма -'); readln(p);
s:=0;
repeat
writeln ('Введите цену товара и его количество');
readln(c,k);
s:=s+c*k;
writeln ('Стоимость покупки равна', s);
until s>p;
writeln('Общая стоимость покупки превысила предельную сумму');
end.
```

2.9. Табуляция функций и рекуррентные вычисления

Циклические операторы хорошо реализуют алгоритм табуляции функций. Пусть некоторая функция y = f(x) должна быть протабулирована на отрезке (a,b) с шагом h = (b - a) / n, где n - количество интервалов табуляции. Эта задача может быть решена с помощью оператора цикла с известным числом повторений (операторf **for**):

```
program Tabula;
const eps = 1e-5;
var a,b,x,y,h: Real;
   n,i:    Integer;
begin
      Write('Введите a,b и n: ');
      ReadLn(a,b,n);
      h:=(b-a)/n;
      i:=0; x:=a;
      WriteLn('x':8,'y':8);
      WriteLn('----------------');
      For i:=1 to n do
```

```
begin
   if Abs(x)<eps then
      y:=1
   else
      y:=Sin(x)/x;
   WriteLn(x:8:3,y:8:3);
    x:=x+h;
 end;
end.
```

Здесь условие Abs(x) < 1e-5 определяет близость переменной x к нулю и введено для исключения возможности появления ошибки "Деление на нуль".

Рекуррентные вычисления основаны на многократном повторном использовании одной и той же формулы. Такие вычисления называют также итерационными. Для их реализации используют циклические программы. Известна итерационная формула для вычисления квадратного корня из x: y[0] = x, y[i+1] = (y[i] + x / y[i]) / 2 для i > 0. Вычисления можно прекратить, когда будет достигнута необходимая точность: Abs(y[i+1] - y[i]) <= Eps. Следующая программа решает поставленную задачу.

```
program MySqrt;
const Eps =1e-5;
var x:  Real;
   y:  Real; {Текущее значение}
   ys: Real; {Следующее значение}
begin
Write('Entry x: ');
ReadLn(x);
ys:=x;
repeat
   y:=ys;
   WriteLn('y = ', y);
   ys:=(y + x/y)/2;
until Abs(ys-y)<Eps;
WriteLn('MySqrt(', x,') = ', ys);
WriteLn('  Sqrt(', x,') = ', Sqrt(x));
end.
```

2.10. Структурированные типы данных. Множества

Множество - совокупность однотипных элементов базового типа соответствующая понятию множества в математике с набором свойственных операций.

Массив - совокупность однородных элементов базового типа, обращение к которым выполняется с помощью индексов ординального типа.

Запись - совокупность неоднородных элементов базовых типов, обращение к которым выполняется с помощью имён полей.

Файл - совокупность однородных элементов базового типа, доступ к которым осуществляется последовательно.

Работа с множествами. Переменные множества должны быть описаны следующим предложением:

var <имя>: **set of** <базовый тип>;

Например:

```
var s: set of 2..n;
d: set of 'a'.. 'z';
```

Для множеств определены три операции: “+”, “*”, “-”, которые обозначают соответственно объединение, пересечение и разность множеств. С их помощью можно строить различные выражения множественного типа. **Объединением двух множеств** называется множество элементов, принадлежащих обоим множествам. **Пересечением двух множеств** называется множество тех элементов, которые принадлежат одновременно двум множествам. **Разностью двух множеств** называется множество, содержащее те элементы первого множества, которые не являются элементами второго.

В языке **Паскаль** определены следующие отношении для множеств:

- принадлежность элемента x множеству A (x in A);
- равенство двух множеств A и B (A = B);
- включение множества A в множество B (A < B);
- нестрогое включение множества A в множество B (A <= B);
- отношения >, >=, < >.

Сведем все операции над множествами в следующую таблицу.

a **in** M	Элемент a принадлежит множеству M
M1 <= M2	M1 содержится в M2
M1 >=M2	M1 включает M2
M1 = M2	M1 и M2 эквивалентны
M1 <> M2	M1 и M2 неэквивалентны
M1 + M2	Объединение множеств M1 и M2
M1 * M2	Пересечение множеств M1 и M2

M1 - M2 Разность множеств M1 и M2 (множество элементов из M1, которых нет в M2)

В Паскале на базовый тип наложено ограничение (Ord(Min) >= 0) and (Ord(Max) < 256), где Min и Max минимальное и максимальное значения типа.

Пример 1. Программа выводит элементы множества.

```
program WrSet;
const MIN = 1; MAX = 12;
var i: Integer;
   S: set of MIN..MAX;
begin
S := [2, 5..8];
for i := MIN to MAX do
   if i in S then Write(i:3);
end.
```

Строковый тип данных

Для обработки строковой информации в Паскаль введен строковый тип данных. Строкой в Паскале называется последовательность из определенного количества символов. Количество символов последовательности называется длиной строки. Синтаксис:

```
var  s :  string[n];
var  s :  string;
```

n - максимально возможная длина строки - целое число в диапазоне 1..255. Если этот параметр опущен, то по умолчанию он принимается равным 255. Строковые константы записываются как последовательности символов, ограниченные апострофами. Допускается формирование строк с использованием записи символов по десятичному коду (в виде комбинации # и кода символа) и управляющих символов (комбинации ^ и некоторых заглавных латинских букв).

Пример:

```
‘текстовая строка'
#54#32#61
'abcde', ^A^M
```

Пустой символ обозначается двумя подряд стоящими апострофами. Если апостроф входит в строку как литера, то при записи он удваивается.

Переменные, описанные как строковые с разными максимальными длинами, можно присваивать друг другу, хотя при попытке присвоить короткой переменной длинную лишние символы будут отброшены.

Выражения типа char можно присваивать любым строковым переменным.

2.11. Структурированные типы данных - массивы. Одномерные массивы

Массив представляет собой совокупность связанных данных, состоящую из фиксированного числа элементов одного типа, который называется ***базовым***. Для определения массива достаточно указать его базовый тип, а также число элементов в массиве и метод их нумерации. К элементу одномерного массива обращаются через индексную переменную, которая записывается как идентификатор массива (имя) с одним индексом в квадратных скобках. Одномерные массивы характеризуются следующим:

- каждый элемент массива может быть явно обозначен и к нему имеется прямой доступ, т.к. индекс указывает на номер элемента в массиве;
- число элементов массива определяется при его описании и в дальнейшем не меняется.

Тип индекса может быть только порядковым (кроме longint). Чаще всего используется интервальный тип (диапазон).

Описание типа массива задается следующим образом:

type
имя типа = **array[***список индексов***] of** *тип элементов*;

Здесь *имя типа* –идентификатор; *список индексов* – список диапазонов значений индексов или их типов, разделенных запятыми; *тип элементов*– любой тип данных.

Определить переменную как массив можно и непосредственно при ее описании, без предварительного описания типа массива, например:

```
var a,b,c : array[1..10] of integer;
```

Таким образом, при описании массива задается имя массива, его размер и тип данных.

Массив можно описать через типизированную константу с исходными значениями элементов.

```
Const a : array[1..5] of integer = (13, 44, 45, 64, 20);
```

Если массивы a и b описаны как:

```
var
a = array[1..5] of integer;
b = array[1..5] of integer;
```

то переменные a и b считаются разных типов. Для обеспечения совместимости применяйте описание переменных через предварительное описание типа.

```
type
     m_type  =  array[1..5]  of byte;
var   mas1,mas2 : m_type;
```

Если типы массивов идентичны, то в программе один массив может быть присвоен другому, например, можно записать так: mas2:=mas1. В этом случае значения всех переменных одного массива mas1 будут присвоены соответствующим элементам второго массива mas2.

Вместе с тем, над массивами не определены операции отношения. Сравнивать два массива можно только поэлементно.

Так как *mun*, идущий за ключевым словом **of** в описании массива, - любой тип Паскаль, то он может быть и другим массивом. Например:

```
type
mas  =  array[1..5]  of array[1..10] of   integer
```

Такую запись можно заменить более компактной:

```
type
mas  =  array[1..5,  1..10] of   integer;
```

Таким образом, возникает понятие многомерного массива. Глубина вложенности массивов произвольная, поэтому количество элементов в списке индексных типов (размерность массива) не ограничена, однако не может быть более 65520 байт.

Программы обработки массивов всегда носят циклический характер. Так, например, ввод и вывод данных в массивы осуществляется с использованием цикла, так как процесс ввода (вывода) приходится повторять столько раз, сколько элементов содержит массив. Рассмотрим пример программы с вводом и выводом одномерного массива.

```
Program pr1;
const       n  =  5;
type       mas  =  array[1...n]  of  integer;
var
        a :  mas;
        i :  byte;
begin
     writeln('введите элементы массива' );
     for  i:=1  to  n  do  readln(a[i]);
     writeln('вывод элементов массива: ' );
```

```
    for i:=1 to n do write(a[i] : 5);
end.
```

Пример. Программа для определения суммы элементов массива с использованием цикла с предусловием.

```
program SumArr1;
type TArr = array[1..5] of Real;
var A: TArr;
   i: Integer;  S: Real;
begin
Write('A[1..5]: '); i := 1;
while i <=5 do begin Read(A[i]); i := i + 1 end;
S := 0; i := 1;
while i <= 5 do begin S := S + A[i]; i := i + 1 end;
WriteLn('S = ', S);
end.
```

Особенно удобен при работе с массивами оператор цикла с известным числом повторений **for**. Напомним структуру оператора.

for i := e1 to e2 do S ,

где i - параметр цикла, простая переменная порядкового типа; e1, e2 - выражения, определяющие соответственно начальное и конечное значения параметра i. S - оператор, выполняемый в цикле. Типы переменных i, e1 и e2 должны совпадать.

Для задания размеров массивов можно применять именованные константы. Ниже приведен пример применения именованной константы и оператора цикла с известным числом повторений для определения суммы элементов массива

```
program SumArr2;
const N = 5;
type TArr = array[1..N] of Real;
var A: TArr;
   i: Integer;
   S: Real;
begin
Write('A[1..5]: ');
for i := 1 to N do Read(A[i]);
S := 0;
for i := 1 to N do S := S + A[i];
WriteLn('S = ', S);
end.
```

Поиск минимального (максимального) элемента массива

Алгоритм поиска: делается предположение, что первый элемент массива является минимальным (максимальным), затем остальные элементы массива сравниваются с этим элементом. Если обнаруживается, что проверяемый элемент меньше (больше) принятого за минимальный (максимальный), то этот элемент принимается за минимальный (максимальный) и продолжается проверка оставшихся элементов.

```
M := A[1];
for i := 2 to N do
  if A[i] < M then M := A[i];
```

Вычисление произведения элементов массива

Фрагмент программы достаточно прост:

```
P := 1;   for  i := 1 to N do  P := P * A[i];
```

Выборка элементов массива

Составить программу подсчета количества положительных элементов массива, состоящего из 10 элементов.

Текст программы:

```
program mas3;
var k, i : integer;
a: array[1..10] of integer;
begin
    for i:=1 to 10 do
    begin
        writeln(' Введите элемент массива');
        readln(a[i]);
    end;
    writeln(' Элементы исходного массива');
    for i;=1 to 10 do
        write (a[i],' ');
    writeln;
    k:=0;
    for i:=1 to 10 do
        if a[i]>0 then k:=k+1;
    writeln(' Количество равно',k);
end.
```

Найти значение и номер наибольшего элемента в массиве из 10 целочисленных элементов.
Текст программы:

```
program mas4;
var n, m, i : integer;
a: array[1..10] of integer;
begin
   for i:=1 to 10 do
   begin
         writeln('Введите элемент массива');
         readln(a[i]);
   end;
writeln('Элеметы исходного массива');
for i:=1 to 10 do
   write(a[i]), ' ');
   writeln;
m:=a[1]; n:= 1;
for i:=2 to 10 do
   if a[i]>m then
         begin
               m:= a[i];
               n:= i;
         end;
writeln('Максимальный элемент равен', m, 'и имеет порядковый номер', n);
end.
```

Сортировка массива. Под сортировкой массива подразумевается процесс перестановки его элементов с целью упорядочивания их в соответствии с каким-либо критерием. Сортировка проводится на основании сравнения значений элементов массива друг с другом. Можно сравнивать значения целых, вещественных, символьных, строчных и производных от них типов.

Сортировка методом прямого выбора. Алгоритм сортировки массива по возрастанию методом прямого выбора может быть представлен так.

1. Просматривая массив от первого элемента, найти минимальный и поместить его на место первого элемента, а первый на место минимального.
2. Просматривая массив от второго элемента, найти минимальный и поместить его на место второго элемента, а второй на место минимального.
3. И так далее до последнего элемента.

Для демонстрации рассмотрим программу сортировки массива целых чисел по возрастанию (для контроля выполнения алгоритма программа выводит массив после каждого обмена элементами).

```
Program sort1;
    Const  size=5;
    Var
    a: array[1..size] of integer;
    i: integer; {номер элемента, от которого ведется поиск минимального элемента}
    min: integer; {номер минимального элемента в части массива от i до верхней границы массива}
    j: integer; {номер элемента сравниваемого с минимальным}
    buf: integer; {буфер, используемый при обмене элементов массива}
    k: integer;
begin
    writeln('Сортировка массива');
    write('Введите через пробел', size:3,'целых в одной строке и нажмите клавишу ввод');
    for k:=1 to size do read(a[k]);
    writeln('Сортировка');
    for i:=1 to size-1 do
        begin
        min:=i;
        for j:=i+1 to size do
        begin
        if a[j]<a[min] then min:=j;
        end;
        buf:=a[i];  a[i]:=a[min];  a[min]:=buf;
        for k:=1 to size do write(a[k],' ');
        writeln;
        end;   end.
```

Сортировка методом прямого обмена. В основе алгоритма лежит обмен соседних элементов массива. Каждый элемент массива, начиная с первого, сравнивается со следующим, и если он больше следующего, то элементы меняются местами. Таким образом, элементы с меньшим значением подвигаются к началу массива (всплывают), а элементы с большим значением – к концу массива (тонут), поэтому этот метод иногда называют «пузырьковым». Этот процесс повторяется на единицу меньше раз, чем элементов в массиве. Ниже представлена программа сортировки массива целых чисел

по возрастанию. Для демонстрации процесса сортировки программа выводит массив после каждого цикла обменов.

```
Program sort2;
    const
    size=5;
    Var
    a: array[1..size] of integer;
    i: integer; {счетчик циклов}
    buf: integer;
    k: integer; {текущий индекс элемента массива}
begin
    writeln('Сортировка массива методом пузырька');
    write('Введите через пробел', size:3,'целых в одной строке и нажмите клавишу ввод');
    for k:=1 to size do read(a[k]);
    writeln('Сортировка');
    for i:=1 to size-1 do
        begin
        for k:=1 to size-1 do
        begin
            if a[k]>a[k+1] then
                    begin
                    buf:=a[k];
                    a[k]:=a[k+1];
                    a[k+1]:=buf;
                    end;
         end;
        for k:=1 to size do write(a[k],' ');
        writeln;
    end;
end.
```

Поиск в массиве

Поиск осуществляется последовательным сравнением элементов массива с образцом до тех пор, пока не будет найден элемент, равный образцу, или не будут проверены все элементы. Алгоритм простого перебора применяется, если элементы массива не упорядочены.

Рассмотрим программу поиска в массиве целых чисел. Перебор элементов массива осуществляет инструкция REPEAT, в теле которой инструкция IF сравнивает текущий элемент массива с образцом и присваивает

переменной naiden значение true, если текущий элемент равен образцу. Цикл завершается, если в массиве обнаружен элемент равный образцу, или если проверены все элементы массива. По завершении цикла по значению переменной found можно определить, успешен поиск или нет.

```
program poisk;
var
massiv: array[1..10] of integer; {массив целых}
obrazec: integer; {образец для поиска}
naiden: boolean; {признак совпадения с образцом}
i: integer;
begin
    writeln('Введите через пробел 10 целых элементов массива и нажмите клавишу ввода');
    for i:=1 to 10 do
    read(massiv[i]);
    write('Введите образец для поиска’);
    readln(obrazec);
    naiden:=false;
    i:=1; {проверяем с первого элемента массива}
    repeat
        if massiv[i]=obrazec then naiden:=true else i:=i+1;
    until (i>10)or(naiden);
    if naiden then writeln('Совпадение с элементом',i:3,'поиск завершен')
    else writeln('Совпадений с образцом нет');
end.
```

2.12. Подпрограммы, пользовательские процедуры и функции

Структура описания процедур и функций до некоторой степени похожа на структуру Паскаль - программы: у них также имеются заголовок, раздел описаний и исполняемая часть. Раздел описаний содержит те же подразделы, что и раздел описаний программы: описания констант, типов, меток, процедур, функций, переменных. Исполняемая часть содержит собственно операторы процедур. Формат описания процедуры имеет вид:

procedure имя процедуры **(формальные параметры) ;**
раздел описаний процедуры
begin
исполняемая часть процедуры
end;

Формат описания функции:

function имя *функции* **(формальные *параметры)*: тип** результата;

раздел описаний функции
begin

исполняемая часть функции

имя *функции:=результат;*

end;

Формальные параметры в заголовке процедур и функций записываются в виде:

var имя параметра: имя типа;

и отделяются друг от друга точкой с запятой. Ключевое слово **var** может отсутствовать для входных параметров, а *для выходных должен быть обязательно*. Если параметры однотипны, то их имена можно перечислять через запятую, указывая общее для них имя типа. При описании параметров можно использовать только стандартные имена типов, либо имена типов, определенные с помощью команды Type вызывающей программы. Список формальных параметров может отсутствовать.

Вызов процедуры производится оператором, имеющим следующий формат:

имя *процедуры* **(список** *фактических параметров***);**

Список фактических параметров - это их перечисление через запятую. При вызове фактические параметры как бы подставляются вместо формальных, стоящих на тех же местах в заголовке. Таким образом, происходит передача входных параметров, затем выполняются операторы исполняемой части процедуры, после чего происходит возврат в вызывающий блок. Передача выходных параметров происходит непосредственно во время работы исполняемой части.

Вызов функции в Паскаль может производиться аналогичным способом, кроме того, имеется возможность осуществить вызов внутри какого-либо выражения. В частности, имя функции может стоять в правой части оператора присваивании, в разделе условий оператора if и т.д.

Для передачи в вызывающий блок выходного значения функции в исполняемой части функции перед возвратом в вызывающий блок необходимо поместить следующую команду:

имя *функции* : = результат;

При вызове процедур и функций необходимо соблюдать следующие правила:

- количество фактических параметров должно совпадать с количеством формальных;
- соответствующие фактические и формальные параметры должны совпадать по порядку следования и по типу.

Процедуры и функции должны обладать определенной независимостью в смысле использования переменных, типов и констант. В связи с этим возникает необходимость разделять данные на *глобальные* и *локальные* по их доступности из других подпрограмм.

Глобальные – это те константы, типы и переменные, которые объявлены в программе вне описания процедуры и функции и до их описания. Они доступны вызывающей программе и вызываемой подпрограмме.

Локальные константы, типы и переменные существуют только внутри подпрограммы, в которой они описаны через разделы объявлений или через список формальных параметров. Они доступны только подпрограмме, в которой они описаны.

Рассмотрим использование процедуры на примере программы поиска максимума из двух целых чисел.

```
var   x,y,m,n : integer;

procedure MaxNumber (a,b: integer; var max: integer);
{в заголовке процедуры a,b-входные параметры, max-выходной параметр};
begin {начало процедуры}
if a>b then max := a  else max := b;
end;{конец процедуры}
begin {начало основной программы}
write('Введите x,y  ');
readln(x,y);
MaxNumber(x,y,m);   {формальный параметр a-получает значение x, b- получает значение y при вызове процедуры, m- получает значение результата из процедуры через параметр max};
MaxNumber(2,x+y,n);
writeln('m=' , m , 'n=' , n );
end. {Конец основной программы}
```

Аналогично задачу, но уже с использованием функций, можно решить так:

```
var  x,y,m,n :  integer;

function MaxNumber(a,b:  integer) :  integer;
```

```
var  max: integer;
begin    {начало функции}
if a>b then max := a else max := b;
MaxNumber := max;
{имя функции получает значение результата целого типа};
end; {конец функции}

begin {начало основной программы}
write('Введите x,y ');
readln(x,y);
m := MaxNumber(x,y);
{в основной программе результат функции копируется в переменную m};
n := MaxNumber(2, x+y);
writeln('m=' , m , 'n=' , n );
end.
```

Рекурсивные подпрограммы

Рекурсия – это способ организации вычислительного процесса, при котором подпрограмма обращается сама к себе, но с более «простыми» параметрами (измененными). Пример вычисления факториала числа (N! = N (N-1)!) с рекурсивным вызовом функции:

```
Program rec1;
Function fact (n : word) : word;
Begin  If n = 0 then fact := 1 else fact := n * fact (n-1);  End;
Begin
Write(‘Введите число ‘); Readln(n); Write(‘ Фасториал числа ‘,n,’равен: ‘,fact(n))
End.
```

В практических задачах глубина рекурсивных вызовов должна быть конечна и невелика. Большая глубина рекурсивных вызовов приводит к переполнению стека. Кроме того, подобно операторам цикла, рекурсивные вызовы могут приводить к «зацикливанию». Чтобы избежать этого в теле процедуры или функции должно быть некоторое управляющее условие с ветвью не содержащей рекурсивный вызов. Эта ветвь будет определять ограничение рекурсии (граничное условие).

Рекурсивные процедуры эффективны при работе со сложными динамическими структурами данных: списками, деревьями, графами.

2.13. Файловый тип данных

Описание типа **Var** <имя файловой переменной>**: file of** <Имя базового типа (типа элементов файла)>

Имя переменной файлового типа принято называть ***логическим именем файла***. Именно это имя используется в программе при обращении к его элементам. Кроме того, у файла есть имя, под которым он фигурирует в операционной системе (ОС) - ***физическое имя***. Установление связи "логическое имя - физическое имя" выполняется специальной процедурой (Assign). Одно логическое имя можно связывать в разное время с разными физическими, а это означает, что программу можно написать так, что её не надо будет переделывать при смене имени файла в ОС или при работе с другим файлом.

Операции с файлами реализуются обращением к стандартным процедурам:

Assign(F, FName)	Связывает логическое имя файла F с его физическим именем FName
Reset(F)	Открывает файл F для ввода
Rewrite(F)	Открывает файл F для вывода
Close(F)	Закрывает файл F
Read(F,R)	Чтение элемента файла F в переменную R
Write(F,R)	Запись элемента файла A из переменной R
Eof(F)	Возвращает булевское значение "Конец файла"
Eof(F)	Возвращает булевское значение "Конец файла" для файла F
Eoln(F)	Возвращает булевское значение "Конец строки" для файла F

Пример. Перенаправление ввода.

```
program InpRedir;
var Ch: Char;
begin
Assign(Input, 'Data.txt'); Reset(Input);
while not Eof(Input) do
   begin
   Read(Ch);
   Write(Ch);
   end;
end.
```

Data.txt:

AB
CDE

Экран монитора:

AB
CDE

Пример. Перенаправление вывода.

```
program OutRedir;
var Ch: Char;
begin
Assign(Output, 'Data.txt'); Rewrite(Output);
while not Eof(Input) do
   begin
   Read(Ch);
   Write(Ch);
   end;
end.
```

Клавиатура:

abc<Enter>
12345<Enter>
<Ctrl+Z><Enter>

Data.txt:

abc
12345

Рассмотрим ещё пример, иллюстрирующиq работу с текстовым файлом. Пусть требуется оценить объем текста рукописи, записанного на диск в так называемых издательских листах, которые вмещают по 40000 символов. В программе реализуем идею подсчета общего количества символов в тексте рукописи, пока не дойдем до конца файла, а затем при выводе ответа накопленное количество поделим на 40000.

```
program Rukopis;
var FTxt: Text;
      s: String;
    sum: LongInt;
begin
Write('Введите имя файла: '); Readln(s);
Assign(FTxt, s); Reset(FTxt);
```

```
Sum := 0;
while not Eof(FTxt) do
   begin
   Readln(FTxt, s);
   Inc(sum, Length(s));
end;
Close(FTxt);
WriteLn('Объем = ', sum/40000:6:2,' издат. листов');
end.
```

2.14. Работа с двумерными массивами

2.14.1. Действия над элементами массива

При работе с двумерным массивом указываются два индекса, например, B[4,4]. Индексированные элементы массива называются индексными переменными и могут быть использованы так же, как и простые переменные, например, они могут находиться в выражениях в качестве операндов, использоваться в операторах for, while, repeat, входить в качестве параметров в операторы read, readln, write, writeln; им можно присваивать любые значения, соответствующие их типу.

Копированием массивов называется присвоение значений всех элементов одного массива всем соответствующим элементам другого массива. Копирование можно выполнить одним оператором присваивания, например A:=B, если массивы идентичны по структуре, или с помощью оператора for поэлементно:

```
For i:=1 to 5 do
For j:=1 to 7 do
A[i,j]:=B[i,j];
```

Рассмотрим задачу формирования одномерных массивов из данных двумерного массива. Пусть M производителей обеспечивают однородной продукцией N потребителей. Задана матрица C[i,j], i=1..M, j=1..N объёмов поставок i-го производителя j-му потребителю. Определить объёмы поставок по производителям, по потребителям и общий объем поставок.

```
program ProCons;
const M = 3;       { Количество производителей }
      N = 4;       { Количество потребителей }
type TRow = array[1..N] of Real;
     TCol = array[1..M] of Real;
     TMat = array[1..M] of TRow;
```

```
var  C:   TMat;    { Матрица поставок }
     Pro: TCol;    { Вектор поставок по производителям }
     Con: TRow;    { Вектор поставок по потребителям }
     i:   Integer; { Индекс производителя }
     j:   Integer; { Индекс потребителя }
     V:   Real;    { Общий объём поставок }
begin
{ Ввод данных }
WriteLn('Введите данные (', M, ' строк по ', N, ' элементов):');
for i := 1 to M do
   begin
   for j := 1 to N do Read(C[i,j]);
   ReadLn;
   end;
{ Определение вектора Pro }
for i := 1 to M do
   begin
   Pro[i] := 0;
   for j := 1 to N do Pro[i] := Pro[i] + C[i,j];
   end;
{ Определение вектора Con }
for j := 1 to N do
   begin
   Con[j] := 0;
   for i := 1 to M do Con[j] := Con[j] + C[i,j];
   end;
{ Определение общего объёма поставок }
V := 0;
for i := 1 to M do V := V + Pro[i];
{ Вывод данных }
WriteLn;
for j := 1 to N do Write(j:6);
WriteLn;
for i := 1 to M do
   begin
   Write(i:2);
   for j := 1 to N do Write(C[i,j]:6:1);
   WriteLn(Pro[i]:8:1);
   end;
Write('  ');
for j := 1 to N do Write(Con[j]:6:1);
WriteLn(V:8:1);
```

end.

2.14.2. Действия над идентичными двумерными массивами

Выражение	Результат
A=B	True, если значение каждого элемента массива A равно соответствующему значению элемента массива B.
A<>B	True, если хотя бы одно значение элемента массива A не равно значению соответствующего элемента массива B.
A:=B	Все значения элементов B присваиваются соответствующим элементам массива A. Значения элементов массива B остаются неизменны.

2.14.3. Использование процедуры ввода и процедуры вывода элементов двухмерного массива

```
  PROGRAM  P_r_i_m_e_r_2;
    const M = 2; { Количество строк в массиве    }
          N = 3; { Количество столбцов в массиве }
    type  T = Array [1..M,1..N] of Integer;
    var   A  : T;
PROCEDURE Wwod_Mass (var S: T);
  { Процедура ввода с клавиатуры элементов массива }
  { процедура с параметрами}
  { S - параметр, вызываемый по адресу}
    var  i,j: Integer;
  BEGIN
    WriteLn ('Введите массив: ');
    For i:=1 to M  do
      For j:=1 to N  do
        begin  Write ('Введите S[',i,',',j, ']: '); ReadLn (S[i,j])  end
  END;
PROCEDURE Wywod_Mass (S: T);
  { Процедура вывода на экран элементов массива }
  { процедура с параметрами                    }
  { S - параметр, вызываемый по значению       }
    var  i,j: Integer;
  BEGIN
    WriteLn ('Вот Ваш массив: ');
    For i:=1 to M  do
      begin  For j:=1 to N  do  Write (S[i,j],' '); WriteLn  end
```

```
  END;
BEGIN
 Wwod_Mass (A);{Вызов процедуры ввода элементов массива}
 Wywod_Mass(A){Вызов процедуры вывода элементов массива}
  END.
```

2.15. Стандартный модуль Graph

Модуль Graph предназначен для работы с монитором в графическом режиме. В этом режиме весь экран монитора представляется в виде прямоугольной матрицы графических элементов - ***пикселов***.
В модуле Graph положение на экране отдельного пиксела задаётся его целочисленными координатами в системе координат (СК) экрана. Начало этой СК находится в левом верхнем углу экрана. Ось 0X проходит горизонтально от начала СК вправо, а ось 0Y - вертикально от начала СК вниз.
Кроме положения на экране каждый пиксел характеризуется цветом. Цвет пиксела кодируется целым числом из диапазона 0..15 (например, 1 - голубой, 2 - зелёный, 4 - красный). Кроме того, существует понятие "цвет фона", который кодируется целым числом из диапазона 0..7.
Интерфейс модуля Graph достаточно велик и сложен. Для первого знакомства с графическим вводом-выводом нам будет достаточно небольшой его части.

unit Craph;
...
procedure arc (x, y, un, uk, r) — вычерчивает дугу окружности радиусом r с центром в точке с координатами (x, y) от начального угла un до конечного угла uk;
procedure bar (xi, yl, x2, y2) — вычерчивает столбец но указанным координатам точек;
procedure ClearDevice;{Заполняет экран цветом фона}
procedure CloseGraph; {Закрывает графический режим}
procedure fillellipse (x, y, xr, yr) — вычерчивает эллипс с центром в точке (x, y) и радиусами xr и yr;
function GetMaxX:Integer;{Возвр.макс.значение координаты X}
function GetMaxY:Integer;{Возвр.макс.значение координаты Y}
procedure SetColor(Color:Word);{Устанавливает цвет линий и литер текста}
procedure SetBkColor(Color:Word); {Уст. цвет фона}
procedure Line(X1,Y1,X2,Y2:Integer); {Проводит линию}
procedure Circle(X,Y:Integer; Radius:Word); {Рисует окружность}

procedure Rectangle(X1,Y1,X2,Y2:Integer); {Рисует прямоугольник}
procedure PutPixel(X,Y:Integer; Color:Word);{Выводит пиксел указанного цвета}
procedure OutTextXY(X,Y:Integer;TextString:string); {Выводит текст с указанной позиции}

Рассмотрим взаимосвязь некоторой заданной СК с СК экрана. В заданной СК с помощью четырёх параметров *xmin*, *xmax*, *ymin* и *ymax* определим прямоугольную область. Рассмотрим задачу отображения этой прямоугольной области на экран монитора. Исходя из предположения, что угловые точки отображаемой области должны отобразиться в угловые пикселы графической матрицы экрана, получим следующие соотношения:
mx = GetMaxX/(xmax - xmin), my = GetMaxY/(ymax - ymin),

*xs = Round((-xmin+x)*mx), ys = Round((ymax-y)*my),*

где *x, y* - координаты некоторой точки в отображаемой СК, *xs, ys* - координаты отображения точки в СК экрана, *mx* и *my* - масштабы преобразования по соответствующим осям.

Эти соотношения использованы в следующем примере программы с применением модуля Graph. Программа Curve.pas строит график функции *y = Sin(x)* на отрезке *(-2*Pi, 2*Pi)*.

```
program Curve;
uses Graph;
const
  xmin = -6.5; xmax = 6.5; { Размеры заданной        }
  ymin = -1.5; ymax = 1.5; { области }
  a = -2*Pi;   b = 2*Pi;  { Отрезок построения графика }
  n = 50;                 { Кол. интервалов графика   }
  h = (b - a)/n;          { Длина одного интервала    }
var
  mx,my: Real;            { Масштабы                  }
  x,y:   Real;       { Текущая точка кривой в зад. СК  }
  xs,ys: Integer;    { Текущая точка графика а СК экрана }
  xc,yc: Integer;    { Начало заданной СК в СК экрана   }
  i:     Integer;    { Номер точки графика        }

procedure GrInit;     {Инициирует графическую систему   }
var   GraphDriver, GraphMode, ErrorCode: integer;
begin
GraphDriver:=Detect; {автоопределение типа адаптера}
InitGraph(GraphDriver,GraphMode,'');
ErrorCode:=GraphResult;{обработка ошибок инициализации}
if ErrorCode<>grOk then
```

```
  begin
  WriteLn('Ошибка графики: ',GraphErrorMsg(ErrorCode));
  WriteLn('Программа остановлена.');
  ReadLn;
  Halt(1);
  end
end; { GrInit }

procedure ToScr(x,y:Real; var xs,ys:Integer);
  { Отображает точку из заданной СК в СК экрана }
begin
xs:=Round((-xmin+x)*mx); ys:=Round((ymax-y)*my);
end; { ToScr }

begin
GrInit;  { Инициируем графическую систему }
{ Определяем масштабы по осям 0X и 0Y}
mx := GetMaxX/(xmax - xmin); my := GetMaxY/(ymax - ymin);
{ Определяем координаты начала заданной СК в СК экрана }
ToScr(0,0,xc,yc);
SetColor(1);             { Устанавливаем цвет линий }
SetBkColor(7);           { Устанавливаем цвет фона  }
Line(0,yc,GetMaxX,yc);    { Проводим ось 0X          }
Line(xc,0,xc,GetMaxY);    { Проводим ось 0Y          }
OutTextXY(20,20,'Y = Sin(X)');{ Выводим заголовок графика}
 РИСУЕМ ГРАФИК }
SetColor(12);
x:=a; y:=Sin(x);
ToScr(x,y,xs,ys);
MoveTo(xs,ys);
for i:=1 to n do
  begin
  x:=x+h; y:=Sin(x);
  ToScr(x,y,xs,ys);
  LineTo(xs,ys);
  end;
ReadLn;       { Для выхода из гр. режима надо нажать Enter }
CloseGraph;   { Возвращаемся в алфавитно-цифровой режим    }
end.
```

3. МЕТОДЫ И АЛГОРИТМЫ ЧИСЛЕННОГО РЕШЕНИЯ УРАВНЕНИЙ

3.1. Основы теории погрешностей

На общую погрешность решения задачи влияет целый ряд факторов. Рассмотрим основные из них.

Пусть R – точное значение результата решения некоторой задачи. Из-за того, что построенная математическая модель не тождественна реальному объекту, а также по причине неточности исходных данных, вместо R будет получен результат, который обозначим R_1. Образовавшаяся таким образом погрешность $\varepsilon_1 = R - R_1$ уже не может быть устранена в ходе последующих вычислений (так называемая неустранимая погрешность). Приступив к решению задачи в рамках математической модели, выбирают приближённый метод (например, численный) и ещё не приступив к вычислениям, допускают новую погрешность, приводящую к получению результата R_2. Погрешность $\varepsilon_2 = R_1 - R_2$ называется погрешностью метода. Реализовав метод получают R_3. Погрешность $\varepsilon_3 = R_2 - R_3$ называется вычислительной погрешностью. В случае использования ЭВМ происходит принудительное округление, которое связано с конечностью разрядной сетки машины. Полная погрешность получается как сумма всех погрешностей

$\varepsilon = \varepsilon_1 + \varepsilon_2 + \varepsilon_3$.

При решении конкретных задач те или иные виды погрешностей могут либо отсутствовать совсем, либо влиять на окончательный результат незначительно. Тем не менее, для исчерпывающего представления о точности результата в каждом случае необходим полный анализ погрешностей всех видов.

3.1.1. Связь между числом верных знаков и погрешностью числа

Пусть положительное приближенное число a содержит s верных десятичных знаков. Тогда его десятичное разложение имеет вид

$$a = n_1 10^r + n_2 10^{r-1} + \ldots + n_s 10^p,$$

причем целые числа r, p ($r > p$) и s связаны очевидным равенством $p - r = 1 - s$.

Пусть число получено с округлением, тогда предельная относительная погрешность

$$\delta_a = \frac{10}{2n_1} 10^{-s}.$$

3.1.2. Погрешность вычисления значений функции

Пусть известны погрешности некоторой системы величин. Требуется определить погрешность данной функции этих величин. Пусть задана дифференцируемая функция $y = f(x_1, x_2, ..., x_n)$ и пусть $|\Delta x_i|$ ($i = 1, 2, \ldots, n$) – абсолютные погрешности аргументов функции. Тогда абсолютная погрешность функции определяется как разность

$$|\Delta u| = |y - y^*| = |f(x_1 + \Delta x_1, x_2 + \Delta x_2, \ldots, x_n + \Delta x_n) - f(x_1, x_2, \ldots, x_n)|,$$

где x_i^* – приближенные значения ее аргументов, для которых $|x_i - x_i^*| \le \Delta(x_i^*)$ – известные абсолютные погрешности.

Для погрешности приближенного значения функции $y^* = f(x_i^*, x_2^*, \ldots, x_n^*)$ получаем $y - y^* \approx \sum_{i=1}^{n} a_i^* (x_i - x_i^*)$.

3.1.3. Погрешности результатов основных арифметических действий

Теорема 1. Абсолютная погрешность суммы нескольких приближенных чисел не превышает суммы абсолютных погрешностей этих чисел.

Доказательство: Пусть $a_1, a_2, \ldots, a_n$ – данные приближенные числа. Рассмотрим их алгебраическую сумму

$$U = \pm a_1 \pm a_2 \pm ... \pm a_n.$$

Очевидно, что истинная ошибка результата

$$\Delta U = \pm \Delta a_1 \pm \Delta a_2 \pm ... \pm \Delta a_n,$$

где Δa – истинная ошибка a_i.

Следовательно, абсолютная погрешность будет равна

$$|\Delta U| \le |\Delta a_1| + |\Delta a_2| + ... + |\Delta a_n|.$$

Следствие. *За предельную абсолютную погрешность алгебраической суммы можно принять сумму предельных абсолютных погрешностей слагаемых.*

Следовательно, при сложении и вычитании абсолютные погрешности складываются.

Так как $|\Delta a_i| \le \Delta_{a_i}$, то за предельную относительную погрешность можно принять

$$\delta U = \frac{\Delta U}{A} = \frac{\Delta_1 + \Delta_2 + ... + \Delta_n}{A_1 + A_2 + ... + A_n}.$$

3.1.4. Погрешность степени

Предельная относительная погрешность n-ной степени приближенного числа (n – натуральное число) равна произведению показателя степени n на предельную относительную погрешность основания.

При возведении приближенного числа в степень в результате следует оставить столько значащих цифр, сколько верных значащих цифр содержится в основании степени.

3.1.5. Погрешность произведения

Пусть задана функция $y = f(x_1, x_2) = x_1 \cdot x_2$. Тогда абсолютная погрешность $\Delta\left(y^*\right) = \left|x_2^*\right| \cdot \Delta(x_1^*) + \left|x_1^*\right| \cdot \Delta(x_2^*)$. Относительная погрешность

$$\delta(y*) = \frac{\Delta(y*)}{\left|x_1^* \cdot x_2^*\right|} = \frac{\Delta(x_1^*)}{\left|x_1^*\right|} + \frac{\Delta(x_2^*)}{\left|x_2^*\right|} = \delta(x_1^*) + \delta(x_2^*).$$

Относительная погрешность произведения нескольких приближенных чисел не превышает суммы относительных погрешностей этих чисел.

$$\delta \le \delta_1 + \delta_2 + ... + \delta_n.$$

Предельная относительная погрешность произведения равна сумме предельных относительных погрешностей сомножителей. Этот результат имеет большое значение, так как предельная относительная погрешность тесно связана с числом верных знаков. Поэтому при умножении нет смысла брать сомножители с различным числом верных знаков.

3.1.6. Погрешность частного

Пусть задана функция $y = f(x_1, x_2) = \frac{x_1}{x_2}$. Тогда абсолютная погрешность может быть выражена формулой $\Delta(y^*) = \frac{1}{\left|x_2^*\right|} \cdot \Delta(x_1^*) + \frac{\left|x_1^*\right|}{\left|x_2^*\right|^2} \cdot \Delta(x_2^*)$.

Относительная погрешность $\delta(y^*) = \frac{\Delta(x_1^*)}{\left|x_1^*\right|} + \frac{\Delta(x_2^*)}{\left|x_2^*\right|} = \delta(x_1^*) + \delta(x_2^*)$.

3.2. Приближенное решение нелинейных уравнений

3.2.1. Отделение корней уравнения

При разработке алгоритмов решения конкретных инженерных задач часто приходится прибегать к методам численного анализа. Рассмотрим наиболее простые численные методы решения некоторых классов задач, которые используются в данной работе. Для рассматриваемых методов приводятся реализующие их подпрограммы с необходимыми инструкциями. Использование некоторых методов иллюстрируется примерами.

Корнем уравнения $f(x) = 0$ называется такое значение $x = \xi$ аргумента функции $f(x)$ при котором это уравнение обращается в тождество: $f(\xi) = 0$. Корень уравнения геометрически представляет собой абсциссу точки пересечения, касания (рис. 3.1) или другой общей точки (рис. 3.2) графика функции $y = f(x)$ и оси OX.

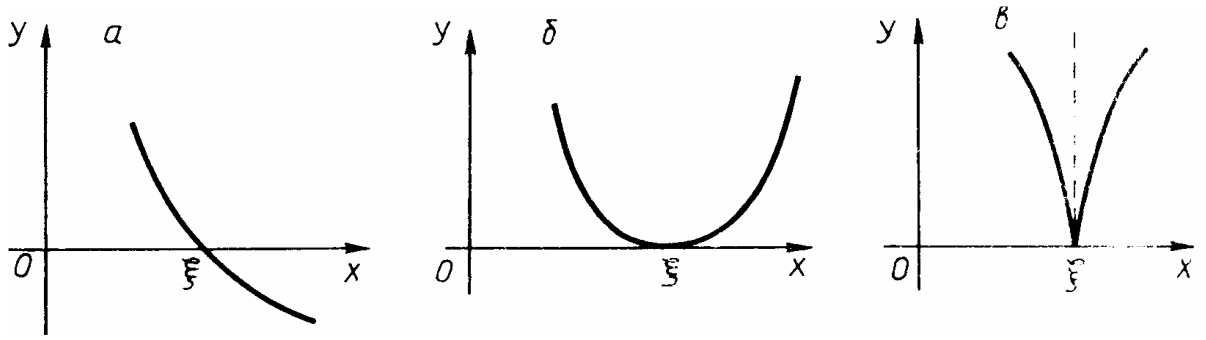


Рис. 3.1. Графики функции $y = f(x)$

Отделить корень уравнения – значит найти такой конечный промежуток, внутри которого имеется единственный корень данного уравнения. Отделение корней уравнения можно выполнить аналитически, исследовав данную функцию или графически, построив график функции $y = f(x)$, по которому можно судить о том, в каких промежутках находятся точки пересечения его с осью OX.

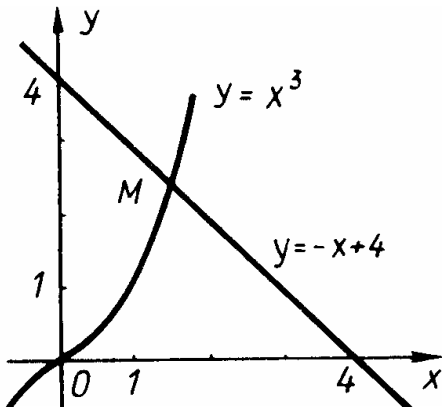


Рис. 3.2. Пример отделения корня методом разбиения функции на 2

3.2.2. Уточнение корня методом половинного деления

Допустим, что искомый корень уравнения отделен, т. е. найден отре-

зок [*a*, *b*], на котором имеется один и только один корень уравнения. Любую точку этого отрезка можно принять за приближенное значение корня. Погрешность такого приближения не превосходит длины [*a*, *b*]. Следовательно, задача отыскания приближенного значения корня с заданной точностью ε сводится к нахождению отрезка [*a*, *b*], содержащего только один корень уравнения. Эту задачу обычно называют задачей уточнения корня.

Задача уточнения корня сводится к делению отрезка пополам, с каждым шагом выделяя новый кусок отрезка [*a*, *c*] или [*c*, *b*] до тех пор, пока значение не будет $x \approx 0$. Примерная схема алгоритма решения уравнения $f(x) = 0$ методом половинного деления представлена на рис. 3.3.

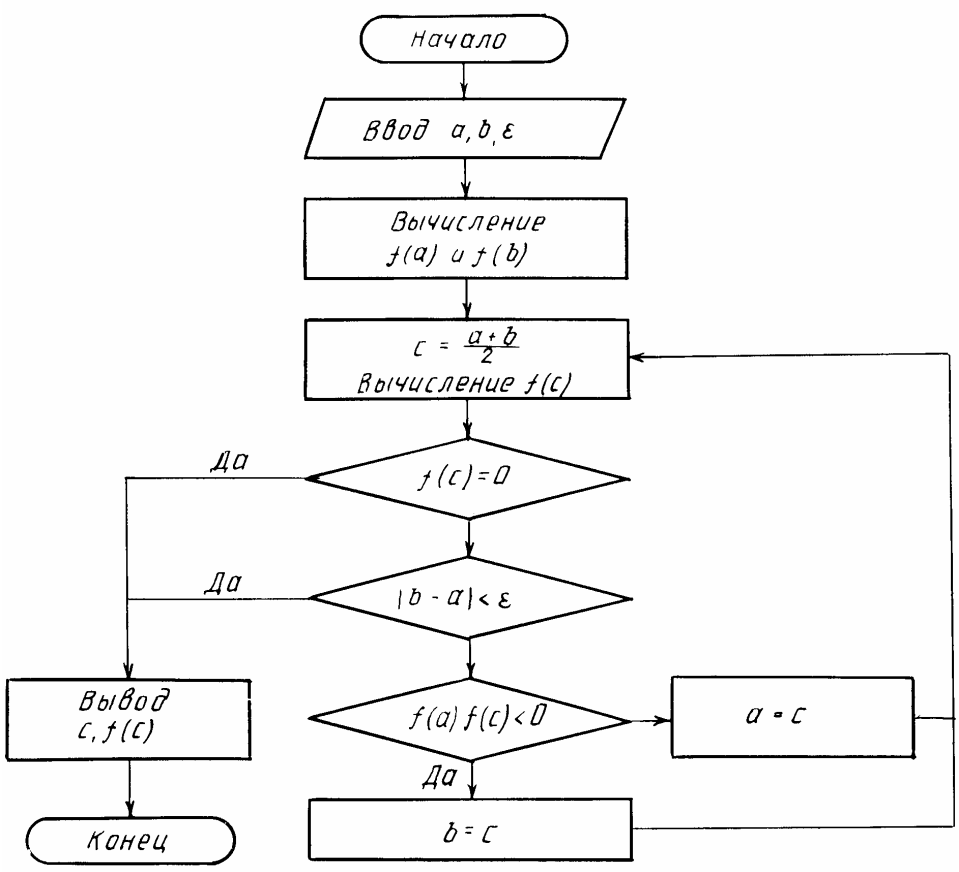


Рис. 3.3. Блок-схема алгоритма решения
нелинейного уравнения методом половинного деления

```
Program MPD;
 VAR A,B,C,FA,FB,FC,EPS: REAL;
 BEGIN
  Writeln('Метод половинного деления');
  Write('Начальное значение интервала A='); Readln(A);
  Write('Конечное значение интервала B='); Readln(B);
```

```
  Write('Погрешность EPs='); Readln(EPS);
  FA:=A*SQR(A)+SQR(A)+A-6;
  FB:=B*SQR(B)+SQR(B)+B-6;
 REPEAT
  C:=(A+B)/2;
  FC:=C*SQR(C)+SQR(C)+C-6;
  WRITE('C=',C,'  FC=',FC); WRITELN;
  IF FC=0 OR ABS(FC)<EPS THEN EXIT;
  IF FA*FC<0 THEN B:=C ELSE A:=C;
 UNTIL ABS(B-A)<EPS;
END.
```

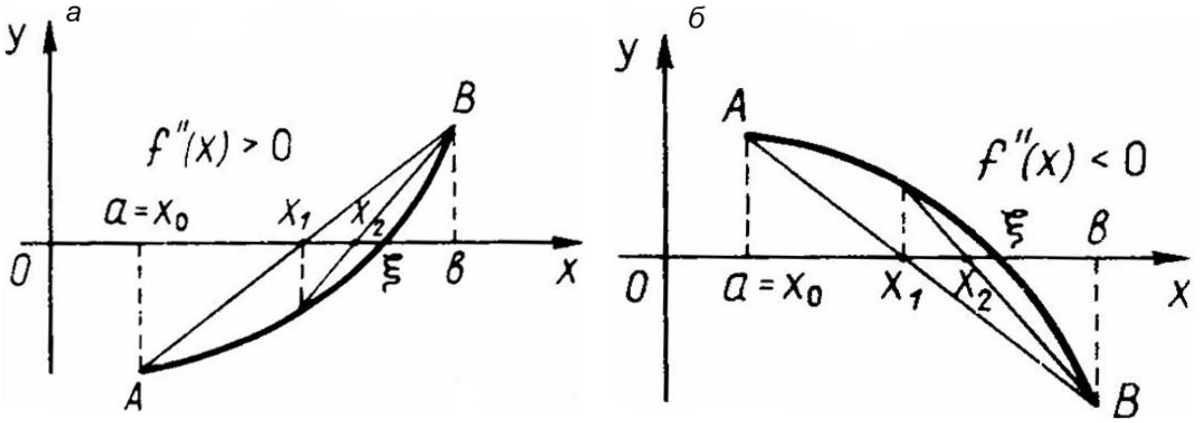
3.2.3. Метод хорд (секущих)

Метод хорд, или метод секущих, приближенного решения уравнения $y = f(x)$ имеет следующую геометрическую иллюстрацию: вместо точки пересечения оси Ox и графика функции $f(x)$, входящей в это уравнение, рассматривается точка пересечения данной оси и отрезка прямой, соединяющей концы дуги графика (рис. 3.4). Если известно $(n - 1)$-е приближение, то n-ное вычисляется по формуле

$$x_n = \frac{bf(x_{n-1}) - x_{n-1}f(b)}{f(x_{n-1}) - f(b)} \quad \text{при } f(b)f''(x) > 0$$

где $n = 1, 2, 3, \ldots,$

$$\text{или} \quad x_n = \frac{af(x_{n-1}) - x_{n-1}f(a)}{f(x_{n-1}) - f(a)} \quad \text{при } f(a)f''(x) > 0.$$



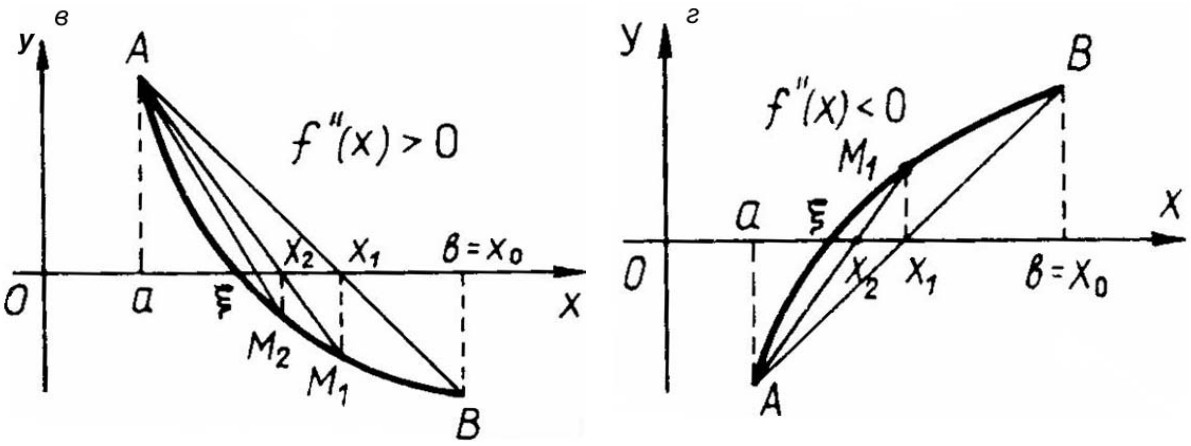


Рис. 3.4. Графики, иллюстрирующие метод хорд

Оценка абсолютной погрешности определяется формулой $|\xi - x_n| \le \dfrac{|f(x_n)|}{\mu}$, $\mu = \max|f'(x)|$, $f'(x) \ne 0$.

Пример. Пусть требуется решить методом секущих нелинейное уравнение: e^x-3-cos(x)=0, с точностью $\varepsilon=10^{-4}$.

Для приближенного решения уравнения используем формулу секущих:

$$x_{n+1} = x_n - \frac{x_{n-1} - x_n}{f(x_{n-1}) - f(x_n)},$$

где $f(x)=e^x$-3-cos(x).

Проверяем условия сходимости:

$$f'(x) = e^x + \sin \quad x > 0, \quad x \in [0,2];$$
$$f''(x) = e^x + \cos \quad x > 0, \quad x \in [0,2].$$

Так как необходимое и достаточное условия выполнены на всем отрезке локализации, то для решения уравнения можем использовать метод секущих.

```
program uravnen;
   {Решение линейных уравнений методом секущих}
   {x0 ,x1 - нулевое и первое приближения значения корня}
   {e - точность решения}
   {x - новое приближение корня}
var x0,x1,e,x of real;
var i: integer;
        function f(y : real) : real;
        {функция для заданного нелинейного уравнения}
            begin
```

```
            f:=exp(y)-3-cos(y);
          end;       {f}
begin
  writeln('Введите задаваемую точность e');
readln (e);
  writeln('Введите нулевое и первое приближения);
readln (x0,x1);
     i:=0;
     while abs(x1-x0)>=e do
        begin
          x:=x1-(x0-x1)/(f(x0)-f(x1));
          x0:=x1;
          x1:=x;
          i:=i+1;
        end;
    writeln('Решение нелинейного уравнения методом секущих');
  writeln('корень x=',x,'    точность e=',e,'    число итераций i=',i);
end.
```

3.2.4. Метод касательных

Метод касательных (или метод Ньютона) отличается от метода хорд тем, что здесь рассматривается не секущая, соединяющая концы дуги графика, а касательная к графику. Точка пересечения касательной с осью *OX* дает приближенное значение корня (рис. 3.5).

В методе касательных $(n + 1)$-е приближение вычисляется по формуле $x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}$, $(n = 1, 2, 3, \ldots)$, в которой за нулевое приближение x_0 принимается такое значение из отрезка $[a, b]$, для которого выполняется условие при $f(x_0)f''(x) > 0$.

Оценка погрешности, как и в методе хорд, определяется формулой

$$|\xi - x_n| \le \frac{|f(x_n)|}{\mu}, \text{ где}$$

$$\mu = \max|f'(x)|, \quad f'(x) \ne 0.$$

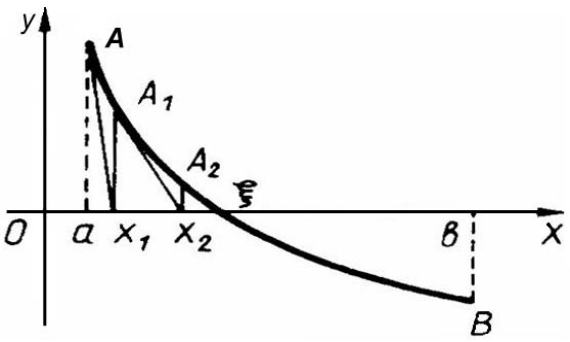


Рис. 3.5. График, иллюстрирующий метод касательных

```
Program Newton;
  VAR A,B,FX,F1X,XN,EPS: REAL;
  BEGIN
    Writeln('Метод Ньютона');
```

```
   Write('Начальное значение интервала A='); Readln(A);
   Write('Конечное значение интервала B='); Readln(B);
   Write('Погрешность EPs='); Readln(EPS);
   X:=0; XN:=1;
   WHILE ABS(X-XN)>EPS DO
        BEGIN
        FX:=X*SQR(X)+SQR(X)+X-6;
        F1X:=3*SQR(X)+2*X+1;
        XN:=X-FX/F1X;
         X:=XN
        END;
  WRITELN ('X=',X,'  FX=',FX) ;
END.
```

3.2.5. Метод итераций

Если каким-либо способом получено приближенное значение x_0 корня уравнения $f(x) = 0$, то уточнение корня можно осуществить методом последовательных приближений или методом итераций. Для этого уравнение $f(x) = 0$ представляют в виде

$$x = \varphi(x),$$

что всегда можно сделать и притом многими способами, например,

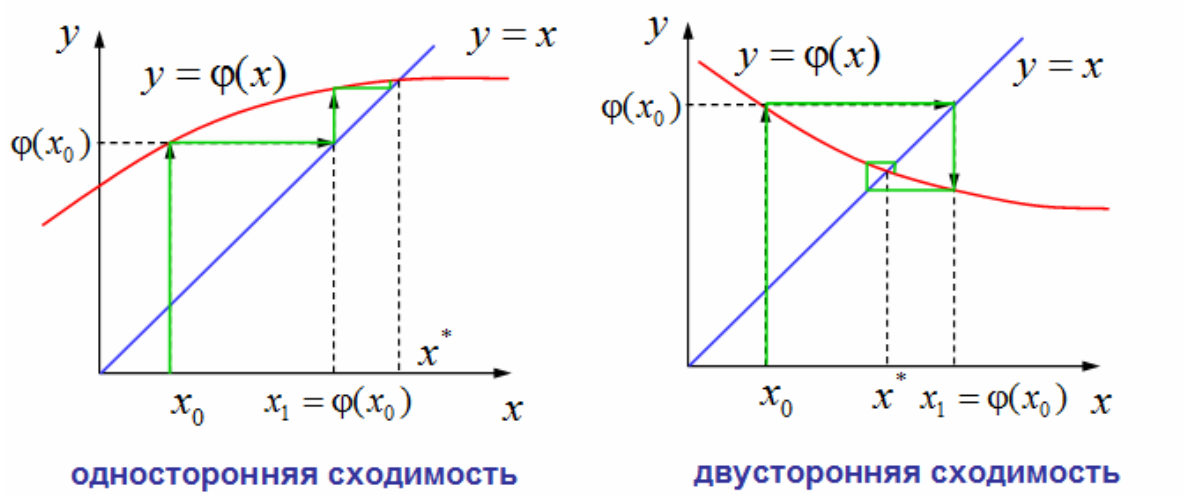
$$x = x + cf(x),$$

где c – произвольная постоянная,

$$x_n = \varphi(x_{n-1}).$$

Процесс последовательного вычисления чисел x_n (n = 1, 2, 3, ...) по этой формуле называется методом последовательных приближений или методом итераций. Процесс итераций сходится, если выполнено условие $|\varphi'(x)| \le q < 1$ на отрезке $[a, b]$, содержащем корень.

Сходящийся итерационный процесс:



Расходящийся итерационный процесс:

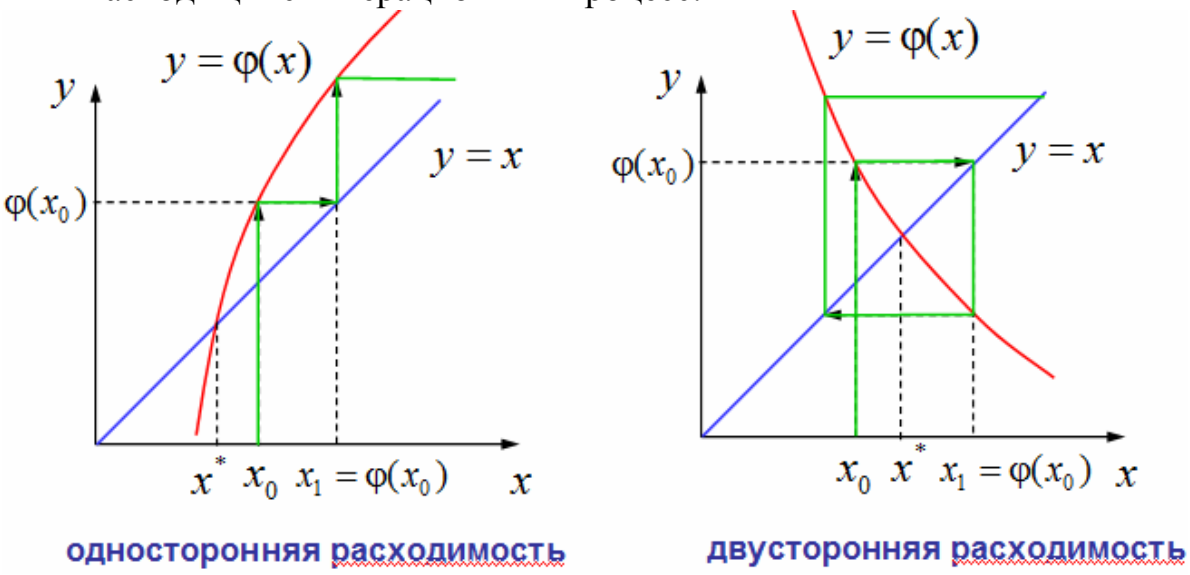


Рис. 3.6. Графики, иллюстрирующие условия сходимости-расходимости итерационных процессов

```
Program Iter;
  VAR A,B,FX,F1X,XR,EPS: REAL;
  BEGIN
    Writeln(‘Метод итераций’);
    Write('Начальное значение интервала A='); Readln(A);
    Write('Конечное значение интервала B='); Readln(B);
    Write('Погрешность EPs='); Readln(EPS);
    X:=0; XR:=1;
    WHILE ABS(X-XR)>EPS DO
        BEGIN
        XR:=X*SQR(X)+SQR(X)+X-6;
```

```
        X:=XR
      END;
  WRITELN ('X=',X) ;
END.
```

3.2.6. Комбинированный метод

Пусть $f''(x)$ сохраняет знак на интервале (a, b). Было показано, что уточнения значения корня методом хорд и методом касательных производятся приближением к корню с разных сторон: одно с недостатком, другое с избытком. Комбинированный метод заключается в последовательном применении метода хорд и метода касательных. При этом метод хорд и метод касательных дают приближенные значения, расположенные по разные стороны от искомого корня.

Если, например, график функции расположен, как на рис. 3.7, то, взяв в качестве исходного отрезок $[a, b]$, по методу хорд находим приближение b, а по методу касательных – приближение a. В результате получаем отрезок $[a, b]$, содержащий искомый корень. Значения a и b определяются по формулам:

$$a_1 = a - \frac{f(a)}{f'(a)}, \qquad b = a - \frac{f(a)}{f(b) - f(a)}(b - a).$$

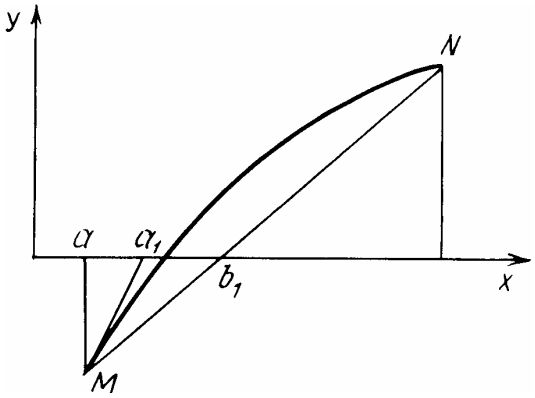


Рис. 3.7. График, иллюстрирующий комбинированный метод решения нелинейных уравнений

Примерная схема алгоритма решения уравнения комбинированным методом приведена на рис. 3.8.

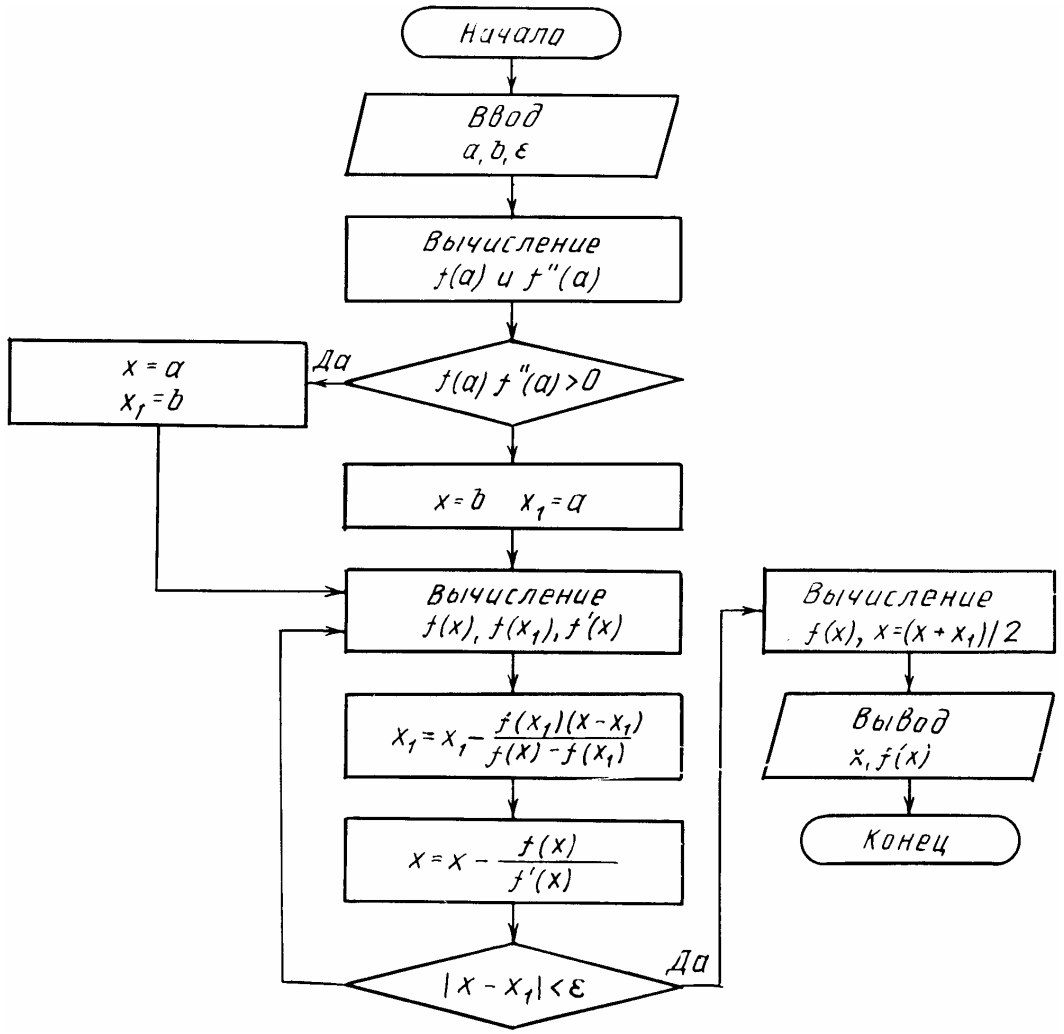


Рис. 3.8. Блок-схема алгоритма решения
нелинейного уравнения комбинированным методом

3.3. Методы решения систем линейных уравнений

Инженеру часто приходится решать алгебраические и трансцендентные уравнения, что может представлять собой самостоятельную задачу или являться частью более сложных задач. В обоих случаях практическая ценность метода в значительной мере определяется быстротой и эффективностью полученного решения. Выбор подходящего метода для решения уравнений зависит от характера рассматриваемой задачи. Задачи, сводящиеся к решению алгебраических и трансцендентных уравнений, можно классифицировать по числу уравнений (рис. 3.9) и в зависимости от предлагаемого характера и числа решений.

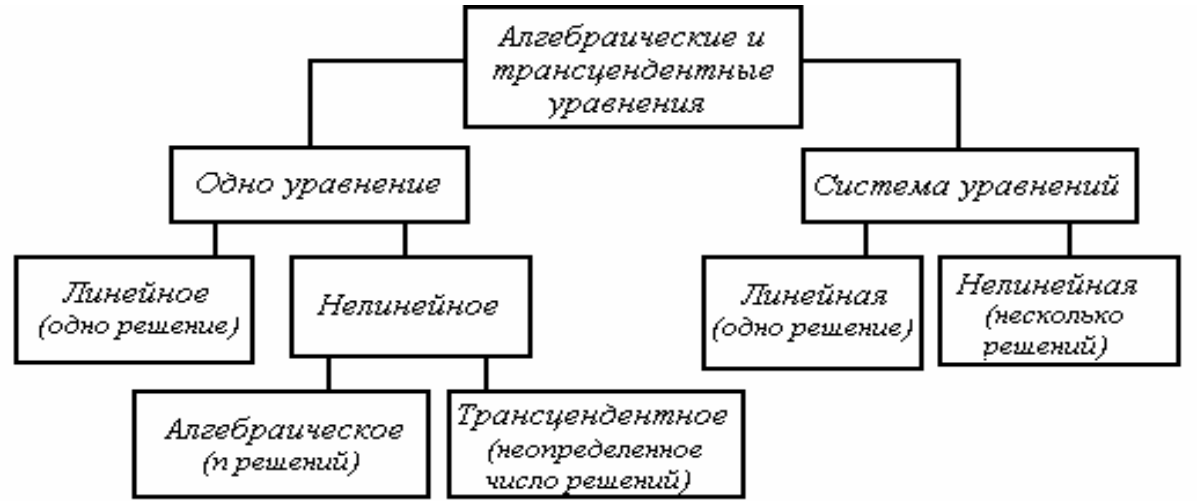


Рис. 3.9. Классификация уравнений

Одно уравнение будем называть *линейным*, *алгебраическим* или *трансцендентным* в зависимости от того, имеет ли оно одно решение, *n* решений или неопределенное число решений. Систему уравнений будем называть *линейной* или *нелинейной* в зависимости от математической природы входящих в нее уравнений.

Способы решения систем линейных уравнений делятся на две группы:

1) *точные методы*, представляющие собой конечные алгоритмы для вычисления корней системы (решение систем с помощью обратной матрицы, правило Крамера, метод Гаусса и др.),
2) *итерационные методы*, позволяющие получить решение системы с заданной точностью путем сходящихся итерационных процессов (метод итерации, метод Зейделя и др.).

Вследствие неизбежных округлений результаты даже точных методов являются приближенными. При использовании итерационных методов, сверх того, добавляется погрешность метода. Эффективное применение итерационных методов существенно зависит от удачного выбора начального приближения и быстроты сходимости процесса.

Пусть дана система линейных алгебраических уравнений (СЛАУ) с *n* неизвестными:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = a_{1n+1} \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = a_{2n+1} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = a_{nn+1} \end{cases} \quad (1)$$

или в матричной форме $AX = B$,

где $A = (a_{ij}) = \begin{bmatrix} a_{11} \_ a_{12} \ldots a_{1n} \\ a_{21} \_ a_{22} \ldots a_{2n} \\ \ldots\ldots\ldots\ldots\ldots \\ a_{n1} \_ a_{n2} \ldots a_{nn} \end{bmatrix}$ – есть матрица коэффициентов; $B = \begin{bmatrix} a_{1n+1} \\ a_{2n+1} \\ \ldots \\ a_{nn+1} \end{bmatrix}$

и $X = \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{bmatrix}$ – соответственно столбец свободных членов и столбец неизвестных.

Если матрица A неособенная, т. е. $\det A \neq 0$, то система имеет единственное решение. Методы решения таких систем разбиваются на две группы: точные и приближенные.

3.3.1. Метод Гаусса

Пусть в системе (1) коэффициент $a_{11} \neq 0$. Назовем a_{11} ведущим элементом первой строки. Умножив первое уравнение системы (1) на $-a_{21}/a_{11}$ и прибавив ко второму, получим новое уравнение, в котором коэффициент при x_1 обращается в нуль. Умножив первое уравнение системы на $-a_{31}/a_{11}$ и прибавив к третьему, получим новое уравнение, в котором коэффициент при x_1 обращается в нуль. Аналогично преобразовываем остальные уравнения.

Полагая, что в полученной системе коэффициент $a'_{22} \neq 0$. Умножив второе уравнение новой системы на $-a'_{32}/a'_{22}$ и прибавив к третьему, получим новое уравнение, в котором коэффициент при x_2 обращается в нуль. Аналогично преобразовываем остальные уравнения. В конечном итоге получим систему, эквивалентную треугольной матрице:

$$\begin{cases} x_1 + a'_{12}x_2 + \ldots + a'_{1n}x_n = a'_{1n+1} \\ x_2 + \ldots + a'_{2n}x_n = a'_{2n+1} \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ x_n = a'_{nn+1} \end{cases}$$

Примерная схема алгоритма решения СЛАУ методом Гаусса приведена на рис. 3.10.

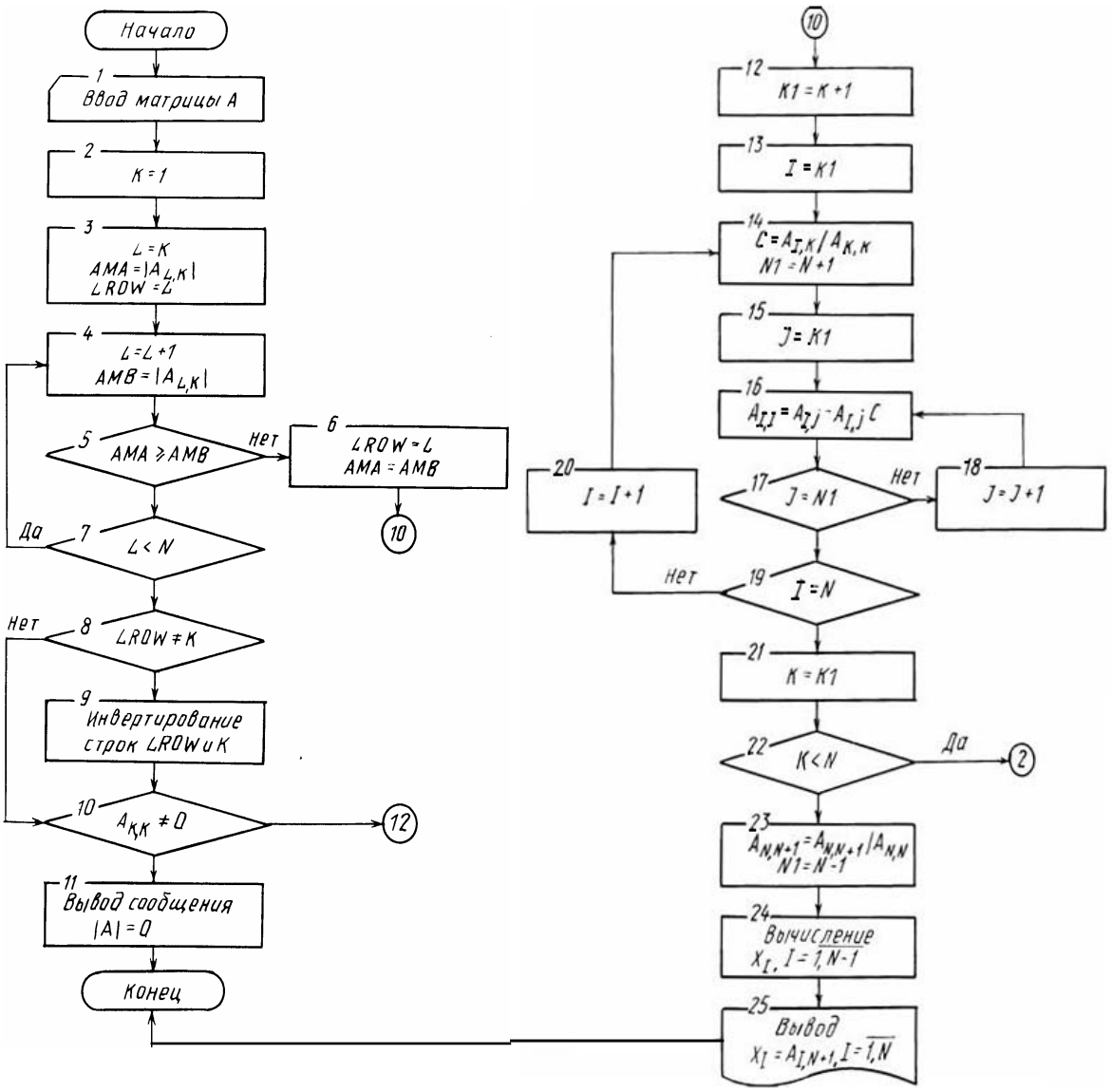


Рис. 3.10. Блок-схема алгоритма решения СЛАУ методом Гаусса

3.3.2. Метод простой итерации

Пусть задана система линейных уравнений:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = a_{1n+1}, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = a_{2n+1}, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = a_{nn+1}. \end{cases}$$

Для реализации метода итераций необходимо привести систему к виду

$$\begin{cases} x_1 = -\dfrac{1}{a_{11}}(a_{12}x_2 + \ldots + a_{1n}x_n - a_{1n+1}), \\ x_2 = -\dfrac{1}{a_{22}}(a_{21}x_1 + \ldots + a_{2n}x_n - a_{2n+1}), \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_n = -\dfrac{1}{a_{nn}}(a_{n1}x_1 + a_{n2}x_2 + \ldots - a_{nn+1}). \end{cases}$$

Коэффициенты полученной системы должны удовлетворять следующему условию сходимости: $|a_{ii}| > \sum_{i \neq j} |a_{ij}|$, $i = 1 \ldots n$, т.е. модули диагональных коэффициентов для каждого уравнения должны быть больше модулей остальных коэффициентов, не считая свободных членов.

3.4. Приближенное вычисление определенных интегралов

3.4.1. Формулы прямоугольников

Формула левых прямоугольников (рис. 3.11, *а*)

$$\int_a^b f(x)dx \approx h\sum_{k=0}^{n-1} y_k = h(y_0 + y_1 + y_2 + \ldots + y_{n-1});$$

формула правых прямоугольников (рис. 3.11, *б*)

$$\int_a^b f(x)dx \approx h\sum_{k=1}^{n} y_k = h(y_1 + y_2 + \ldots + y_n),$$

где $h = (b - a)/n$, $y_k = f(x_k)$, $x_k = a + kh$, $(k = 0, 1, 2, \ldots, n)$.

Абсолютная погрешность метода прямоугольников определяется неравенством

$$|R_n(f)| \le \frac{(b-a)^2 M}{2n}, \qquad \text{где } M = \max|f'(x)|.$$

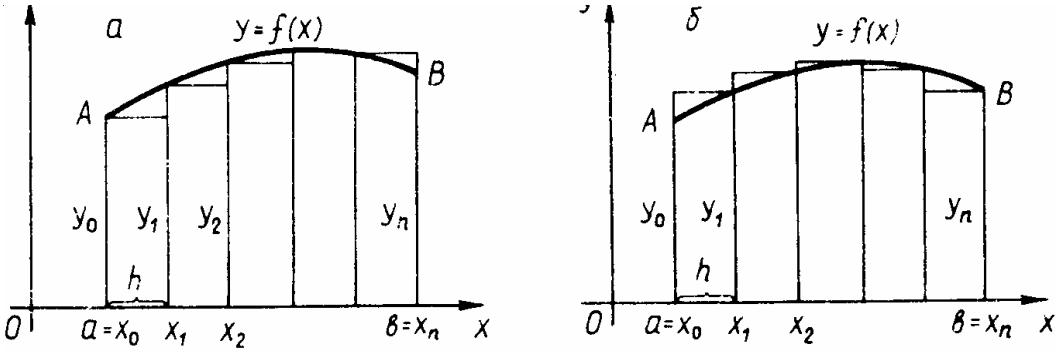


Рис. 3.11. Метод прямоугольников

3.4.2. Формула трапеций

$$\int_a^b f(x)dx \approx h(y_0/2 + y_1 + y_2 + \ldots + y_n/2),$$

где $h = (b-a)/n$, $y_k = f(x_k)$, $x_k = a + kh$, $(k = 0, 1, 2, \ldots, n)$.

Правая часть этой формулы выражает площадь фигуры, состоящей из трапеций, высота каждой из которых равна h (рис. 3.12). Остаточный член приближенной формулы трапеций

$$|R_n| \le \frac{(b-a)^3 M}{12n}, \qquad \text{где } M = \max|f''(x)|.$$

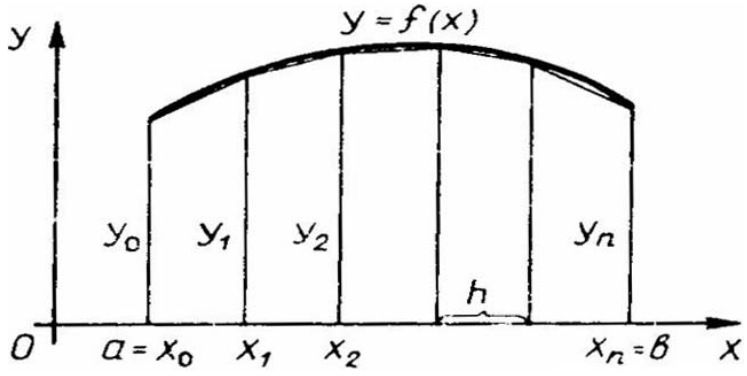


Рис. 3.12. Метод трапеций

3.4.3. Формула парабол

Формула парабол (или формула Симисона) имеет вид

$$\int_a^b f(x)dx \approx \frac{h}{3}(y_0 + 4(y_1 + y_3 + \ldots + y_{2n-1}) + 2(y_2 + y_4 + \ldots + y_{2n-2}) + y_{2n}),$$

где $h = (b-a)/(2n)$, $y_k = f(x_k)$, $x_k = a + kh$, $(k = 0, 1, 2, \ldots, 2n)$.

Правая часть формулы парабол выражает площадь фигуры, составленной из параболических трапеций $x_0M_0M_2x_2$, $x_2M_2M_4x_4$ (рис. 3.13).

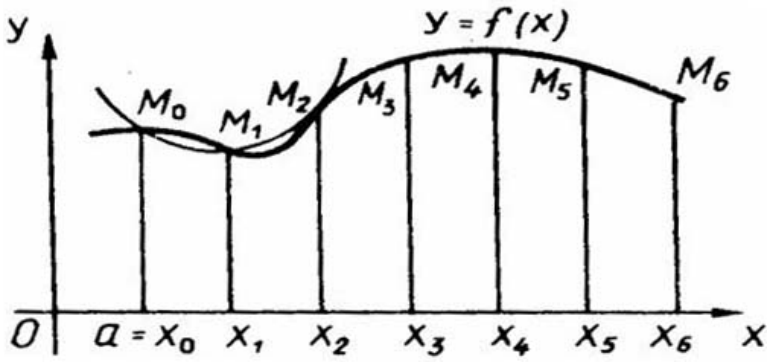


Рис. 3.13. Метод парабол

Дуга $M_0M_1M_2$ графика подынтегральной функции здесь заменена дугой параболы, проходящей через точки $M_0\ M_1\ M_2$. Аналогичная замена произведена и остальных дуг. Для остаточного члена формулы парабол выполняется неравенство

$$|R_n| \le \frac{(b-a)^5 M}{180(2n)^4}, \quad \text{где} \quad M = \max\left|f^{(IV)}(x)\right|.$$

Пример. По формуле парабол вычислить $\int_0^1 \frac{dx}{x^2+1}$, приняв $2n = 8$.

По первой из формул находим h. Составляем таблицу значений (табл. 3.1).

Таблица 3.1

k	h=(b-a)/	x_i	y_i	y_0, y_{2n}	y_i (нечётн.)	y_i (чётн.)
0	0,063	0	1,000	1,000		
1	0,063	0,063	0,996		0,996108949	0
2	0,063	0,125	0,985		0	0,984615385
3	0,063	0,188	0,966		0,966037736	0
4	0,063	0,25	0,941		0	0,941176471
5	0,063	0,313	0,911		0,911032028	0
6	0,063	0,375	0,877		0	0,876712329
7	0,063	0,438	0,839		0,839344262	0
8	0,063	0,5	0,800		0	0,8
9	0,063	0,563	0,760		0,759643917	0
10	0,063	0,625	0,719		0	0,719101124
11	0,063	0,688	0,679		0,679045093	0
12	0,063	0,75	0,640		0	0,64
13	0,063	0,813	0,602		0,602352941	0
14	0,063	0,875	0,566		0	0,566371681
15	0,063	0,938	0,532		0,532224532	0
16	0,063	1	0,500	0,500		
Сумма=				1,500	6,286	5,528
Интеграл =			**0,785**			

3.4.4. Приближенное вычисление определенных интегралов с помощью рядов

Если подынтегральная функция разлагается в степенной ряд, а пределы интегрирования принадлежат области сходимости этого ряда, то соответствующий определенный интеграл можно вычислить с заданной точностью.

Пример 2.6. Вычислить интеграл $\int_0^{1/4} \frac{\sin x}{x} dx$ с точностью до 0,00001.

Разложив подынтегральную функцию в ряд Маклорена (для sin x на x), получим

$$\frac{\sin x}{x} = 1 - \frac{x^2}{3!} + \frac{x^4}{5!} + \frac{x^6}{7!} + \ldots = (-1)^n \sum_{i=0}^{n} \frac{x^{2n}}{(2n+1)!}.$$

Интегрируя этот ряд почленно (это возможно, т. к. пределы интегрирования принадлежат интервалу сходимости данного ряда), получаем

$$\int_0^{1/4} \frac{\sin x}{x} dx = \int_0^{1/4} \left(1 - \frac{x^2}{3!} + \frac{x^4}{5!} + \frac{x^6}{7!} + \ldots\right) dx =$$

$$\left(x - \frac{x^3}{3 \cdot 3!} + \frac{x^5}{5 \cdot 5!} + \frac{x^7}{7 \cdot 7!} + \ldots\right)\Bigg|_0^{1/4} = (-1)^n \sum_{i=0}^{n} \frac{x^{2n+1}}{(2n+1)(2n+1)!}\Bigg|_0^{1/4} < 0{,}00001.$$

3.5. Приближенное решение дифференциальных уравнений

3.5.1. Интегрирование дифференциальных уравнений с помощью рядов

Решения многих дифференциальных уравнений не выражаются в элементарных функциях. В этих случаях пользуются приближенными методами интегрирования дифференциальных уравнений. Одним из таких методов является представление решения уравнения в виде степенного ряда; сумма конечного числа членов этого ряда будет приближенно равна искомому решению. Указанный степенной ряд находят способом неопределенных коэффициентов или способом, основанным на применении ряда Тейлора (или Маклорена).

Способ неопределенных коэффициентов особенно удобен в применении к линейным уравнениям и состоит в следующем. Если все коэффициенты уравнения и свободный член разлагаются в ряды по степеням, схо-

дящиеся в интервале ($a - h$, $a + h$), то искомое решение $y = y\{x\}$ также представляется степенным рядом сходящимся в том же интервале

$$y(x) = C_0 + C_1(x-a) + C_2(x-a)^2 + ... + C_n(x-a)^n.$$

Подставляя в уравнение функцию $y(x)$ и её производные, приравнивают коэффициенты при одинаковых степенях. Из полученных при этом уравнений и заданных начальных условий находят коэффициенты. Способ, основанный на применении ряда Тейлора – Маклорена, заключается в последовательном дифференцировании данного уравнения

$$y(x) = y(a) + y'(a)(x-a) + \frac{y''(a)}{2!}(x-a)^2 + ... + \frac{y^{(n)}(a)}{n!}(x-a)^n.$$

Это дает возможность найти значения производных, входящих в выражения для коэффициентов ряда, являющегося решением уравнения.

3.5.2. Метод Эйлера

Пусть требуется решить задачу Коши: найти решение дифференциального уравнения

$$y' = f(x,y),$$

удовлетворяющее начальному условию $y(x_0) = y_0$.

При численном решении этого уравнения задача ставится так: в точках x_0, x_1, x_2, ..., x_n найти приближения y_k (k = 0, 1, 2, ..., n) для значений точного решения $y(x_k)$. Разность $\Delta x_k = x_{k-1} - x_k$ называется шагом сетки. Во многих случаях величину Δx_k принимают постоянной h, тогда

$$x_k = x_{k-1} + kh \quad (k = 0, 1, 2, ..., n).$$

Метод Эйлера основан на непосредственной замене производной разностным отношением по приближенной формуле $\Delta y/\Delta x = f(x, y)$, где $\Delta y = y(x + h) - y(x)$, $\Delta x = (x+h) - x = h$.

Приближенное значение y_k в точке $x_k = x_0 + kh$ вычисляется по формуле

$$y_{k+1} = y_k + hf(x_k, y_k) \quad (k = 0, 1, 2, \ldots, n).$$

```
Program Eiler;
{РЕШЕНИЕ ДИФФЕРЕНЦИАЛЬНОГО УРАВНЕНИЯ Y'=2X-Y МЕТОДОМ ЭЙЛЕРА}
Const M=100;
Var
X, Y: array[0..M] of real;
```

```
X0, XN, H, FXY, A, B: real;
K, N: byte;

    Begin
    Write ('НАЧАЛО ОТРЕЗКА X0=');
    Readln (X[0]);
    Write ('КОНЕЦ ОТРЕЗКА XN=');
    readln (X[N]);
    Write ('ЧИСЛО РАЗБИЕНИЙ ОТРЕЗКА N=');
    readln (N);
    H:=(X[N]-X[0])/N;
    Write ('НАЧАЛЬНОЕ ЗНАЧЕНИЕ ФУНКЦИИ Y0=');
    readln (Y[0]);

    FOR K:=0 TO N-1 DO
    begin
    X[K]:=X[0]+K*H;
    FXY:=2*X[K]-Y[K];
    Y[K+1]:=Y[K]+H*FXY;
    End;
    Write ('РЕЗУЛЬТАТ=');
    Writeln (Y[N]:3:3);
END.
```

3.5.3. Метод Рунге – Кутта

Схема четвертого порядка точности:

$$y_{n+1} = y_n + \left(k_1 + 2k_2 + 2k_3 + k_4\right)/6,$$

$$k_1 = h_n\varphi\left(x_n, y_n\right),\ k_2 = h_n\varphi\left(x_n + h_n/2, y_n + k_1/2\right),$$

$$k_3 = h_n\varphi\left(x_n + h_n/2, y_n + k_2/2\right),\ k_4 = h_n\varphi\left(x_n + h_n, y_n + k_3\right).$$

В общем виде методы Рунге – Кутта можно описать следующим разностным уравнением:

$$\frac{y_{n+1} - y_n}{h} = \sum_{i=1}^{m} \sigma_i k_i ,$$

где $k_i = h\varphi\left(x_n + a_i \cdot h,\ y_n + \sum_{j=1}^{i-1} b_{ij} k_j \right),\ i = 2,3,...,m,\quad k_1 = h\varphi(x_n, y_n).$

Погрешность метода Рунге – Кутта $R_S \le \frac{x_l - x_0}{2880} \max(h_n^4) \max\left|\varphi^{(IV)}\right|$.

```
Program Runge_Kutt;
{РЕШЕНИЕ ДИФФЕРЕНЦИАЛЬНОГО УРАВНЕНИЯ Y'=2X-Y МЕТОДОМ РУНГЕ-КУТТА}
Const M=100;
Var
X, Y: array[0..M] of real;
H, DY, K1, K2, K3, K4: real;
I, N: byte;
        Begin
        Write ('НАЧАЛО ОТРЕЗКА X0=');
        Readln (X[0]);
        Write ('КОНЕЦ ОТРЕЗКА XN=');
        readln (X[N]);
        Write ('ЧИСЛО РАЗБИЕНИЙ ОТРЕЗКА N=');
        readln (N);
        H:=(X[N]-X[0])/N;
        Write ('НАЧАЛЬНОЕ ЗНАЧЕНИЕ ФУНКЦИИ Y0=');
        readln (Y[0]);
                FOR I:=0 TO N-1 DO
                begin
                X[I]:=X[0]+I*H;
                K1:=H*(2*X[I]-Y[I]);
                K2:=H*(2*(X[I]+H/2)-(Y[I]+K1/2));
                K3:=H*(2*(X[I]+H/2)-(Y[I]+K2/2))
                K4:=H*(2*(X[I]+H)-(Y[I]+K3));
                DY:=(K1+2*K2+2*K3+K4)/6
                Y[I+1]:=Y[I]+DY;
                End;
        Write ('РЕЗУЛЬТАТ=');
        Writeln (Y[N]:3:3);
END.
```

3.6. Интерполирование функций

В задачах математического моделирования очень часто возникает необходимость заменить используемую в расчетах функциональную зависимость $y(x)$ приближенной функцией $\varphi(x,\bar{a})$, по которой легко вычисляются значения исходной функции и которая в определенном смысле близка к $y(x)$. После такой замены все расчеты выполняют, используя за-

висимость $\varphi(x,\bar{a})$, причем близость $y(x)$ и $\varphi(x,\bar{a})$ обеспечивается подбором свободных параметров

$$\bar{a} = \{a_0, a_1, ..., a_n\}. \qquad (4.6.1)$$

Интерполяция может быть линейной или нелинейной, в соответствии с характером зависимости функции $\varphi(x,\bar{a})$ от параметров $\bar{a}$.

3.6.1. Линейная интерполяция

При линейной интерполяции функция $\varphi(x,\bar{a})$ имеет вид

$$\varphi(x,\bar{a}) = \sum_{j=0}^{n} a_j \varphi_j(x), \qquad (4.6.2)$$

где все функции $\varphi_j(x)$ линейно независимы.

Подставляя (4.6.2) в (4.6.1), получим систему линейных уравнений для определения a_i:

$$\sum_{j=0}^{n} a_j \varphi_j(x_i) = y_i, \ 0 \le i \le n. \qquad (4.6.3)$$

Единственность решения обеспечивается требованием неравенства нулю определителя системы (4.6.3):

$$\Delta = \begin{pmatrix} \varphi_0(x_0) & \varphi_1(x_0)... & \varphi_n(x_0) \\ . & ... & . \\ \varphi_0(x_n) & \varphi_1(x_n)... & \varphi_n(x_n) \end{pmatrix} \neq 0 \qquad (4.6.4)$$

при $x_k \neq x_l$ (т. е. при любых несовпадающих узлах).

Для практических вычислений удобно использовать многочлен $P_n(x)$ в форме интерполяционного полинома Лагранжа или Ньютона.

3.6.2. Интерполяционный многочлен Лагранжа

Интерполяционным многочленом Лагранжа называется многочлен вида

$$y = P_n(x) = \sum_{k=0}^{n} y_k \frac{(x-x_0)(x-x_1)...(x-x_{k-1})(x-x_{k+1})...(x-x_n)}{(x_k-x_0)(x_k-x_1)...(x_k-x_{k-1})(x_k-x_{k+1})...(x_k-x_n)} =$$

$$= \sum_{k=0}^{n} y_k \prod_{\substack{k=0 \\ j\neq k}}^{n} \frac{x-x_j}{x_k-x_j}.$$

Этот многочлен удовлетворяет условиям

$$P_n(x) = y_k\,,\ \ k = 0, 1, \ldots, n, \qquad (4.6.5)$$

где x_k – узлы (или полюсы) интерполяции, y_k – заданные числа.
Производя интерполирование функции $f(x)$ по формуле Лагранжа, заменяют эту функцию полиномом $P_n(x)$, совпадающим с ней в $n + 1$ данных точках отрезка $[a, b]$. В остальных точках этого отрезка разность $R_n(x) = f(x) – P_n(x)$ отлична от нуля и представляет собой погрешность метода. Термин «интерполяция» впервые употребил Д. Валлис (1656 г.) при составлении астрономических и математических таблиц.

```
Program Lagranzh;
{ЛИНЕЙНАЯ ИНТЕРПОЛЯЦИЯ МЕТОДОМ ЛАГРАНЖА}
Const M=100;
Var
X, Y, P: array[0..M] of real;
XX, S: real;
J, K, N: byte;
        Begin
        Write (' ЧИСЛО УЗЛОВ ИНТЕРПОЛЯЦИИ N=');
        readln (N);
        Write ('X=');
        Readln (XX);
        Write (' ВВЕДИТЕ ВХОДНЫЕ ПЕРЕМЕННЫЕ Xi И Yi ');
                FOR K:=0 TO N-1 DO
                Begin
                Writeln (K);
                Write (‘Xi=’);
                readln (X[K]);
                Write (‘Yi=’);
                readln (Y[K]);
                End;
        P[K]:=1;
                FOR K:=0 TO N DO;
                FOR J:=0 TO N DO;
                IF K<>J THEN P[K]= P[K]*(XX-X[J])/ (X[K]-X[J]);
                End;
                S=S+Y[K]*P[K];
                End;
        Write ('РЕЗУЛЬТАТ=');
        Writeln (S:3:3);
```

END.

3.6.3. Интерполяционный многочлен Ньютона

$$P_n(x) = P_n(x_0) + (x - x_0)P(x_0, x_1) + (x - x_0)(x - x_1)P(x_0, x_1, x_2) + ...$$

$$+(x - x_0)(x - x_1)...(x - x_{n-1})P(x_0, x_1, x_2, ..., x_n).$$

Таким образом, выражают многочлен n-ной степени через его значения в $(n+1)$ узлах $x_0, x_1, x_2, ..., x_n$. Ввиду того, что значения интерполяционного полинома в этих узлах совпадают со значениями интерполируемой функции, разделенные разности выражаются через узловые значения функции. В результате получается полином, называемый интерполяционным многочленом Ньютона:

$$y(x) \approx y(x_0) + \sum_{k=1}^{n} (x - x_0)(x - x_1)...(x - x_{k-1})y(x_0, ..., x_k).$$

При вычислениях по этой формуле точность расчетов удобно оценивать, наблюдая за тем, насколько быстро убывают члены ряда. Если это происходит достаточно быстро, можно оставлять только те члены, которые больше заданной погрешности расчетов. Безразличен порядок нумерации узлов, что очень удобно при подключении новых узлов для построения полинома более высокого порядка.

Погрешность многочлена Ньютона оценивают по формуле

$$\left(y(x) - P_n(x)\right(< \sqrt{\frac{2n}{\pi}} M_{n+1} \left(\frac{h}{2}\right)^{n+1},$$

где $M_{n+1} = \max(y^{(n+1)}(\xi))$ – максимальное значение производной интерполируемой функции на отрезке между наименьшим и наибольшим из значений $x_0, x_1, x_2, ..., x_n$, а $h = x_{i+1} - x_i$ (предполагается, что шаг постоянен).

Эта формула свидетельствует, что с уменьшением расстояния между узлами (шага) h погрешность представления функций полиномом Ньютона убывает.

Трудность использования формулы на практике состоит в том, что производные интерполируемой функции обычно неизвестны, поэтому для определения погрешности удобнее воспользоваться оценкой первого отброшенного члена.

Пример. Построить интерполяционный полином Ньютона четвертой степени для функции $y(x)=\cos(2x)$ в области значений аргумента $0 \le x \le \pi/4$.

Заполним таблицу разделенных разностей (табл. 3.2), вычисляемых по пяти узлам, представив для удобства вычислений

$$y(z)=\cos[(\pi/2)z],\ 0 \le z \le 1.$$

Таблица 3.2

z_0	$y(z_0)$	$y(z_0,z_1)$	$y(z_0,z_1,z_2)$	$y(z_0,...,z_3)$	$y(z_0,...,z_4)$
0	1				
		-0,304			
0,25	0,924		-1,128		
		-0,868		0,363	
0,5	0,707		-0,856		0,149
		-1,296		0,512	
0,75	0,383		-0,472		
		-1,532			
1	0				

Полином Ньютона

$$y(z) \approx 1-0,304z-1,128z(z-0,25)+0,363z(z-0,25)(z-0,5)+$$
$$+0,149z(z-0,25)(z-0,5)(z-0,75).$$

Вычислим

$$y(0,6) \approx 1-0,182-0,237+7,623\cdot 10^{-3}-4,694\cdot 10^{-3}=0,589.$$

Точное значение $y(0,6)=0,588$, т. е. точность вычислений по приближенной формуле оказалась весьма высокой.

В заключение отметим, что в практике вычислений для интерполяции полиномы степени выше пятой обычно не используют, т. е. число узлов интерполяции не превышает шести. Если при таком числе узлов не обеспечивается заданная погрешность, следует уменьшать расстояние между узлами.

3.6.4. Нелинейная интерполяция

Для табулирования быстроменяющихся функций требуется весьма малый шаг, т. е. возникает необходимость создавать таблицы очень больших объемов, что в ряде случаев неприемлемо. Оказывается, что преобразованием переменных $\eta=\eta(y)$ и $\xi=\xi(x)$ можно добиться того, чтобы в

новых переменных график $\eta(\xi)$ был близок к прямой хотя бы на отдельных участках. В этом случае интерполяцию проводят в переменных (η,ξ), а затем обратным интерполированием находят $y_i = y(\eta_i)$.

Преобразования $\eta(y)$ и $\xi(x)$ должны быть достаточно простыми (логарифмическая, экспоненциальная, тригонометрические и некоторые другие функции). При этом надо заботиться о том, чтобы и обратное преобразование $y(\eta)$ оказалось несложным.

Во многих задачах теплофизики, гидродинамики, оптики (особенно в задачах переноса излучения) и других областей науки и техники часто встречается степенная зависимость функции от своих аргументов. В этом случае удобны преобразования типа логарифмирования.

Пример. Получить формулу для нелинейной двухточечной интерполяции функции $y(x)$, если переменные можно преобразовать по формулам $\eta = \ln(y)$ и $\xi = x$.

Составим интерполяционный полином Ньютона на двухточечном шаблоне:

$$\eta = \eta_0 + \frac{\eta_1 - \eta_0}{\xi_1 - \xi_0}(\xi - \xi_0).$$

В исходных переменных имеем

$$\ln(y) = \ln(y_0) + \frac{\ln(y_1) - \ln(y_0)}{x_1 - x_0}(x - x_0),$$

и окончательно $y = y_0(y_1 / y_0)^{(x-x_0)/(x_1-x_0)}$.

3.6.5. Интерполяция сплайнами

Слово «сплайн» переводится как «гибкая линейка». Такую «линейку» можно использовать для проведения кривых через заданную совокупность точек, изгибая и придерживая ее так, чтобы ребро проходило через все точки на плоскости. Равновесие гибкой линейки описывается уравнением $\psi''(x) = 0$, т. е. интерполяционный полином на участке между каждой парой соседних точек имеет третью степень:

$$\psi(x) = a_i + b_i(x - x_{i-1}) + c_i(x - x_{i-1})^2 + d_i(x - x_{i-1})^3,$$

$$x_{i-1} \le x \le x_i,\ 0 \le i \le N.$$

В узлах значения многочлена и интерполируемой функции совпадают:

$$\psi(x_{i-1}) = y_{i-1} = a_i,$$

$$\psi(x_i) = y_i = a_i + b_i h_i + c_i h_i^2 + d_i h_i^3,$$

$$h_i = x_i - x_{i-1}, 1 \le i \le N .$$

Число таких уравнений меньше числа неизвестных в два раза. Недостающие уравнения получают, приравнивая во внутренних узлах первые и вторые производные, вычисляемые по коэффициентам на соседних участках:

$$\psi'(x) = b_i + 2c_i(x - x_{i-1}) + 3d_i(x - x_{i-1})^2,$$

$$\psi''(x) = 2c_i + 6d_i(x - x_{i-1}), \quad x_{i-1} \le x \le x_i,$$

$$b_{i+1} = b_i + 2c_i h_i + 3d_i h_i^2,$$

$$c_{i+1} = c_i + 3d_i h_i, \ 1 \le i \le N - 1.$$

Недостающие условия можно получить, полагая, например, что вторая производная равна нулю на концах участка интерполирования:

$$\psi''(x_0) = 0, c_1 = 0,$$

$$\psi''(x_N) = 0, c_N + 3d_N h_N = 0,$$

Уравнения позволяют определить все $4N$ неизвестных коэффициентов: $a_i, b_i, c_i, d_i (1 \le i \le N)$.

3.6.6. Многомерная интерполяция

В различных приложениях широко используют двумерные и трехмерные таблицы. Например, теплофизические свойства различных веществ зависят от температуры и давления, а оптические характеристики – еще и от длины волны излучения.

При многомерной интерполяции из-за громоздкости таблиц необходимо брать достаточно большие шаги по аргументам, т.е. сетка узлов, на которой строят таблицу, получается довольно грубой. Поэтому требуется вводить преобразование переменных $\eta = \eta(y)$, $\xi = \xi(x)$, $\varphi = \varphi(z)$, подбирая подходящие формулы. При удачном выборе таких формул можно использовать в новых переменных интерполяционный полином невысокой степени.

Осуществляя многомерную интерполяцию, следует помнить, что расположение узлов не может быть произвольным. Например, при интерполяции полиномом первой степени $P_1(x, y)$ узлы не должны лежать на одной прямой в плоскости. Действительно, определитель системы трех уравнений

$$z_i = a + bx_i + cy_i,\ 1 \le i \le 3$$

записывается в виде

$$\Delta = x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2).$$

Условие размещения трех точек на одной прямой выглядит следующим образом:

$$\frac{x_2 - x_1}{y_2 - y_1} = \frac{x_3 - x_2}{y_3 - y_2}.$$

После простых преобразований имеем

$$x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2) = 0,$$

т.е. если узлы лежат на одной прямой, то определитель Δ обращается в нуль и построить полином $P_1(x, y)$ вида z_i невозможно.

Проверять условия подобного типа достаточно сложно, поэтому на практике стремятся строить регулярные сетки, как правило, прямоугольные и равномерные, когда узлы являются точками пересечения двух взаимно перпендикулярных систем параллельных прямых. На этой сетке проводят простую последовательную интерполяцию: сначала по строкам, а затем по столбцам.

При последовательной интерполяции завышается степень интерполяционного полинома. При треугольной конфигурации расположения узлов степень многочлена будет минимальной. Многочлен n-ной степени в форме Ньютона в этом случае можно представить как обобщение одномерного варианта записи:

$$P_n(x, y) = \sum_{i=0}^{n} \sum_{j=0}^{n-1} z(x_0, ..., x_i, y_0, ..., y_j) \prod_{p=0}^{i-1} (x - x_p) \prod_{q=0}^{j-1} (y - y_q).$$

3.7. Регрессия

Термин «регрессия» ввел в употребление Ф. Гальтон – один из создателей математической статистики. Сопоставляя рост детей и их родителей, он обнаружил, что соответствие между ростом отцов и детей выражено слабо. Оно оказалось меньшим, чем он ожидал. Гальтон объяснил это явление наследственностью не только от родителей, но и от более отдаленных предков. По его предположениям, рост определяется наполовину родителями, на четверть – дедушкой и бабушкой, на одну восьмую – пра-

дедом и прабабкой и т. д. Неважно, насколько был прав Гальтон. Главное в том, что он обратил внимание на движение назад по генеалогическому дереву и назвал его регрессией, позаимствовав понятие движения назад, противоположное прогрессу – движению вперед.

Впоследствии слово «регрессия» заняло в статистике заметное место, хотя, как это часто бывает в любом языке, в том числе и в языке науки, в него теперь вкладывают другой смысл – оно означает статистическую связь между случайными величинами.

3.7.1. Линейная регрессия

При обработке экспериментальных данных химики часто строят различные графики, пользуясь декартовой (прямоугольной) системой координат. Иногда зависимости между измеряемыми величинами x и y носят линейный характер, поэтому экспериментальные точки группируются около некоторой прямой линии (рис. 3.14).

Уравнение прямой линии имеет вид

$$y = a + bx.$$

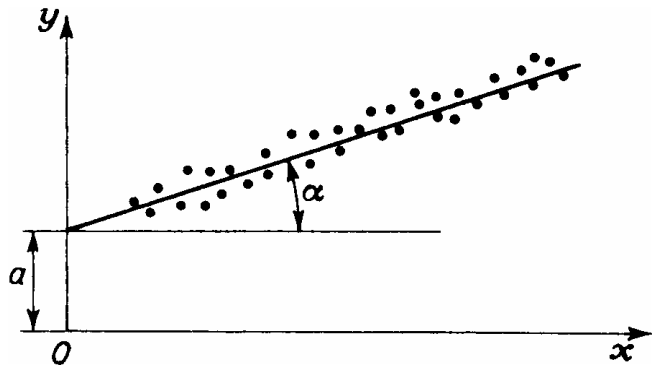


Рис. 3.14. Зависимость между измеряемыми величинами

Здесь a – длина отрезка от начала координат до точки пересечения прямой с осью Oy, b – тангенс угла наклона а. прямой к оси Ox ($b = \operatorname{tg}\alpha$). Прямую линию стараются провести так, чтобы сумма квадратов отклонений расчетных значений y_p от экспериментальных значений $y_э$ была минимальной для всех n рассматриваемых опытов:

$$s = \sum_{i=1}^{n} (y_p - y_э)^2 \to \min.$$

Другими словами, n – это общее число опытов, в которых измерялись значения величин x и y, s_2 характеризует сумму всех измеренных зна-

чений x, а s_3 – сумму всех значений величины y. Остальные обозначения понятны из формул.

Уравнение $y = a + bx$, в которое подставлены найденные значения коэффициентов, принято называть *уравнением линейной регрессии*. Пользуясь этим уравнением, можно, не проводя дальнейших опытов, рассчитать для заданного x соответствующее значение y. Таким образом, с помощью уравнения регрессии можно прогнозировать величину y. Разумеется, такой прогноз не будет абсолютно точным. Близость прогнозируемого значения y к фактическому значению зависит в основном от точности, с которой выполняется эксперимент, и от того, насколько существующая зависимость между y и x близка в действительности к линейной.

Расчеты коэффициентов линейной регрессии всегда выполняются по одним и тем же формулам и требуют при ручном счете значительных затрат времени. Чтобы избавить исследователей от этой рутинной работы, применяют ПК.

По программе вычисляются не только коэффициенты уравнения регрессии, но и среднее квадратичное отклонение расчетных значений y от экспериментальных. Если сумму квадратов отклонений обозначить через s, то среднее квадратичное отклонение σ вычисляется по формуле $\sigma = \sqrt{\frac{s}{n-1}}$.

Рассмотрим применение программы для выполнения практических расчетов.

Известно, что с повышением температуры растворимость солей в воде увеличивается. Напомним, что растворимостью называется концентрация соли в насыщенном растворе.

Экспериментальные данные по растворимости соли при различных температурах приведены в табл. 3.3.

Таблица 3.3

Температура (x), °С	10	20	30	40	50	60	70	80
Растворимость (y), г/л	42	50	62	67	75	86	92	103

Обработав экспериментальные данные с помощью компьютера, получаем уравнение регрессии $y = 33{,}71 + 0{,}85x$ и среднее квадратичное отклонение 1,46. Программа оказалась достаточно простой и удобной для использования, но, к сожалению, область ее применения ограничена только линейными зависимостями. Однако нужно иметь в виду, что некоторые нелинейные зависимости можно преобразовать в линейные.

Например, гиперболическая связь $y = a_0 + a_1/x$ линеаризуется заме-

ной переменной $z = 1/x$, тогда $y = a_0 + a_1 z$.

Показательная связь $y = a \cdot e^{a_1 x}$ линеаризуется путем логарифмирования: $v = \ln y = \ln a_0 + a_1 x$.

Степенная связь после логарифмирования линеаризуется заменой $\ln y = p$, $\ln a_0 = c$, $\ln x = z$, тогда $p = c + a_1 z$.

Логарифмическая связь $y = a_0 + a_1 \ln x$ линеаризуется заменой $z = \ln x$, тогда $y = a_0 + a_1 z$.

Комбинированная связь $y = 1/(a_0 + a_1 e^{-x})$ линеаризуется заменой $v = 1/y$ и $z = e^{-x}$, тогда $v = a_0 + a_1 z$.

Широкое использование линейных зависимостей и связей, легко приводимых к линейным, объясняется следующим. Линейные связи просты и требуют относительно малого объема вычислений, а методика их установления более глубоко разработана.

Рассмотрим пример линеаризации. Как известно, зависимость константы скорости химической реакции от температуры имеет экспоненциальный характер

$$k = k_0 e^{-\frac{E}{RT}}.$$

Прологарифмировав это выражение, получим

$$\ln k = \ln k_0 - \frac{E}{R}\frac{1}{T}.$$

Введем следующие обозначения:

$$y = \ln k; \quad a = \ln k_0; \quad b = -\frac{E}{R}; \quad x = \frac{1}{T}.$$

С учетом этих обозначений получаем линеаризованное уравнение регрессии $y = a + bx$. Теперь снова можно воспользоваться нашей программой. Вычислив коэффициенты a и b, можно рассчитать предэкспоненциальный множитель k_0 и энергию активации E.

Таким образом, благодаря линеаризации область применения программы значительно расширяется.

3.7.2. Корреляция

Желая установить связь между двумя величинами, исследователь наносит их значения на график. Каждой паре значений величин x и y на графике соответствует точка. Обычно такие точки не ложатся на одну линию, а занимают на плоскости чертежа некоторую область, образуя так называемую диаграмму рассеивания (рис. 3.15). Причины рассеивания могут

быть самыми различными. Среди них можно назвать погрешности измерений, влияние неучтенных факторов и т. д. Даже если один и тот же опыт в химической лаборатории проводится несколько раз, результаты измерений обычно отличаются друг от друга.

Связь между двумя величинами, характеризующими химико-технологический процесс, можно выразить уравнением регрессии. Однако оно не позволяет с абсолютной точностью предсказать значение y, если известно значение x. Прогнозирование y осуществляется только с некоторой вероятностью. Такая связь между двумя величинами называется корреляционной. Она существенно отличается от связи детерминистической, описываемой строгими физико-химическими формулами.

Благодаря рассеянию, точки на графике располагаются в некоторой конечной области, обычно имеющей форму эллипса. Чем теснее связь между рассматриваемыми величинами, тем более вытянут эллипс вдоль одной из своих осей (см. рис. 3.15, *а* и *б*). Наоборот, если связь между x и y отсутствует, то разброс точек на диаграмме рассеивания по форме приближается к кругу (см. рис. 3.15, *в*).

Корреляционную связь между величинами называют положительной, если при увеличении одной из них возрастает другая величина. Отрицательной называют такую корреляционную связь, когда при увеличении значений одной из величин другая уменьшается.

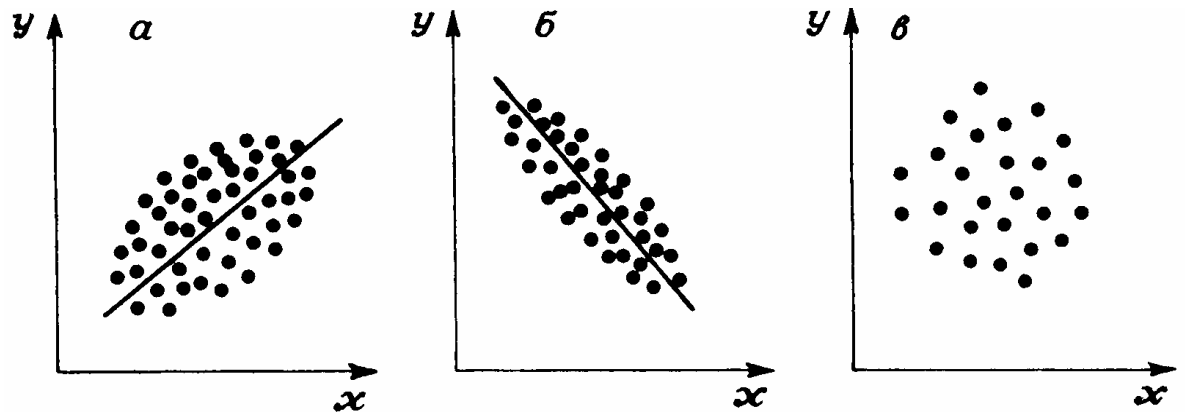


Рис. 3.15. Диаграммы рассеивания

Видя на графике корреляционную связь между величинами, мы можем дать ей качественную оценку. Для количественной оценки линейной корреляции используется коэффициент парной корреляции r_{xy}, который рассчитывается по формуле

$$r_{xy} = \frac{s_1 s_5 - s_2 s_3}{\sqrt{(s_1 s_4 - s_2^2)(s_1 s_6 - s_3^2)}};$$

где $s_1 = n$; $s_2 = \sum_{i=1}^{n} x_i$; $s_3 = \sum_{i=1}^{n} y_i$; $s_4 = \sum_{i=1}^{n} x_i^2$; $s_5 = \sum_{i=1}^{n} x_i y_i$; $s_6 = \sum_{i=1}^{n} y_i^2$.

Этот коэффициент может принимать следующие значения:

1) $r_{xy} = 0$, что свидетельствует об отсутствии корреляционной связи между x и y;

2) $r_{xy} = 1$, в данном случае существует строгая положительная детерминистическая связь;

3) $r_{xy} = -1$; между x и y существует строгая отрицательная детерминистическая связь;

4) $-1 < r_{xy} < +1$, это наиболее распространенный случай: корреляционная связь может быть как положительной, так и отрицательной и характеризоваться различной степенью тесноты связи. Расчеты коэффициентов парной корреляции полезно выполнять на ПК.

Чем ближе абсолютное значение коэффициента корреляции $|r_{xy}|$ к единице, тем сильнее линейная связь между x и y. Следует отметить, что r_{xy} одновременно отражает степень случайности и криволинейности связи между величинами x и y. Например, зависимость y от x может быть близкой к функциональной, но существенно нелинейной. В этом случае коэффициент корреляции будет значительно меньше единицы.

Таким образом, тесноту связи между двумя величинами можно объективно оценить только после рассмотрения графика «корреляционного поля» и вычисления коэффициента парной корреляции.

4. ТАБЛИЧНЫЙ ПРОЦЕССОР MICROSOFT EXCEL

Microsoft Excel является мощным программным средством для работы с таблицами данных. С его помощью можно выполнять:

– автоматическую обработку зависящих друг от друга данных;

– автоматизацию итоговых вычислений и создание сводных таблиц;

– ведение простых баз данных и бработку записей;

– совместную работу с экономическими документами и подготовку табличных документов;

– построение диаграмм и графиков по имеющимся данным.

Основное рабочее пространство Microsoft Excel – это *рабочая книга*, представляющая собой набор *рабочих листов* (рис. 2.1).

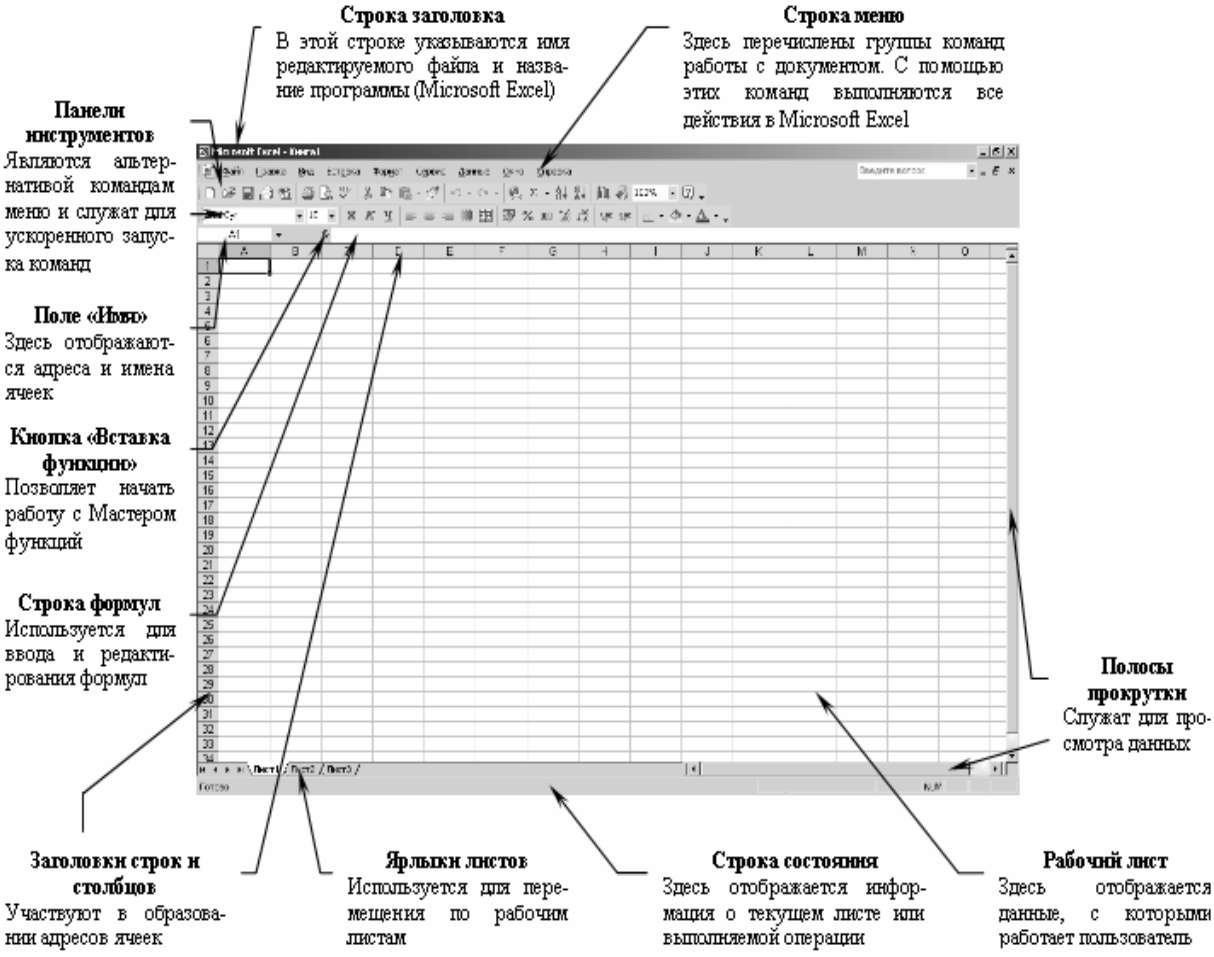


Рис. 2.1. Элементы окна Microsoft Excel

Рабочие листы имеют табличную структуру и могут содержать одну или несколько таблиц. Каждый рабочий лист имеет *название*, которое отображается на ярлычке листа, находящемся в его нижней части. Помимо

рабочих листов книга Microsoft Excel может содержать листы *диаграмм* и *модулей* Visual Basic for Applications.

Табличное пространство рабочего листа состоит из *строк* и *столбцов*. Столбцы озаглавлены латинскими буквами или двухбуквенными комбинациями (заголовки столбцов), строки последовательно нумеруются числами. Всего рабочий лист содержит 256 столбцов (от A до IV) и 65536 строк.

Минимальными элементами хранения данных являются *ячейки* таблицы. Обозначение отдельной ячейки сочетает в себе номера столбца и строки, пересечением которых она образована. Это сочетание называется *адресом* или *ссылкой* на ячейку (A1, DE234). Одна из ячеек всегда является активной и выделяется рамкой активной ячейки (*табличный курсор*). Ячейку можно активизировать (сделать текущей) с помощью мыши или клавиш управления курсором. Адрес текущей ячейки всегда отображается в *поле имени* с левой стороны *Строки формул* над рабочим листом. Операции ввода и редактирования всегда производятся в активной ячейке.

На данные, расположенные в соседних ячейках, можно ссылаться в формулах, как на единое целое. Такую группу ячеек называют *диапазоном*. Диапазон ячеек обозначают, указывая через двоеточие номера ячеек, расположенных в левом верхнем и правом нижнем углах диапазона (A1:C15).

4.1. Ввод и редактирование данных

4.1.1. Перемещение по рабочему листу

Выполняется с помощью мыши (однократный щелчок левой кнопкой на нужной ячейке) или клавиш управления курсором (табл. 2.1):

Таблица 2.1

Сочетания клавиш для перемещения по рабочему листу

Чтобы активизировать ячейку	Нажать
Слева	← или Shift + Tab
Справа	→ или Tab
Сверху, в предыдущей строке	↑ или Shift Enter
Снизу, в следующей строке	↓ или Enter
Выше на одну экранную страницу	Page Up
Ниже на одну экранную страницу	Page Down
У правой границы текущего диапазона	Ctrl + →
У левой границы текущего диапазона	Ctrl + ←
В начале рабочего листа	Ctrl + Home
На пересечении последних строки и столбца, которые содержат данные	Ctrl + End

При помощи полос прокрутки можно быстро просмотреть части рабочего листа, которые не поместились на экране, но в этом случае меняется вид рабочего листа, а текущей остается прежняя ячейка.

4.1.2. Выделение группы ячеек

Иногда при работе с данными возникает необходимость в выделении группы ячеек. Для этого MS Excel предоставляет следующие возможности.

Чтобы выделить несколько смежных ячеек, нужно щелкнуть на первой из них, затем нажать клавишу `Shift` и, не отпуская ее, щелкнуть на последней из выделяемых ячеек.

Можно выделить группу смежных ячеек, пользуясь только мышью. Для этого нужно нажать левую кнопку мыши на первой ячейке и, не отпуская ее, перетащить указатель мыши в последнюю ячейку.

Выделение группы смежных ячеек называется выделением *блока* или *диапазона ячеек*.

Для выделения несмежных ячеек, нужно щелкнуть на первой из них, а затем нажать клавишу `Ctrl` и, не отпуская ее, щелкать на других ячейках, которые необходимо выделить.

Чтобы выделить строку или столбец, нужно щелкнуть по заголовку соответствующей строки или столбца.

Чтобы выделить весь рабочий лист, следует щелкнуть на пересечении строки заголовков столбцов и столбца заголовков строк.

4.1.3. Ввод и редактирование данных

В ячейки рабочего листа можно вводить данные следующих типов: *текст*, *числа* или *формулы*. Для этого нужно выделить (активизировать) ячейку и набрать с клавиатуры данные. В процессе ввода Microsoft Excel автоматически распознает тип данных.

Все, что набирается с клавиатуры, появляется и в текущей ячейке, и в строке формул. Занести набранные данные в текущую ячейку можно, нажав клавишу `Enter`, или щелкнув мышью на другой ячейке. Отменить ввод данных можно нажатием клавиши `Esc`.

Редактировать данные в ячейках можно двумя способами:

1) можно выделить ячейку и, щелкнув в строке формул, отредактировать данные в строке формул;

2) редактировать данные прямо в ячейке. Для редактирования в ячейке нужно выполнить на ней двойной щелчок левой кнопкой мыши, а затем щелчком мыши поместить текстовый курсор на место, куда требуется внести изменения.

Чтобы удалить данные из ячейки, ее нужно выделить и нажать клавишу `Del`.

4.1.4. Ввод последовательностей данных в ячейки:

1) выделить первую из заполняемых ячеек;

2) ввести начальное значение для ряда значений;

3) ввести следующее значение в соседнюю ячейку, чтобы определить образец заполнения;

4) выделить ячейку или ячейки, содержащие начальные значения;

5) перетащить маркер заполнения (рис. 2.2) при нажатой левой кнопке мыши через заполняемые ячейки.

Рис. 2.2. Маркер заполнения

Для заполнения в возрастающем порядке перетащить маркер вниз или вправо. Для заполнения в убывающем порядке перетащить маркер вверх или влево.

Пример 2.1. Если требуется получить ряд 2, 3, 4, 5…, введите 2 и 3 в первые две ячейки. Если требуется получить ряд 2, 4, 6, 8…, введите 2 и 4. Если требуется получить ряд 2, 2, 2, 2…, вторую ячейку можно оставить пустой. Чтобы задать тип ряда значений, перетащите маркер заполнения правой кнопкой мыши, а затем выберите соответствующую команду в контекстном меню.

Пример 2.2. Если начальное значение – дата «янв-2002», то для получения ряда «фев-2002», «мар-2002» и т. д. выберите команду *Заполнить по месяцам*, а для получения ряда «янв-2003», «янв-2004» и т. д. выберите команду *Заполнить по годам*.

Чтобы точно сформулировать условия заполнения ячеек, следует выполнить команду: меню *Правка→Заполнить→Прогрессия* – откроется окно диалога *Прогрессия* (рис. 2.3). Здесь выбрать тип прогрессии, величину шага и предельное значение. После щелчка на кнопке OK Microsoft Excel автоматически заполнит ячейки в соответствии с заданными правилами.

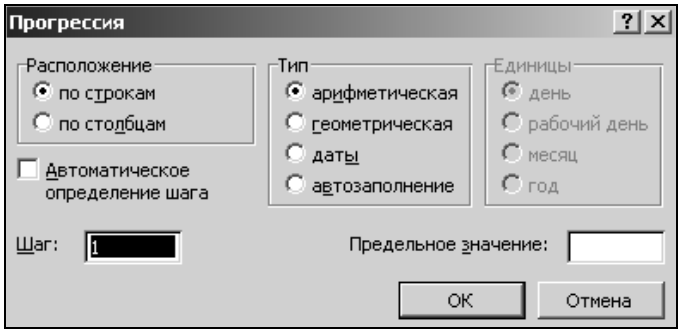


Рис. 2.3. Окно диалога *Прогрессия*

4.1.5. Настройка Автозаполнения:

1) выполнить команду меню *Сервис→Параметры* – откроется окно диалога *Параметры*, в этом окне выбрать вкладку *Списки* (рис. 2.4);

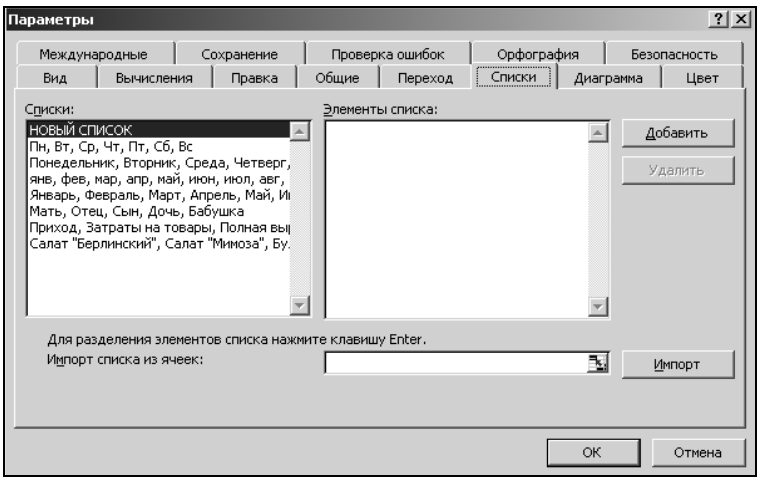


Рис. 2.4. Окно диалога *Параметры*, вкладка *Списки*

2) чтобы ввести новый список, выбрать *Новый список* из списка *Списки*;

3) ввести данные в поле *Элементы списка*, начиная с первого элемента. После ввода каждой записи нажимать клавишу `Enter`;

4) После того, как список будет введен полностью, нажать кнопку `Добавить`.

4.1.6. Переупорядочивание содержимого ячеек

Введенные данные, будь то текст, числа или формулы, не привязаны жестко к конкретным ячейкам. Данные можно копировать из одних ячеек в другие, добавлять и удалять ячейки, строки, столбцы и перемещать данные между ячейками.

Копирование и вставка данных в ячейки. Копировать данные в другое место рабочего листа можно: с помощью кнопок копирования и вставки на панели инструментов; с помощью соответствующих команд меню *Правка*; перетаскивая данные мышью.

Копирование с помощью мыши – самый быстрый способ, особенно если ячейки находятся на одном рабочем листе и расстояние между ними невелико.

Процедура:

1) выделить нужную ячейку или диапазон ячеек;

2) поместить указатель мыши на любом участке границы выделенного диапазона;

3) нажать и не отпускать клавишу `Ctrl`. При нажатии на клавишу `Ctrl`, если указатель мыши находится на границе диапазона, он помечает-

ся маленьким плюсом. Этот плюс указывает, что содержимое диапазона подлежит копированию, а не перемещению;

4) перетащить рамку диапазона на новое место, отпустить кнопку мыши и затем клавишу Ctrl. Копия выделенной информации будет вставлена в новый блок.

Копирование и вставка элементов ячеек. Копировать и вставлять можно не только ячейку целиком, но и отдельные элементы ее содержимого. Например, если в ячейке записана некоторая формула, то в новую ячейку можно перенести только результат вычислений.

Для того чтобы выборочно скопировать и вставить отдельные элементы ячейки, выполняют следующие действия:

1) выделить ячейку или диапазон ячеек;
2) скопировать выделенное в буфер обмена;
3) определить место вставки;
4) щелкнуть правой кнопкой мыши. В появившемся контекстном меню выбрать команду Специальная вставка. Откроется окно диалога Специальная вставка (рис. 2.5).
5) в этом окне определить параметры вставки данных путем выбора соответствующей опции;
6) щелкнуть на кнопке OK.

Перемещение данных между ячейками. Переместить данные в другое место рабочего листа можно: с помощью кнопок *Вырезать* и *Вставить* на панели инструментов; с помощью соответствующих команд меню *Правка*; перетаскивая данные мышью.

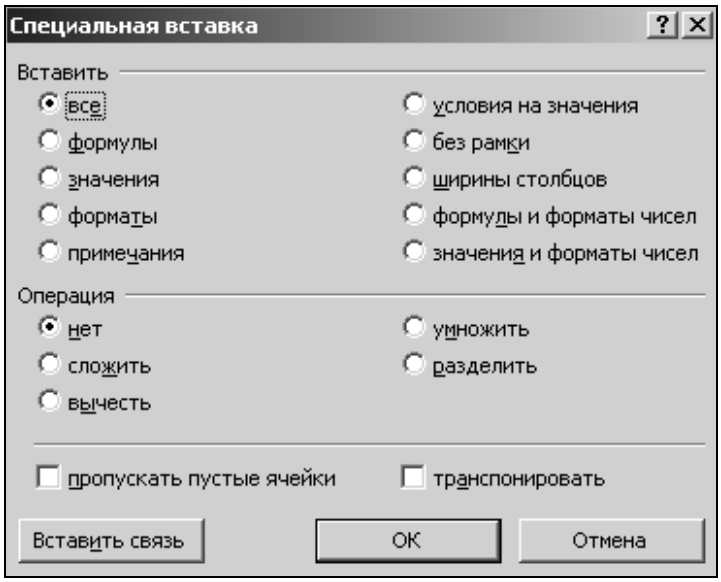


Рис. 2.5. Окно диалога *Специальная вставка*

Чтобы переместить содержимое ячейки с помощью мыши, достаточно выделить ячейку перетащить ее рамку на новое место.

Вставка и удаление ячеек, строк и столбцов. При оформлении рабочих листов можно копировать ячейки в любое место рабочей книги. Если не хватает пространства для записи данных в конкретную область рабочей таблицы, можно добавлять или удалять ячейки, строки и столбцы. Когда происходит удаление ячеек рабочего листа, уничтожается их содержимое, а другие ячейки из правой или из нижней областей таблицы передвигаются на место удаленных. Вставить ячейку, строку или столбец можно с помощью команд *Ячейки*, *Строки*, *Столбцы* из меню *Вставка*, а удалить – с помощью команды *Удалить* из меню *Правка*.

Вставляя или удаляя строку или столбец, следует выделять их целиком, а не какую-то отдельную их часть. Выделить строку или столбец целиком можно, щелкнув на их заголовках.

Вставляя ячейку, строку или столбец, необходимо выделить ячейку, строку или столбец, расположенные непосредственно справа или внизу от позиции вставки. При вставке с помощью команд меню следует определить, какие из старых ячеек передвинуть, чтобы освободить пространство для новой, а в случае удаления – какие ячейки займут место удаленных. Вставлять, удалять и перемещать столбцы и строки можно так же, как и отдельные ячейки.

4.2. Работа с книгами Microsoft Excel

4.2.1. Перемещение по рабочей книге

Выбирать различные рабочие листы в рабочей книге можно следующими способами:

1) щелчком мыши на соответствующем ярлычке в нижней части рабочих листов;

2) с помощью клавиатуры: одновременное нажатие клавиш `Ctrl+Page Down` позволяет перейти на следующий лист, `Ctrl+Page Up` – на предыдущий.

Используя кнопки прокрутки ярлычков в левом нижнем углу экрана  , можно переместиться к ярлычку первого листа рабочей книги, перейти к ярлычку предыдущего или последующего листов, переместиться к ярлычку последнего листа (если в книге много листов).

4.2.2. Выделение рабочих листов

Можно выделить сразу несколько рабочих листов. Выделение выполняется аналогично выделению группы ячеек: выделение смежных листов выполняется щелчком мыши по первому и последнему листам при на-

жатой клавише Shift, а несмежных – щелчками мыши по нужным листам при нажатой клавише Ctrl.

Выделив несколько рабочих листов, можно вводить одни и те же данные одновременно в каждый из них. Данные появятся в соответствующих ячейках каждого из выделенных листов.

4.2.3. Манипулирование рабочими листами

Каждая новая рабочая книга состоит из трех рабочих листов с именами *Лист1*, *Лист2* и *Лист3*. Исходный состав рабочей книги можно изменить, добавляя, удаляя или переименовывая ее листы.

Вставка листов. Для вставки новых рабочих листов следует выполнить команду меню *Вставка→Лист* (добавится столько листов, сколько их было предварительно выделено).

Удаление листов. Для удаления рабочих листов следует выполнить команду меню *Правка→Удалить лист* (удалится столько листов, сколько их было предварительно выделено).

Переименование листов. Для переименования рабочего листа его нужно выделить, затем выполнить команду меню *Формат→Лист→Переименовать*, ввести новое имя поверх старого и нажать клавишу Enter или щелкнуть мышью вне ярлычка листа.

Перемещение и копирование листов внутри рабочей книги. Для выполнения этих операций можно использовать следующие действия:

1) перемещение выполняется обычным перетаскиванием ярлычка соответствующего листа на новое место;

2) копирование выполняется перетаскиванием ярлычка при нажатой клавише Ctrl.

4.3. Форматирование данных

4.3.1. Автоматическое форматирование данных

Этот вид форматирования позволяет быстро придать диапазону ячеек эстетичный вид. Microsoft Excel предлагает различные варианты форматирования, включая форматы для финансовых данных, учетных сведений, списков, а также цветные и трехмерные форматы.

Процедура:

1) выделить диапазон, который требуется отформатировать;

2) выполнить команду меню *Формат→Автоформат* – откроется окно диалога *Автоформат* (рис. 2.6).

3) в списке форматов выбрать наиболее подходящий. По умолчанию форматирование производится по всем элементам формата. Для частично-

го применения *Автоформата* нажать кнопку Параметры и снять флажки для элементов форматирования, которые не нужно применять.

Рис. 2.6. Окно диалога *Автоформат*

Для удаления примененного *Автоформата* следует выполнить пункты 1) и 2), а затем выбрать формат с подписью *Нет*.

4.3.2. Копирование форматов в другие ячейки

Применить к данным формат, уже использованный в какой-то части рабочего листа, можно с помощью кнопки *Формат по образцу* .

Процедура:

1) выделить ячейку с нужным форматом;
2) щелкнуть на кнопке *Формат по образцу*;
3) указать область применения копируемого формата.

4.3.3. Форматирование с помощью панели инструментов *Форматирование*

Изменение формата чисел. Быстро задать стандартный формат для числа в ячейке можно при помощи следующих кнопок:

1) *Денежный формат* ;
2) *Процентный формат* ;
3) *Формат с разделителями* .

Каждый из этих форматов по умолчанию оставляет определенное количество десятичных знаков, которое может быть изменено кнопками *Уменьшить разрядность* и *Увеличить разрядность* .

Изменение шрифта. Шрифты и их размер можно быстро менять, используя поля *Шрифт* Arial Cyr и *Размер шрифта* 10 на панели инструментов *Форматирование*. Каждый шрифт имеет три модификации:

Полужирный, *Курсив* и *Подчеркнутый*, которые задаются соответственно кнопками  ,  ,  .

Изменение выравнивания. По умолчанию Microsoft Excel автоматически выравнивает вводимый текст по левому, а числовые значения – по правому краю. Для изменения способа выравнивания данных в ячейках можно воспользоваться соответствующими кнопками на панели инструментов *Форматирование*:

1) *По левому краю* –  ;
2) *По центру* –  ;
3) *По правому краю* –  .

Примечание. Если нужно, например, выровнять заголовок по центру нескольких столбцов, выделяются ячейки, относительно которых надо центрировать заголовок, и выполняется щелчок по кнопке выравнивания *Объединить и поместить в центре*  .

Добавление рамок и изменение цветов. Отдельные ячейки или области листа можно оформить, выделив их рамками или выбрав различные цвета текста. Рамки могут выделять ячейки дополнительными линиями сверху, снизу, по бокам, а также вокруг ячеек. Ячейку можно закрасить подходящим цветом:

1) цвет текста можно изменить с помощью кнопки *Цвет шрифта*  ;
2) границы ячеек можно установить с помощью кнопки *Границы*  .;
3) закрасить ячейку можно с помощью кнопки *Цвет заливки*  .

Процедура:

1) выделяется ячейка или диапазон ячеек;
2) выполняется щелчок по стрелке нужной кнопки;
3) в соответствующих палитрах щелчком мыши выбирается цвет или рамка.

Примечание. Изменять формат данных в ячейках можно также с помощью меню *Формат→Ячейки→*окно *Формат ячеек*. Вкладки этого диалогового окна предоставляют больше возможностей, чем кнопки на панели инструментов *Форматирование*. Здесь можно выбирать формат записи данных (количество знаков после запятой, указание денежной единицы, способ записи даты и пр.), задавать направление текста и метод его выравнивания, определять шрифт и начертание символов, управлять отображением и видом рамок, задавать фоновый цвет.

4.3.4. Изменение высоты строк и ширины столбцов

При работе с длинными текстовыми заголовками, высокими шрифтами и записью чисел денежным стилем может оказаться, что не хватает стандартной ширины столбца или высоты строки. Поменять ширину столбцов и высоту строк можно с помощью мыши:

– перетащить правую границу заголовка столбца и/или нижнюю границу заголовка строки (перетаскиванием можно не только увеличивать, но и уменьшать ширину столбца и высоту строки);

– выполнить двойной щелчок на правой границе заголовка столбца и/или на нижней границе заголовка строки. Строка или столбец будут увеличены до размеров самого высокого символа и самого длинного текста в этих рядах ячеек.

4.4. Организация вычислений

При помощи Microsoft Excel удобно выполнять различные вычисления, оперируя данными, расположенными на рабочем листе. Результат вычислений определяется *формулами*, которые необходимо внести в ячейку рабочего листа. Microsoft Excel сам производит требуемые подсчеты и результат помещает в ячейку, содержащую формулу.

Формула может содержать числовые константы, ссылки на ячейки и функции Microsoft Excel, соединенные знаками арифметических операций: +, –, *, ^, /, %. Скобки позволяют изменять стандартный порядок выполнения действий. Если ячейка содержит формулу, то в рабочем листе отображается только текущий результат вычисления этой формулы. Чтобы увидеть саму формулу, надо выделить ячейку (сделать ее текущей) и посмотреть на запись, которая отображается в строке формул.

4.4.1. Автоматическое суммирование строк и столбцов

Автосуммирование выполняется с помощью кнопки *Автосумма*  на панели инструментов *Стандартная*.

Автосуммирование можно использовать для трех типов задач: обнаружить и просуммировать данные в строках или столбцах ближайшего к текущей ячейке диапазона; просуммировать данные в любом выделенном диапазоне ячеек; добавить итоговые суммы к ряду, содержащему частичные суммы.

Чтобы автоматически просуммировать данные в ближайшем к текущей ячейке диапазоне, нужно щелкнуть на кнопке *Автосумма* и убедиться, что требуемый диапазон выделен правильно (при необходимости выделение можно скорректировать) и нажать клавишу `Enter`. Просуммировать данные в любом диапазоне можно, выделив его, щелкнуть на кнопке *Автосумма*.

4.4.2. Составление элементарных формул

Элементарные формулы могут состоять только из арифметических операторов и адресов ячеек. В Microsoft Excel ввод формул в ячейках необходимо начинать со знака равенства =, а завершать нажатием клавиши Enter. Вместо чисел в формулах используются адреса ячеек, иначе говоря, *ссылки на ячейки*. Это означает, что результат расчета зависит от того, какие числа находятся в ячейках, участвующих в вычислении. Таким образом, ячейка, содержащая формулу, является *зависимой*. Значение в зависимой ячейке подлежит пересчету всякий раз, когда изменяются значения в ячейках, на которые указывают ссылки, входящие в формулу.

Ссылку на ячейку можно задать следующими способами:

1) ввести адрес ячейки вручную;

2) щелкнуть на нужной ячейке или выделить нужный диапазон, адрес которого нужно внести в формулу. Ячейка или диапазон при этом выделяются цветной пунктирной рамкой (рис. 2.7).

Для редактирования формулы следует дважды щелкнуть на соответствующей ячейке. При этом ячейки (диапазоны), от которых зависит значение формулы, выделяются на рабочем листе цветными рамками, а сами ссылки отображаются в ячейке и в строке формул тем же цветом. Это облегчает редактирование и проверку правильности формул.

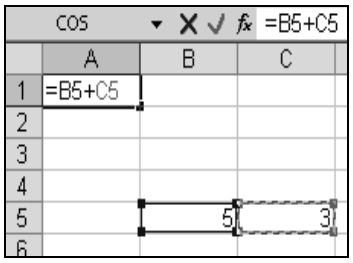


Рис. 2.7. Ввод формулы

Примеры записи формул:

1)= (B4 + D2) / F5;

2)= A1 * B2;

3)= C3 ^ 3.

4.4.3. Использование Мастера функций при составлении формул

Иногда в вычислениях приходится использовать формулы, содержащие числа и функции. *Функции* – заранее определенные формулы, которые выполняют вычисления по заданным величинам, называемым *аргументами*, и в указанном порядке. Эти функции позволяют выполнять как простые, так и сложные вычисления.

Примеры записи формул, содержащих функции:

1) =СУММ (C22:C26) – в этой формуле используется функция СУММ для того чтобы сложить значения в диапазоне ячеек C22:C26 (диапазон C22:C26 – аргумент). Эта формула аналогична формуле =(C22+C23+C24+ C25+C26);

2) =КОРЕНЬ(B10) – извлекает квадратный корень из числа, содержащегося в указанной ячейке;

3) =СЕГОДНЯ() – вставляет в ячейку текущую дату;

4) =ТДАТА() – вставляет в ячейку текущую дату и время.

Примечание. Функции СЕГОДНЯ() и ТДАТА() не требуют указания аргументов.

Некоторые функции, например, статистические или финансовые, используют несколько аргументов. В таких случаях аргументы отделяются друг от друга точкой с запятой.

Существуют различные типы аргументов: число, текст, логическое значение (*ИСТИНА* и *ЛОЖЬ*), массивы, значение ошибки (например, *#Н/Д*), или ссылки на ячейку. В качестве аргументов могут использоваться константы, формулы, или функции.

Чтобы упростить ввод функций при создании формул, содержащих функции, рекомендуется использовать диалоговое окно *Мастер функций*.

Процедура:

1) выделить ячейку, в которой нужно разместить формулу, содержащую функцию;

2) щелкнуть по кнопке *Вставка функции* f_x в строке формул. Откроется диалоговое окно *Мастер функций* (рис. 2.8);

3) в списке *Категория* выбрать нужную категорию функций. В поле *Выберите функцию* появится список функций выбранной категории;

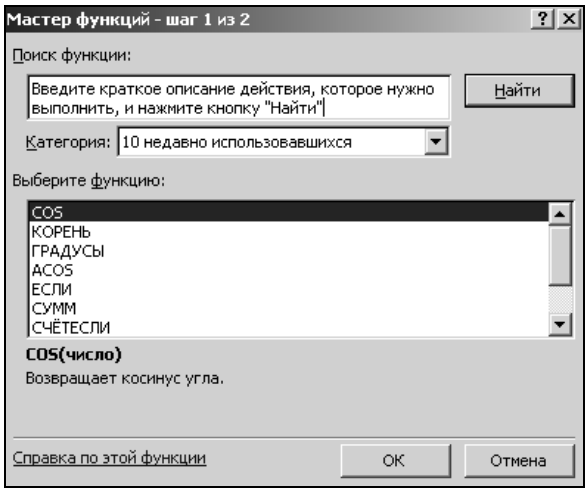


Рис. 2.8. Окно диалога *Мастер функций*

4) из списка функций выбрать нужную функцию и щелкнуть на кнопке ОК. Откроется окно *Аргументы функции* (рис. 2.9). При вставке функции в формулу диалоговое окно *Мастер функций* отображает имя

функции, все ее аргументы, описание функции и каждого аргумента, текущий результат функции и всей формулы;

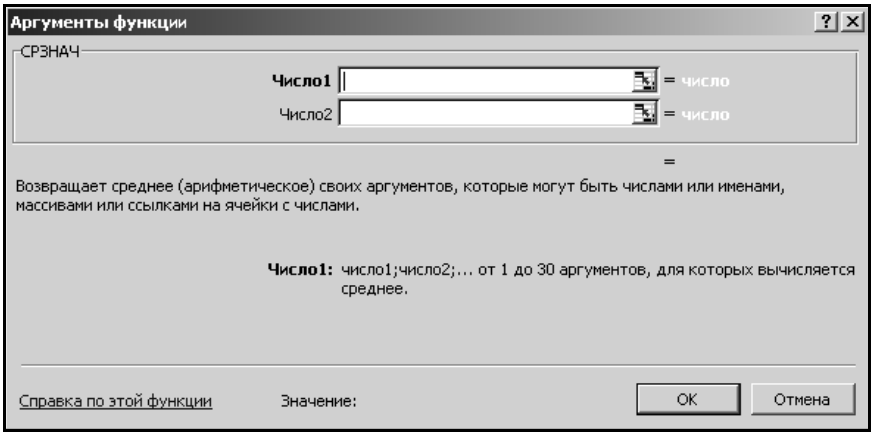


Рис. 2.9. Окно диалога *Аргументы функции*

5) в окне *Аргументы функции* ввести требуемые аргументы в соответствующие поля либо путем ввода с клавиатуры, либо выделением нужных ячеек или диапазонов ячеек непосредственно на рабочем листе. Если название аргумента указано полужирным шрифтом, то этот аргумент является *обязательными* и соответствующее поле должно быть заполнено. Аргументы, названия которых указаны обычным шрифтом, можно опустить;

6) щелкнуть на кнопке OK.

Все диалоговые окна программы Microsoft Excel, которые требуют указания номеров или диапазонов ячеек, содержат кнопки, присоединенные к соответствующим полям. При щелчке на такой кнопке диалоговое окно сворачивается до минимально возможного размера, что облегчает выбор нужной ячейки (диапазона) с помощью мыши (рис. 2.10).

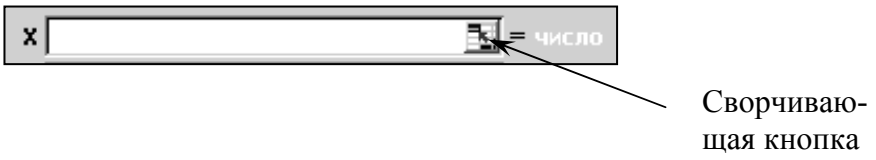


Рис. 2.10. Сворачивающая кнопка в окне *Аргументы функции*

4.4.4. Копирование формулы в диапазон ячеек

Копирование формулы в диапазон ячеек осуществляется так же, как и копирование данных. Чтобы внести формулу в диапазон, нужно выделить ячейку, содержащую формулу, и перетащить маркер заполнения вниз, вверх, вправо или влево на сколько требуется.

4.4.5. Создание формул с относительными и абсолютными адресами

Относительные адреса. Относительный адрес в формуле, например A1, основан на относительной позиции ячейки, содержащей формулу, и ячейки, на которую указывает адрес. При изменении позиции ячейки, содержащей формулу, изменяется и адрес, т. е. при копировании формулы адрес автоматически корректируется.

Пример 2.3. Пусть ячейка B2 содержит ссылку на ячейку A1. Тогда при копировании относительного адреса A1 из ячейки B2 в ячейку B3, она автоматически изменяется с = A1 на = A2 (рис. 2.11).

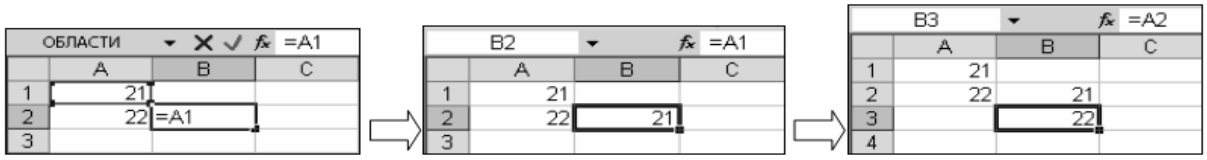


Рис. 2.11. Использование относительных ссылок

Абсолютные адреса. Запись *абсолютных адресов* содержит *знаки доллара* (например, A1). Абсолютный адрес ячейки в формуле всегда ссылается на ячейку, расположенную в определенном месте. При изменении позиции ячейки, содержащей формулу, абсолютный адрес не изменяется, т. е. при копировании формулы абсолютный адрес не корректируется.

Пример 2.4. Пусть ячейка B2 содержит ссылку на ячейку A1. Тогда при копировании абсолютного адреса A1 из ячейки B2 в ячейку B3, он остается прежним (рис. 2.12).

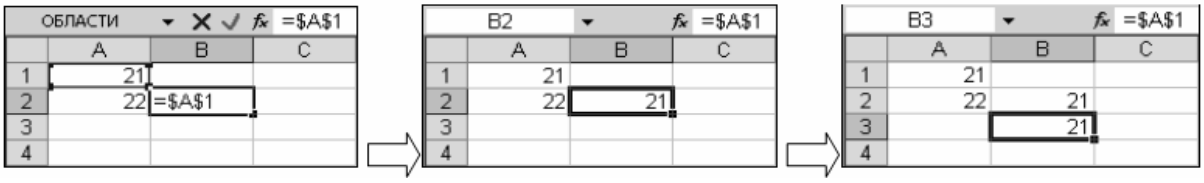


Рис. 2.12. Использование абсолютных ссылок

Смешанные адреса. В формулах можно также использовать *смешанные адреса*, которые задают столбец относительно, а строку абсолютно, или наоборот. При изменении позиции ячейки, содержащей формулу, относительный адрес изменяется, а абсолютный – нет, т. е. при копировании формулы относительная ссылка автоматически корректируется, а абсолютная ссылка не корректируется.

Пример 2.4. Абсолютный адрес столбцов приобретает вид $A1, $B1 и т. д. Абсолютный адрес строк приобретает вид A$1, B$1.

4.5. Диаграммы

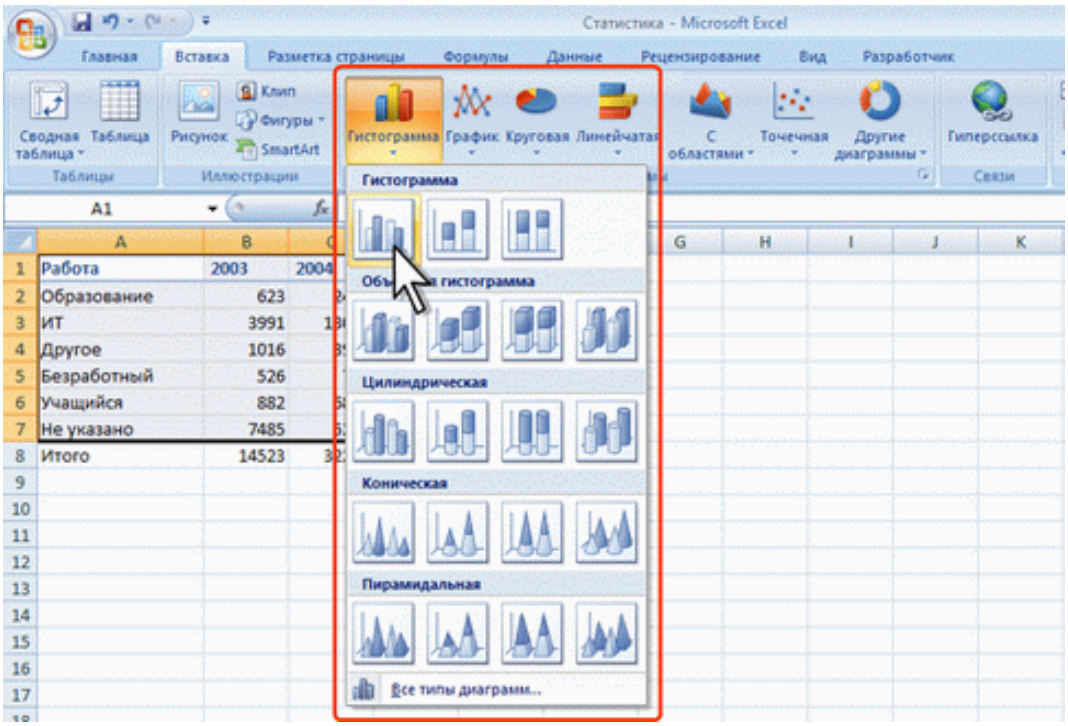
Диаграммы являются средством наглядного представления данных и облегчают выполнение сравнений, выявление закономерностей и тенденций данных. Можно составить графики, показывающие изменения величин во времени, или диаграммы, определяющие, какая доля целого приходится на отдельные его части. В составленные диаграммы можно вносить исправления: переупорядочивать данные или добавлять ранее неучтенные. С помощью *Мастера диаграмм* можно легко показать изменения данных в динамике, что актуально для выступлений и докладов. При изменении данных в рабочей таблице диаграммы изменяются автоматически.

4.5.1. Построение диаграмм

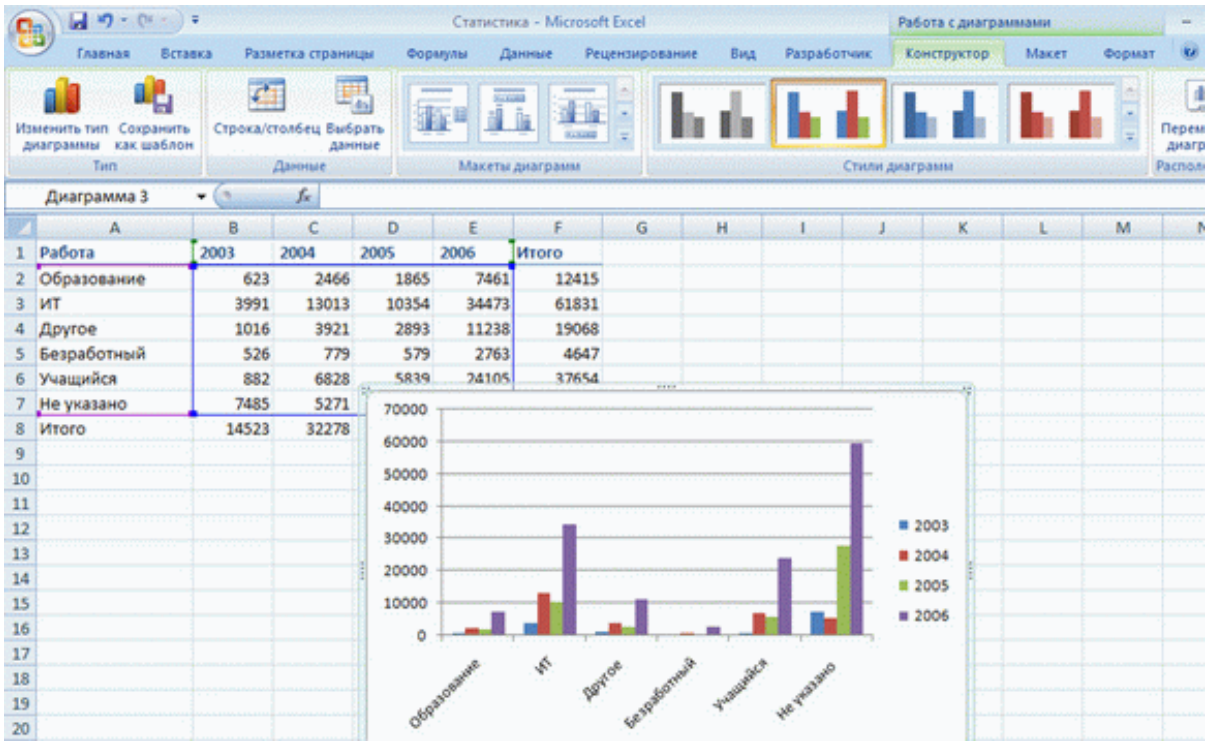
Выделите фрагмент таблицы, для которого создается диаграмма.

	A	B	C	D	E	F
1	Работа	2003	2004	2005	2006	Итого
2	Образование	623	2466	1865	7461	12415
3	ИТ	3991	13013	10354	34473	61831
4	Другое	1016	3921	2893	11238	19068
5	Безработный	526	779	579	2763	4647
6	Учащийся	882	6828	5839	24105	37654
7	Не указано	7485	5271	27892	59467	100115
8	Итого	14523	32278	49422	139507	235730

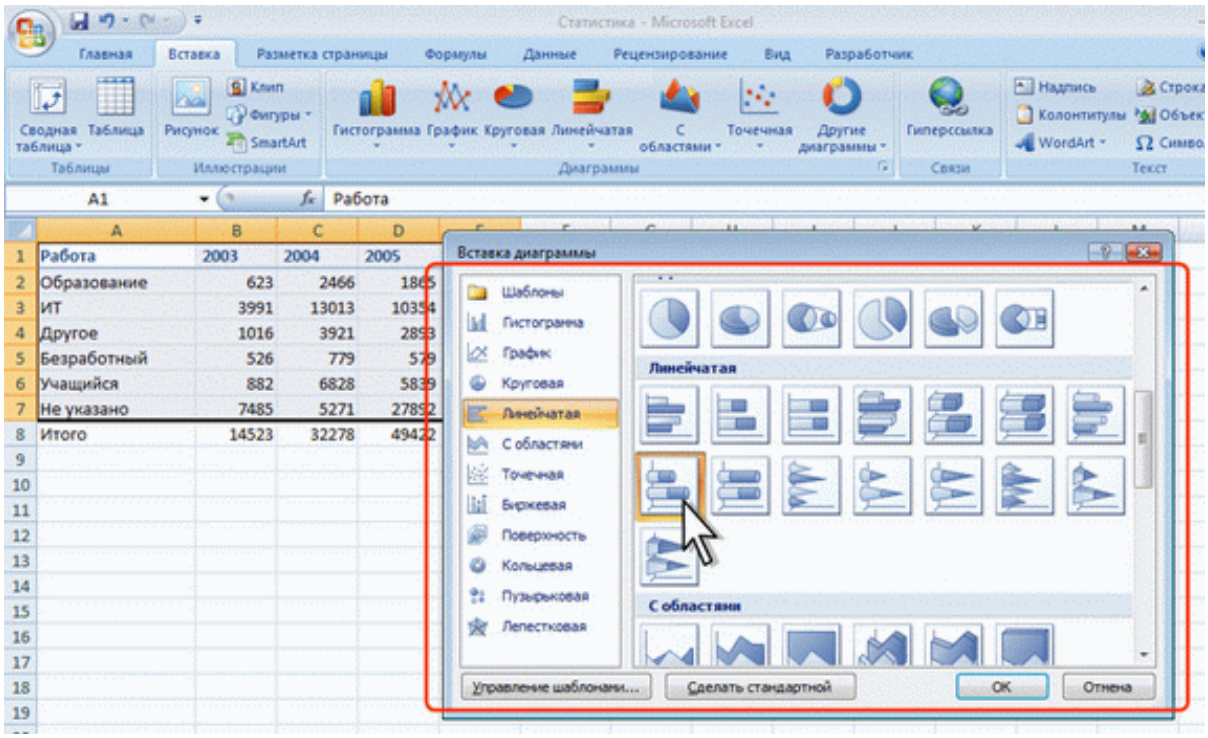
1. На вкладке **Вставка** в группе **Диаграммы** щелкните по кнопке с нужным типом диаграмм и в галерее выберите конкретный вид диаграммы.



2. На листе будет создана диаграмма выбранного вида.



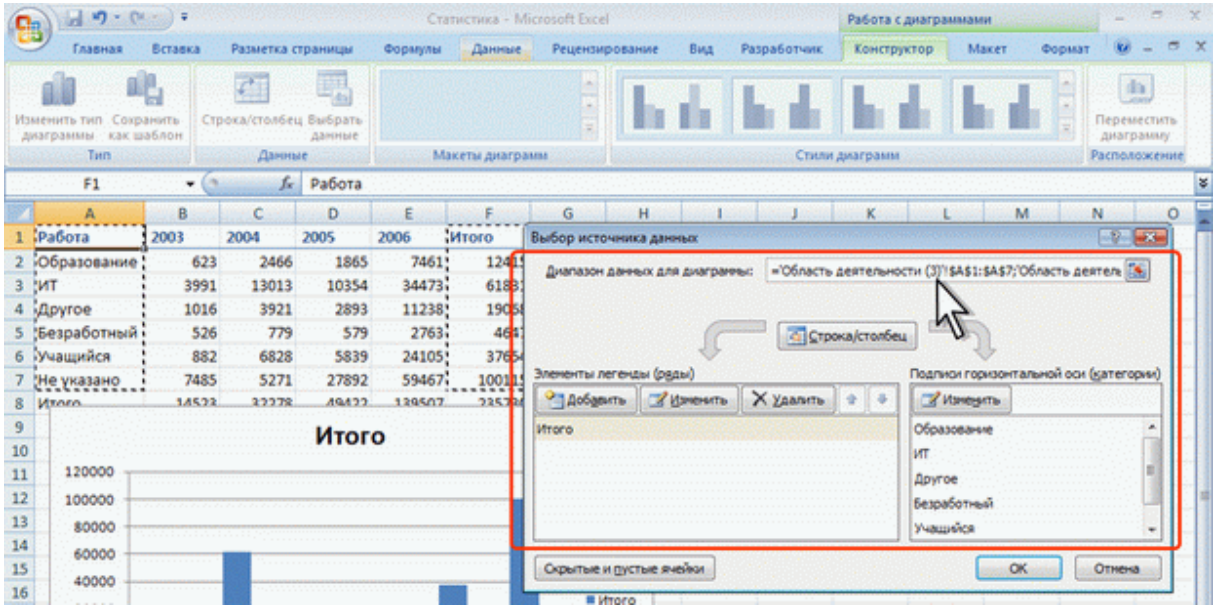
Если в группе **Диаграммы** не отображается подходящий тип и вид диаграммы, щелкните по кнопке группы **Диаграммы** и выберите диаграмму в окне **Вставка диаграммы**.



Для создания диаграммы стандартного типа достаточно выделить фрагмент листа и нажать клавишу **F11**. Для удаления диаграммы достаточно выделить ее и нажать клавишу **Delete**.

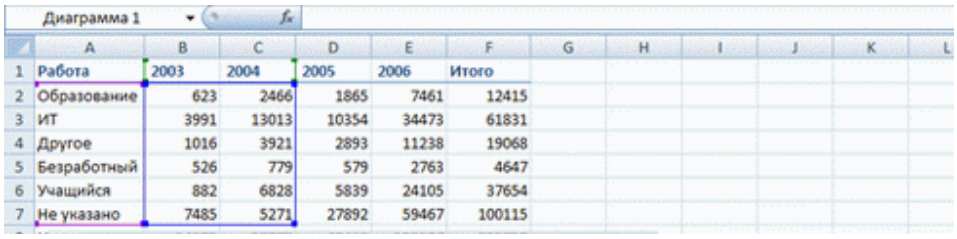
4.5.2. Изменение данных диаграммы

В группе **Работа с диаграммами/Конструктор** нажмите кнопку **Выбрать данные**. В окне **Выбор источника данных** очистите поле **Диапазон данных для диаграммы**, а затем выделите на листе новый диапазон данных.



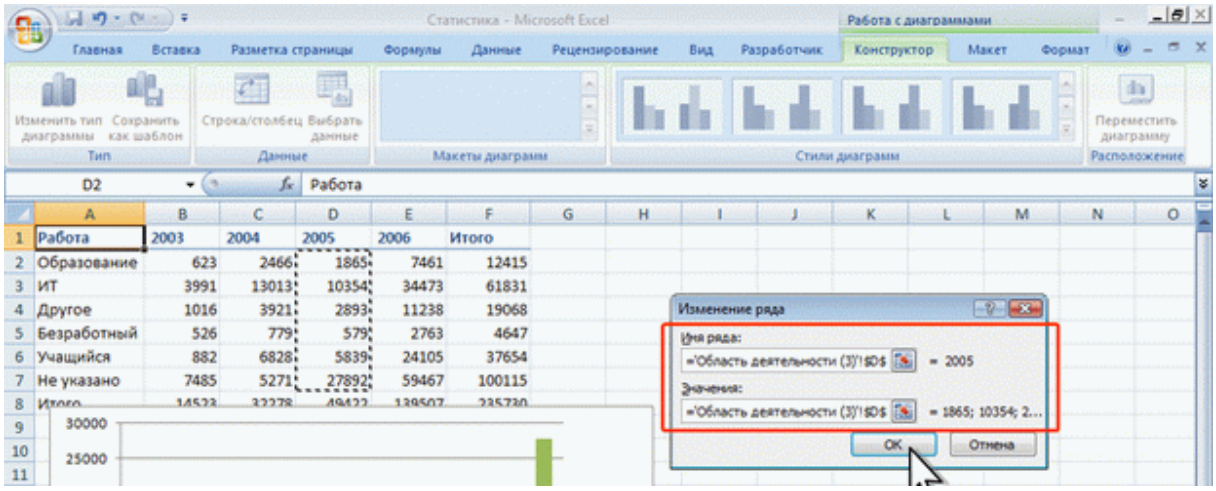
Изменение диапазона источника данных

В тех случаях, когда диаграмма расположена на листе с данными, изменить диапазон источника данных можно перетаскиванием маркеров диапазонов данных. Маркеры диапазонов отображаются на листе при выделении диаграммы. Как правило, выделяются три диапазона: в зеленой рамке – названия рядов диаграммы (в таблице на ячейки B1:C1), в сиреневой рамке – названия категорий (в таблице на ячейки A2:A7), в синей рамке – значения рядов данных (в таблице на ячейки B2:C7).



Для того чтобы изменить ряды данных, следует перетащить зеленую рамку на другие ячейки, а для добавления или удаления рядов данных следует перетащить зеленый маркер выделения. Для того чтобы изменить категории, следует перетащить сиреневую рамку на другие ячейки, а для добавления или удаления категорий следует перетащить сиреневый маркер выделения. Для того чтобы изменить одновременно категории и ряды данных, следует перетащить синюю рамку на другие ячейки (зеленая и сиреневая рамки при этом переместятся автоматически), а для добавления или удаления одновременно категорий и рядов данных следует перетащить синий маркер выделения. Для изменения рядов данных, подписей осей, легенды можно использовать окно **Выбор источника данных**.

В группе **Работа с диаграммами/Конструктор** нажмите кнопку **Выбрать данные**. Для добавления ряда данных в окне **Выбор источника данных** нажмите кнопку **Добавить**. В окне **Изменение ряда** очистите поле **Имя ряда**, а затем выделите на листе ячейку, содержащую название ряда данных; очистите поле **Значение**, а затем на листе выделите ячейки, содержащие значения ряда данных.



Для удаления ряда данных в окне **Выбор источника данных** выделите название этого ряда и нажмите кнопку **Удалить**.

4.5.3. Редактирование диаграмм

Готовую диаграмму можно изменить. При щелчке на элементе диаграммы (ряды данных, оси координат, заголовок диаграммы, область построения и проч.) он выделяется маркерами, а при наведении на него указателя мыши – комментируется *всплывающей подсказкой*.

Изменение типа диаграммы.

1) щелчком мыши выделить диаграмму, вывести на экран вкладки **Работа с диаграммами**, если она не отображается на экране;

2) щелкнуть на присоединенной стрелке кнопки *Тип диаграммы* и в появившейся таблице типов выбрать нужный тип диаграммы. При этом тип построенной диаграммы автоматически изменяется на выбранный.

Тип диаграммы можно также изменить с помощью команды *Тип диаграммы* из контекстного меню. При этом открывается диалоговое окно *Тип диаграммы*, в котором можно выбрать иной тип и подтип диаграммы.

Форматирование сетки и легенды. Форматы некоторых диаграмм содержат *координатную сетку* и *легенду*. Сетку и легенду можно добавить к любому типу диаграмм. На диаграмме с сеткой легче оценить порядок величин, а наличие легенды помогает пояснить диаграмму и делает ее легко читаемой. Если же сетка и легенда мешают, то их можно удалить.

Операции с сеткой производятся на вкладке *Линии сетки* диалогового окна *Параметры диаграммы*, которое можно вывести на экран либо с помощью вкладки **Работа с диаграммами** →*Параметры диаграммы*, либо с помощью контекстного меню области диаграммы.

Добавить или удалить легенду, а также управлять ее размещением в поле диаграммы, можно на вкладке *Легенда* диалогового окна *Параметры диаграммы*. Переместить легенду и изменить ее размеры можно также с помощью мыши.

4.5.4. Оформление диаграмм

Если необходимо выделить на диаграмме отдельный элемент данных или придать какой-либо диаграмме особый вид, можно сделать это, внося в диаграмму дополнительные линии, стрелки и выделяя элементы диаграммы различными цветами.

Маркеры отдельных данных можно выделить стрелочкой или линией, а также изменить цвет ряда данных или отдельных их значений. В диаграмму можно также вносить текстовые поля, менять шрифт и другие атрибуты уже имеющихся текстовых фрагментов.

Изменение цвета элементов диаграммы. Если пользователя не устраивает цветовая гамма на диаграмме и нужно представить данные другими цветами, то цвета любых элементов диаграммы (сетки, фона легенды и текста, рамок, линий, маркеров рядов) можно легко изменить с помощью кнопки *Цвет заливки*  из панели инструментов *Форматирование* или соответствующей команды из контекстного меню. Изменять формат объектов можно и при помощи команд меню *Формат*.

Примечание. Имя команды форматирования в контекстном меню зависит от типа выбранного объекта. Для легенды она называется *Формат легенды*, для маркеров ряда данных или сетки эта команда будет называться *Формат рядов данных* или *Формат линий сетки* соответственно, для всей диаграммы – *Формат области диаграммы* и т. д. Выполнение такой команды открывает диалоговое окно форматирования элемента диаграммы, различные вкладки которого позволяют изменять не только цвета, но и другие параметры отображения выбранного элемента данных.

4.6. Управление данными

4.6.1. Сортировка данных

При работе с Microsoft Excel часто приходится сортировать и обрабатывать списки данных. В списке заголовки столбцов определяют *поля*, а строки содержат *записи*. В каждом поле содержится информация определенного типа, например фамилии, имена и т. д., а запись состоит из описания элемента списка. Совокупность данных в виде таблицы полей и записей называется *базой данных*. В Microsoft Excel понятия список и база данных взаимозаменяемы.

Прежде чем вывести на экран или напечатать данные списка в определенном порядке, необходимо произвести сортировку данных. Для каждого столбца списка данных можно задать свой способ сортировки. Имя столбца (поля), по которому проводится сортировка, называется *ключом сортировки*.

Сортировка диапазона данных по возрастанию или убыванию:

1) выделить любую ячейку этого диапазона. Весь диапазон, включающий выделенную ячейку, автоматически подлежит сортировке;

2) меню *Данные→Сортировка* – откроется окно диалога *Сортировка диапазона* (рис. 2.18);

3) в поле *Сортировать по* в раскрывающемся списке выбрать ключ сортировки и указать вид сортировки *по возрастанию* (от *А* до *Я* или от *0* до *9*) или *по убыванию* (от *Я* до *А* или от *9* до *0*) с помощью соответствующих переключателей;

4) OK.

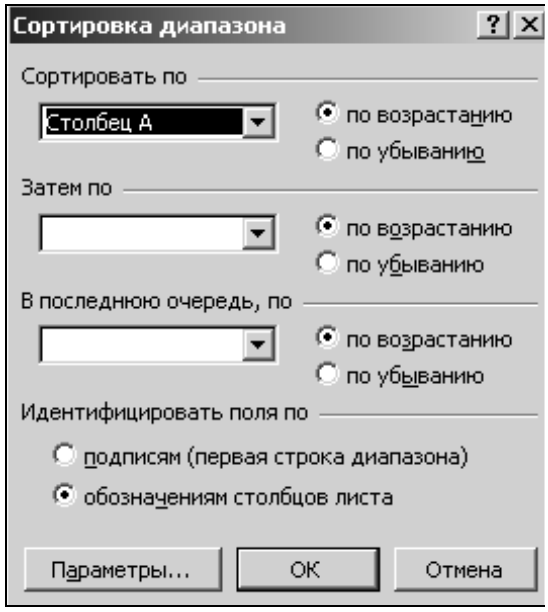


Рис. 2.18. Диалоговое окно *Сортировка диапазона*

Примечание. Простейшую сортировку по возрастанию или по убыванию можно также выполнять с помощью соответствующих кнопок на панели инструментов *Стандартная* (и).

Сортировка строк по двум или трем ключам. Иногда приходится использовать более одного ключа сортировки. Лучше всего, если в сортируемом списке будут заголовки столбцов.

Процедура:

1) выделить ячейку в списке, который требуется отсортировать;

2) меню *Данные→Сортировка* – откроется окно диалога *Сортировка диапазона* (см. рис. 2.18);

3) указать ключи сортировки в полях *Сортировать по* и *Затем по*, если выполняется сортировка по двум критериям. Если нужно выполнить сортировку по трем критериям, то дополнительно указывается ключ сортировки в поле *В последнюю очередь, по*;

4) выбрать остальные параметры сортировки OK.

4.6.2. Подведение итогов

Добавление промежуточных итогов. Подведение промежуточных итогов можно осуществлять по нескольким показателям: определению количества элементов списка, суммированию величин, нахождению максимального, минимального или среднего значения, а также использовать более сложные статистические функции, такие как поиск стандартного отклонения или дисперсии.

При вставке автоматических промежуточных итогов Microsoft Excel изменяет разметку списка, что позволяет отображать и скрывать строки каждого промежуточного итога.

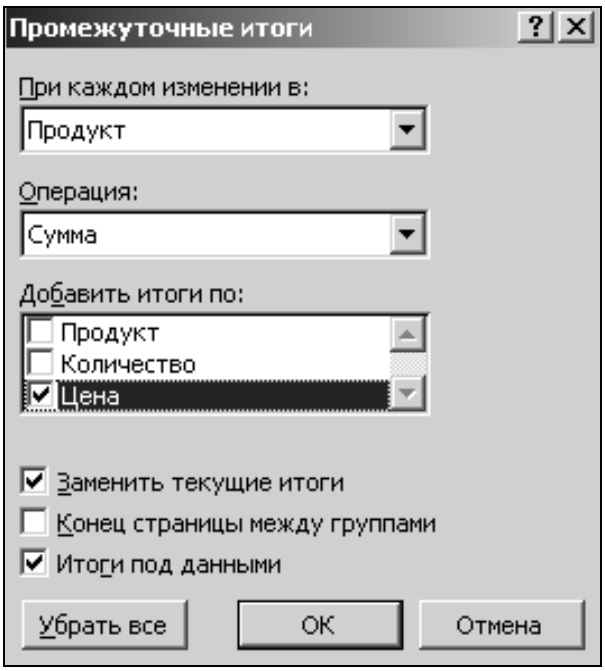


Рис. 2.19. Диалоговое окно *Промежуточные итоги*

Перед тем, как вставить промежуточные итоги, необходимо отсортировать список, чтобы сгруппировать строки, по которым нужно подвести итоги. После этого можно подсчитать промежуточные итоги любого столбца, содержащего числа.

Процедура:

1) убедиться, что данные для подсчета итогов представлены в формате списка: первая строка каждого столбца содержит подпись, остальные строки – однотипные данные. Пустые строки или столбцы в списке отсутствуют;

2) указать ячейку в столбце для итогов;

3) отсортировать список по возрастанию или убыванию;

4) меню *Данные→Итоги* – откроется окно диалога *Промежуточные итоги* (рис. 2.19);

5) в поле *При каждом изменении в* выбрать столбец для подсчета итогов;

6) в поле *Операция* выбрать функцию для вычисления итогов;

7) в поле *Добавлять итоги по* выбрать столбцы, содержащие значения, по которым необходимо подвести итоги;

8) чтобы за каждым итогом следовал разрыв страницы, установить флажок *Конец страницы между группами*;

9) чтобы итоги отображались над строками данных, а не под ними, снять флажок Итоги под данными и OK.

4.7. Анализ данных

4.7.1. Подбор параметра

При анализе данных бывает необходимо определить, как повлияет на результат формулы изменение одной из переменных. Например, пользователю может потребоваться найти, на сколько следует увеличить торговлю, чтобы достичь определенного уровня дохода, или какой кредит нужно взять его фирме, чтобы сумма ежемесячной выплаты не превосходила фиксированного значения. Когда нужно определить, насколько следует изменить переменную, чтобы результат формулы, в которую она входит, равнялся заданной величине, используют функцию *Подбор параметра* из меню *Сервис*. Эта функция позволяет исследовать уравнения и формулы, исходя из итогового результата. Иными словами, задается требуемый результат, выбирается изменяемый параметр формулы и запускается программа поиска значения параметра, при котором будет достигнут указанный результат.

Пример:

– составить формулу, вычисляющую размер платежей по кредиту фиксированного размера в зависимости от величины процентной ставки.

– определить максимально допустимый размер кредита по заданной величине выплат и при фиксированном проценте.

1) Для расчета величины ежемесячных выплат по кредиту используется функция *ПЛТ*. Ее синтаксис: *ПЛТ (ставка;кпер;пс;бс;тип)*, где *ставка* – процентная ставка по ссуде (в примере 8,5 %), *кпер* – общее число выплат по ссуде (в примере 360 месяцев), *пс* – приведенная к текущему моменту стоимость, или общая сумма, которая на текущий момент равноценна ряду будущих платежей, называемая также основной суммой (в примере 120 000 р. Величина в данном случае включается в формулу со знаком «–», т. к. мы исходим из того, что на момент расчета никакие вы-

платы не осуществлялись). Аргументы *бс* и *тип* для данной функции обязательными не являются (рис. 2.21).

2) Функция *ПЛТ* помещает в ячейку B5 величину ежемесячных выплат для заданного кредита при процентной ставке 8,5 % годовых.

3) Пусть имеется возможность ежемесячно выплачивать по кредиту сумму 900 р. при процентной ставке 8,5 %:

– выделить ячейку, для которой задается величина выплат (в примере – ячейка B5);

– выполнить команду меню *Сервис→Подбор параметра* – откроется окно диалога *Подбор параметра* (рис. 2.22);

B5 =ПЛТ(B1/12;B2;-B3)

	A	B
1	Проценты	8,5%
2	Срок кредита (в месяцах)	360
3	Кредит	120 000р.
4		
5	Величина ежемесячных выплат	923р.

Рис. 2.21. Использование функции ПЛТ

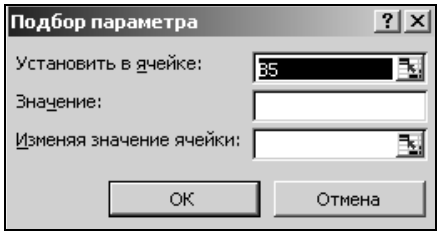


Рис. 2.22. Окно диалога *Подбор параметра*

– в поле *Значение* ввести заданную величину выплат (в примере – 900 р.)

– ОК – откроется окно *Результат подбора параметра*, сообщающее, что решение найдено (рис. 2.23);

– в поле *Изменяя значение ячейки* ввести адрес ячейки, для которой выполняется подбор (в примере – ячейка B3);

– ОК – новые значения будут введены в соответствующие ячейки (рис. 2.24).

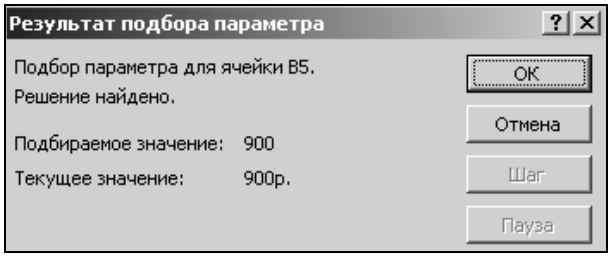


Рис. 2.23. Окно диалога *Результат подбора параметра*

B5 =ПЛТ(B1/12;B2;-B3)

	A	B
1	Проценты	8,5%
2	Срок кредита (в месяцах)	360
3	Кредит	117 048р.
4		
5	Величина ежемесячных выплат	900р.

Рис. 2.24. Результат работы подбора параметра

4.7.2. Таблицы подстановки данных

Введя формулы в ячейки рабочего листа, можно провести их анализ и изучить область возможных значений этих формул. *Таблицы подстановки данных* позволяют оперативно вычислять все значения в каждой операции.

Эти таблицы представляют собой диапазоны ячеек, показывающие результаты подстановки различных значений в одну или несколько формул.

Например, есть несколько допустимых комбинаций данных, которые нужно сравнить. Пользователю может потребоваться сравнить размеры выплат по кредиту для различных процентных ставок или для различных сроков кредита. Или потребуется оценить влияние роста различных показателей торговли на текущий доход. Вместо того чтобы подбирать параметры и поочередно следить за изменением соответствующих величин, можно составить таблицу данных и сравнить сразу несколько результатов. В таблицах подстановки данных варьируются одна или две переменные, а количество строк таблицы может быть произвольным. Например, можно получить размер платежей по кредиту в зависимости от величины процентной ставки, колеблющейся между 6 и 12 %, или найти влияние процента роста торговли (на 2, 3, 4 или 5 %) на текущий доход.

Существуют два типа таблиц подстановки данных: *таблицы подстановки с одной переменной* и *таблицы подстановки с двумя переменными*. Названия говорят сами за себя. Таблицы данных с одной переменной позволяют исследовать влияние различных значений одной переменной на результат одной или нескольких формул. В таблицах с двумя переменными анализируется зависимость результата одной формулы от изменения двух входящих в нее переменных.

Создание таблицы подстановки с одной переменной. Предположим, следует сформировать таблицу подстановки с одной переменной, чтобы введенные значения были расположены либо в столбце (ориентированные по столбцу), либо в строке (ориентированные по строке). Формулы, используемые в таблицах подстановки с одной переменной, должны ссылаться на одну и ту же на ячейку ввода.

Примечание. Ячейка ввода – это ячейка, в которую подставляются все значения из таблицы данных. Ячейкой ввода может быть любая ячейка листа. Хотя ячейка ввода не обязана входить в таблицу данных, формулы в таблице данных должны ссылаться на ячейку ввода.

Процедура:

1) либо в отдельный столбец, либо в отдельную строку ввести список значений, которые следует подставлять в ячейку ввода;

2) выполнить одно из следующих действий:

– если значения в таблице подстановки ориентированы по столбцу, ввести формулу в ячейку, расположенную на одну строку выше и на одну

ячейку правее первого значения. Правее первой формулы можно ввести любые другие формулы;

– если значения в таблице подстановки ориентированы по строке, ввести формулу в ячейку, расположенную на один столбец левее и на одну строку ниже первого значения. В том же столбце, но ниже можно ввести любые другие формулы.

3) выделить диапазон ячеек, содержащий формулы и значения подстановки;

4) выполнить команду меню *Данные→Таблица подстановки→*окно диалога *Таблица подстановки* (рис. 2.25);

5) выполнить одно из следующих действий:

6) если значения в таблице расположены по столбцам, ввести ссылку на ячейку ввода в поле *Подставлять значения по строкам в*;

7) если значения в таблице расположены по строкам, ввести ссылку на ячейку ввода в поле *Подставлять значения по столбцам в*;

8) ок – в столбец, находящийся справа от столбца подставляемых значений, или в строку, находящуюся ниже строки подставляемых значений, будут помещены результаты заданной формулы для различных аргументов.

Создание таблицы подстановки с двумя переменными. С помощью таблиц подстановки данных с двумя переменными можно исследовать влияние одновременно двух переменных на результат формулы. Формула должна ссылаться на две различные ячейки ввода.

Процедура:

1) в ячейку листа ввести формулу, которая ссылается на две ячейки ввода;

2) в том же столбце ниже формулы ввести значения подстановки для первой переменной;

3) ввести значения подстановки для второй переменной правее формулы в той же строке;

4) выделить диапазон ячеек, содержащий формулу и оба набора данных подстановки;

5) выполнить команду меню *Данные→Таблица подстановки→*окно диалога *Таблица подстановки* (рис. 2.25);

6) в поле *Подставлять значения по столбцам в* ввести ссылку на ячейку ввода для значений подстановки в строке;

7) в поле *Подставлять значения по строкам в* введите ссылку на ячейку ввода для значений подстановки в столбце;

8) OK – результаты формул будут занесены в таблицу подстановки.

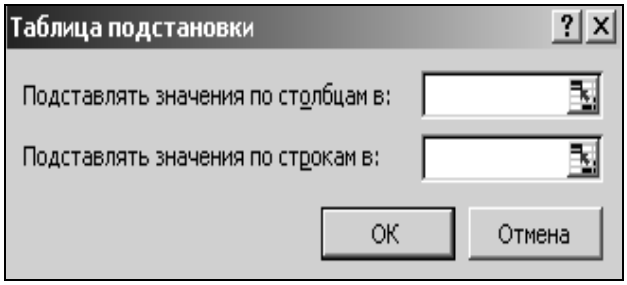


Рис. 2.25. Окно диалога *Таблица подстановки*

5. МАТЕМАТИЧЕСКИЙ ПАКЕТ MCAD

5.1. Детали интерфейса

В верхней части окна Mcad видны шесть характерных элементов интерфейса:

– строка заголовка – строка с именем системы и текущего документа, а также кнопками управления окном системы;

– строка меню – строка, открывающая доступ к пунктам меню с различными командами;

– панель инструментов – панель с кнопками (значками), обеспечивающими быстрое исполнение наиболее важных команд при работе с системой;

– панель форматирования – панель с кнопками (значками), обеспечивающими быстрое форматирование текстовых и формульных блоков в документах;

– панель вывода палитр математических знаков – панель с кнопками (значками), выводящими палитры специальных математических знаков и греческих букв;

– координатная линейка – линейка с делениями, позволяющая (если это нужно) точно располагать блоки по горизонтали.

Курсор ввода и линия раздела страниц. Обычно в окне редактирования можно увидеть два важных объекта – курсор ввода в виде красного крестика и вертикальную линию, отделяющую текущую страницу от соседней (справа). Положение линии определяется заданными по умолчанию параметрами страницы документа. Курсор ввода устанавливается мышью.

Строка меню. В системе Mcad_2000 представлены следующие меню:

– *Файл* – работа с файлами, сетью Internet и электронной почтой;

– *Правка* – редактирование документов;

– *Просмотр* – изменение способов представления документа и скрытие/отображение элементов интерфейса;

– *Вставка* – вставка объектов и их шаблонов (включая графику);

– *Форматирование* – изменение формата объектов;

– *Математика* – управление процессом вычислений;

– *Символика* – выбор операций символьного процессора;

– *Окно* – управление окнами системы;

– *Помощь* – работа со справочной базой данных о системе, центром ресурсов и электронными книгами.

Меню Mcad – контекстные, т. е. число позиций в них и их назначение зависят от состояния системы. Указанные выше меню характерны для рабочего состояния, когда идет редактирование документа.

Для активизации строки меню без применения мыши достаточно нажать клавишу `Alt`. Затем, нажимая клавиши перемещения курсора, можно перемещать световое выделение по позициям меню. Выбрав нужное меню, для его раскрытия достаточно нажать клавишу `Enter`. Можно также нажать клавишу `Alt` и одновременно клавишу, которая подчеркнута в названии нужного меню.

Панель инструментов содержит несколько групп кнопок управления, каждая из которых дублирует наиболее важные команды меню. При остановке указателя мыши на любой из кнопок появляется подсказка с именем этой кнопки. Рассмотрим назначение кнопок на панели инструментов.

Кнопки операций с файлами:

New (Новый) – создание нового документа с очисткой окна редактирования;

Open (Открыть) – загрузка ранее созданного документа с выбором его файла из диалогового окна;

Save (Сохранить) – запись текущего документа с его текущим именем.

Печать и контроль документов:

Print (Печать) – распечатка документа на принтере;

Print Preview (Предпросмотр) – предварительный просмотр документа;

Check Spelling (Правописание) – проверка орфографии в документе.

Кнопки операций редактирования:

Cut (Вырезать) – перенос выделенной части документа в буфер обмена с очисткой этой части документа;

Copy (Копировать) – перенос выделенной части документа в буфер обмена с сохранением выделенной части документа;

Paste (Вставить) – перенос содержимого буфера обмена в окно редактирования на место, в котором находится курсор ввода.

Undo (Отменить) – отмена предшествующей операции редактирования;

Redo (Восстановить) – повторение ранее отмененной операции. 

Кнопки размещения блоков (текстовых, формульных, графических и т.д.):

Align Across (Привязка через) – блоки выравниваются по горизонтали, располагаясь слева направо в одной строке; 

Align Down (Привязать) – блоки выравниваются по вертикали, располагаясь сверху вниз в одном столбце. 

Кнопки операций с выражениями:

Insert Function (Вставить функцию) – вставить функцию из списка, появляющегося в диалоговом окне; 

Insert Unit (Вставить юнит) – вставка размерных единиц; 

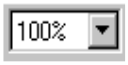
Calculate (Подсчет) – вычисление выделенного выражения. Если документы большие, то при их изменениях не всегда выгодно запускать вычисления с самого начала. Эта кнопка позволяет инициировать вычисления только для выделенных блоков, что может сократить время вычислений. 

Доступ к дополнительным возможностям Mathcad:

Insert Giperlink (Вставить гиперссылку) – обеспечивает создание гиперссылки; 

Component Wizard (Вставить компонент) – открывает окно мастера, дающего удобный доступ ко всем компонентам системы; 

Run MathConnex (Запуск MathConnex) – запуск системы для симулирования блочно заданных устройств. 

Для оперативного изменения масштаба отображения символов в текущем документе на панели инструментов имеется раскрывающийся список *Zoom* (Масштаб). 

Кнопки управления ресурсами:

Resource Center (Центр ресурсов) – обеспечивает доступ к центру ресурсов. Центр ресурсов – это мощная база данных, объединяющая в себе встроенные в систему электронные книги, обучающую систему, справочную систему, многочисленные примеры применения – «шпаргалки», средства общения с фирмой-разработчиком системы. Кроме того, центр ресурсов предоставляет выход в Internet, средства регистрации и доступа к Web-библиотеке Mathcad и т. д.; 

Help (Помощь) – обеспечивает доступ к ресурсам справочной базы данных системы. 

Панель форматирования

Для выбора различных вариантов отображения текстовых блоков и символов служат три раскрывающихся списка:

Style (Стиль) – выбор стиля отображения текстовых блоков;

Font (Шрифт) – выбор шрифта для символов;

Font Size (Размер шрифта) – выбор размера символов.

Для изменения начертания символов предназначены три кнопки:

Bold (Жирный) – полужирное начертание;

Italic (Курсив) – курсивное начертание;

Underline (Подчеркнутый) – подчеркнутое начертание.

Кнопки задания режима выравнивания текста:

Align Left (Привязка влево) – выравнивание текстов по левой границе;

Align Center (Привязка к центру) – выравнивание текстов по центру;

Align Right (Привязка вправо) – выравнивание текстов по правой границе.

Кнопки для создания списков:

Bullets (Маркеры) – создание маркированного списка;

Numbering (Номера) – создание нумерованного списка.

Строка состояния находится в нижней части системы. Она является элементом оперативной помощи по работе с системой, поскольку отображает ее текущее состояние.

Палитры математических знаков служат для вывода шаблонов математических операторов (цифр, знаков арифметических операций, матриц, интегралов, производных и т. д.), функций системы и отдельных символов, например, греческих букв.

5.2. Редактирование с применением клавиатуры

Формы курсора ввода:

– крестообразный красный курсор (+) служит для указания места ввода новых блоков (текстовых, формульных или графических). Курсор

имеет такой вид только на пустом месте экрана и может перемещаться клавишами управления курсором или устанавливаться щелчком мыши;

– курсор в виде красной вертикальной черты (курсор ввода) служит для указания на отдельные элементы блоков и обычно используется для ввода данных и заполнения шаблонов. В текстовых блоках позволяет указать места вставки или удаления отдельных объектов;

– курсор в виде синих уголков разного размера (⌞ или ⌟), выделяющих отдельные части выражения или выражение целиком. Вид курсора зависит от направления ввода.

Клавиши для выделения:

↑ – превращает курсор в выделяющий уголок и расширяет его;

↓ – сужает выделяющий уголок;

→ – перемещает курсор и меняет вид уголка;

← – перемещает курсор и меняет вид уголка;

`Shift` ↑ – выводит курсор из выражения вверх в свободное поле, делая его крестообразным;

`Shift` ↓ – выводит курсор из выражения вниз в свободное поле;

`Shift` → – выводит курсор из выражения вправо в свободное поле

`Shift` ← – выводит курсор из выражения влево в свободное поле

Пробел – заключает в рамку операнд, действуя как несколько нажатий клавиши ↑, [и выводит курсор из выделенного выражения (двойное нажатие)];

`Ins` – меняет направление ввода и расположение курсора. Например, если курсор имеет вид ⌟, то ввод будет идти вправо.

Управляющие клавиши:

`Tab` – в тексте перемещает курсор в начало следующего слова, в уравнении управляет выделением частей блока (в частности, выделяя выражения в скобках);

`Shift+Tab` – в тексте перемещает курсор в начало предыдущего слова, в уравнении управляет выделением частей блока (в частности, выделяя выражения в скобках);

`PgUp` – перемещает курсор и вызывает прокрутку на пять строк вверх;

`PgDn` – перемещает курсор и вызывает прокрутку на пять строк вниз;

`Ctrl+PgUp` – вызывает прокрутку на одно окно вверх;

`Ctrl+PgDn` – вызывает прокрутку на одно окно вниз;

`Home` – устанавливает курсор в начало предыдущего блока;

Ctrl+Home – устанавливает курсор в начало документа;
Ctrl+End – устанавливает курсор в конец документа.
Клавиши для создания объектов:
@ – создание шаблона двумерной графики;
Ctrl+@ – создание шаблона трехмерной графики;
Ctrl+G – замена латинской буквы на греческую;
Ctrl+V – задание шаблона вектора или матрицы;
Ctrl+P – ввод греческой буквы π;
Ctrl+Z – ввод математического символа бесконечности.

5.2.1. Правила ввода текстовых комментариев с помощью текстового редактора

1. Ввести символ " (одна двойная кавычка). В появившемся прямоугольнике можно начинать вводить текст.

2. С помощью раскрывающегося списка выбора шрифта на панели форматирования установить нужный шрифт.

3. Набрать текст.

4. Для завершения ввода текста отвести курсор мыши в сторону от текстового блока и щелкнуть левой кнопкой мыши или нажать Ctrl+Shift+Enter.

Блок имеет маркеры изменения размера в виде маленьких черных прямоугольников, перемещая которые, блок можно увеличивать или уменьшать в том или ином направлении. Выделенные рамкой текстовые блоки можно переносить на другое место, поместив курсор мыши на рамку – он при этом превращается в изображение черной ладошки.

Для коррекции текста надо подвести курсор мыши к месту коррекции и щелкнуть левой кнопкой мыши. Появится рамка текстового блока, а на месте щелчка мышью – курсор ввода.

5.3. Алфавит входного языка системы Mathcad

1) Алфавит системы Mathcad содержит:

- малые и большие латинские буквы и греческие буквы;
- арабские цифры от 0 до 9;
- системные переменные;
- математические операторы;
- имена встроенных функций;
- спецзнаки;

– малые и большие буквы кириллицы (при работе с русифицированными документами);

– все, что находится в палитрах математических знаков.

2) Константы:

– целочисленные (0,1, 23, -45 и т. д.);

– вещественные числа с мантиссой и порядком (12.3*10-5 – десятичная константа с мантиссой 12,3 и порядком -5);

– восьмеричные числа (идентифицируются латинской буквой «O» – от слова «octal» – восьмеричное);

– шестнадцатеричные числа (имеют в конце отличительный признак в виде буквы *h* или *H*, от слова «hexagonal» – шестнадцатеричное; если число начинается с буквы, то перед ней надо ввести ноль);

– комплексные числа $Z = \mathrm{Re}Z + i*\mathrm{Im}Z$, где ReZ – действительная часть комплексного числа Z, ImZ – его мнимая часть, а символ *i* обозначает мнимую единицу, т. е. $\sqrt{-1}$;

– системные константы, хранящие определенные параметры системы;

– строковые константы – любые цепочки символов, заключенные в кавычки. Арифметические выражения в строковых константах рассматриваются как текст и не вычисляются;

– единицы измерения физических величин.

3) Переменные

Тип переменной определяется ее значением. Переменные могут быть числовыми, строковыми, символьными и т. д., поэтому тип переменной предварительно не задается. Каждой переменной должно быть присвоено значение (т. е. переменная должна быть определена). Попытка использования неопределенной переменной ведет к выводу сообщения об ошибке.

Константы, переменные и иные объекты имеют имена (идентификаторы). Имена могут иметь практически любую длину, и в них могут входить любые латинские и греческие буквы, а также цифры и некоторые спецсимволы (например, _), но начинаться имя может только с буквы. Нельзя использовать в именах буквы русского языка. Малые и большие буквы различаются. Имена не могут совпадать с именами встроенных или определенных пользователем функций.

4) Операторы

Это элементы языка, предназначенные для создания математических выражений совместно с данными, именуемыми операндами. Это знаки арифметических операций, вычисления сумм, произведений, производной, интеграла и т. д.

5) Встроенные функции

Это функции определенные в самой системе и готовые к немедленному использованию (например, sin(x), atan(x), ln(x) и др.).

6) Функции пользователя

Это те функции, которые создаются самим пользователем. Благодаря этим функциям обеспечивается расширение входного языка Mathcad и его адаптация к специфическим задачам пользователя.

5.3.1. Ввод математических выражений и работа с формульным редактором

Операции вывода и присваивания

Для вычисления любого выражения достаточно установить после него оператор вывода (знак =). Оператор = можно использовать только для первого присваивания.

Для присвоения переменным новых значений используется оператор :=, который вводится своим первым символом – двоеточием (:).

Существует также «жирный» знак равенства, который используется в логических операциях сравнения

Использование оператора := обеспечивает локальное присваивание. С помощью знака ≡ можно выполнить глобальное присваивание, т. е. независимо от того, в каком месте документа расположен оператор глобального присваивания, переменная получает это значение.

Оператор умножения вводится звездочкой (*), но представляется точкой в середине строки.

Оператор деления вводится как косая черта (/), но заменяется горизонтальной чертой.

Оператор возведения в степень вводится знаком ∧, но число в степени представляется в обычном виде.

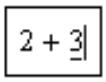
Для ввода десятичных чисел в качестве разделителя целой и дробной части используется точка.

По умолчанию десятичные числа имеют представление с тремя знаками после разделительной точки.

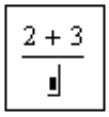
Mathcad идентифицирует наиболее распространенные константы, например, e – основание натурального логарифма.

5.4. Простые вычисления арифметических выражений и их редактирования

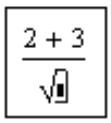
Рассмотрим пример на вычисление отношения суммы 2 + 3 к корню квадратному из числа 5. Вначале введем подряд символы 2 + 3. Получим



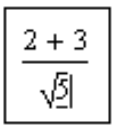
Введем знак деления. Для этого нажатием клавиши Пробел выделим все выражение 2 + 3, а затем нажмем клавишу со знаком /. Получим



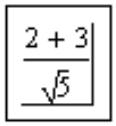
Введем знак квадратного корня. Для этого нажмем клавишу со знаком \. Получим



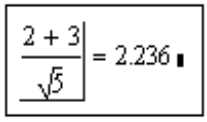
Введем подкоренное выражение. Для этого нажмем клавишу с цифрой 5. Получим



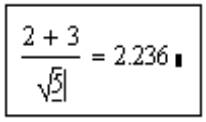
Выведем результат вычисления. С помощью пробела выделим все выражение,



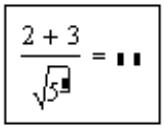
а затем в конце выражения поставим оператор вывода = . Получим



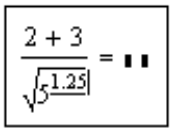
Рассмотрим случай, когда нужно изменить подкоренное выражение. Пусть подкоренное выражение будет число 5 в степени 1,25. Сначала аккуратно поместить курсор мыши после числа 5 и щелкнуть левой кнопкой. Получим



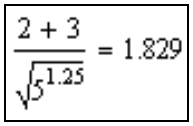
Поставим знак возведения в степень. Для этого на клавиатуре нажмем клавишу со значком ∧. Получим



Введем показатель степени числа 5, т. е. число 1,25. Получим



Получим новый результат. Для этого отведем курсор мыши от формульного блока на свободное поле документа и щелкнем левой кнопкой. Получим



Задача 1. Простые вычисления в Mathcad.

Функция 1 $f1(x) := \left|\sin(x)\right| \cdot \exp(x)$.

Функция 2 $f2(x) := 10 - x^{1.9}$.

Интервал аргумента $a := 0.5$, $b := 4$.

Число точек и шаг $N := 40$ $h := \frac{b-a}{N}$.

Цикл аргумента $n := 0 \ldots N$.

Задача 2.

Расчет функции 1 $y_n := f1(a+nh)$

Расчет функции 2 $z_n := f2(a+nh)$

y =

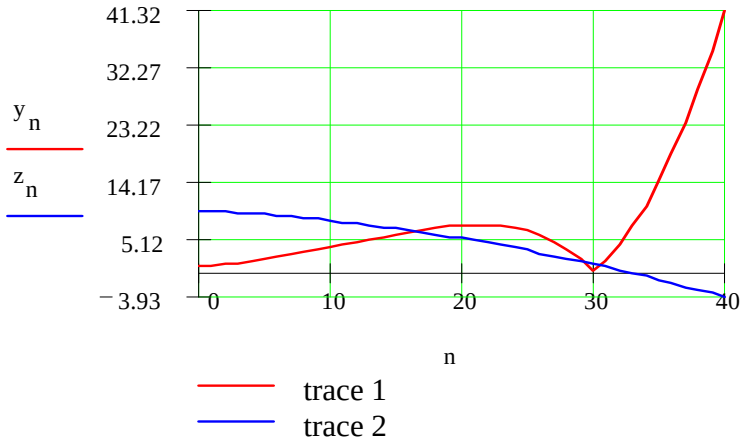
	0
0	0.79
1	0.997
2	1.227
3	1.481
4	1.758
5	2.058
6	2.382
7	2.728
8	3.094
9	3.479
10	3.88
11	4.291

z^T =

	0	1	2	3	4	5	6
0	9.732	9.636	9.526	9.403	9.266	9.115	8.952

Задача 3.

Графики



Ранжированные переменные имеют множественные значения. Если требуется задать ряд чисел с шагом, то ранжированная переменная записывается следующим образом:

$$X := X_1, X_2 .. X_{end},$$

где X_1, X_2, X_{end} – начальное, второе, конечное значение переменной X, шаг вычисляется системой как $X_2 - X_1$. Если значение X_2 не указано, то шаг принимается равным 1.

Пример

$i := 1..5$ $x := 2, 1.5..0$ $z := -0.5, -0.2..0.5$

i =

1
2
3
4
5

x =

2
1.5
1
0.5
0

z =

-0.5
-0.25
0
0.25
0.5

ORIGIN := 1

$f_i := i^2$ $\quad f = \begin{pmatrix} 1 \\ 4 \\ 9 \\ 16 \\ 25 \end{pmatrix}$ $\quad f_2 = 4$ $\quad f_3 = 9$

5.5. Работа с графикой

1. Построение двумерного графика одной функции:

– ввести функцию, набрав выражение sin(*x*)^3;

– на панели математических знаков щелкнуть на кнопке  ;

– на экране появится палитра графиков;

– в палитре графиков щелкнуть на кнопке  ;

– на экране появится шаблон графика;

– на оси *X* введите имя аргумента – *x*;

– на оси *Y* введите sin(*x*)^3;

– для завершения построения графика щелкнуть левой кнопкой мыши вне пределов графика (рис. 4.1).

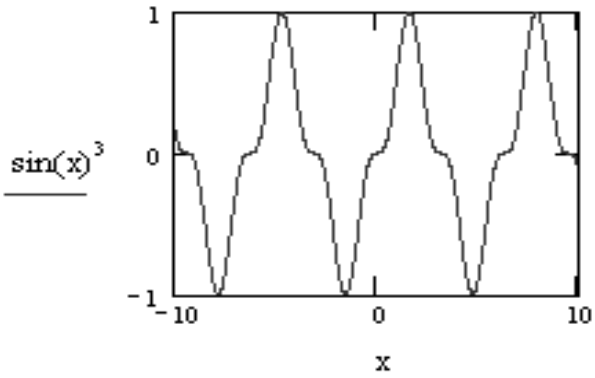


Рис. 4.1

2. Изменение размеров и перемещение графика:

– поместить курсор мыши в область графика и щелкнуть левой кнопкой (активизировать график) – вокруг графика появится черная рамка;

– подвести курсор мыши к маркеру изменения размера в правом нижнем углу рамки и, нажав левую кнопку, растянуть график по диагонали;

– навести курсор мыши на любую сторону рамки (кроме маркеров изменения размера), при этом курсор должен превратиться в черную ладошку;

– нажав левую кнопку мыши, передвинуть весь блок графика в желаемую область экрана. В результате этих действий получим увеличенный и перемещенный в другую область экрана график с обрамляющей его рамкой (см. рис. 4.1).

3. Построение графиков нескольких функций

Для примера рассмотрим построение графиков функций $\sin(x)^2$ и $\cos(x)$:

– щелкнуть левой кнопкой мыши точно в конце выражения $\sin(x)^3$;

– нажатием клавиши Пробел добиться, чтобы все выражение $\sin(x)^3$ было охвачено синим уголком, и поставить запятую;

– ввести выражение sin(*x*)^2;

– нажатием клавиши Пробел добиться, чтобы все выражение sin(*x*)2 было охвачено синим уголком, и поставить запятую;

– ввести выражение cos(*x*);

– для завершения построения графика щелкнуть левой кнопкой мыши вне пределов графика (рис. 4.2).

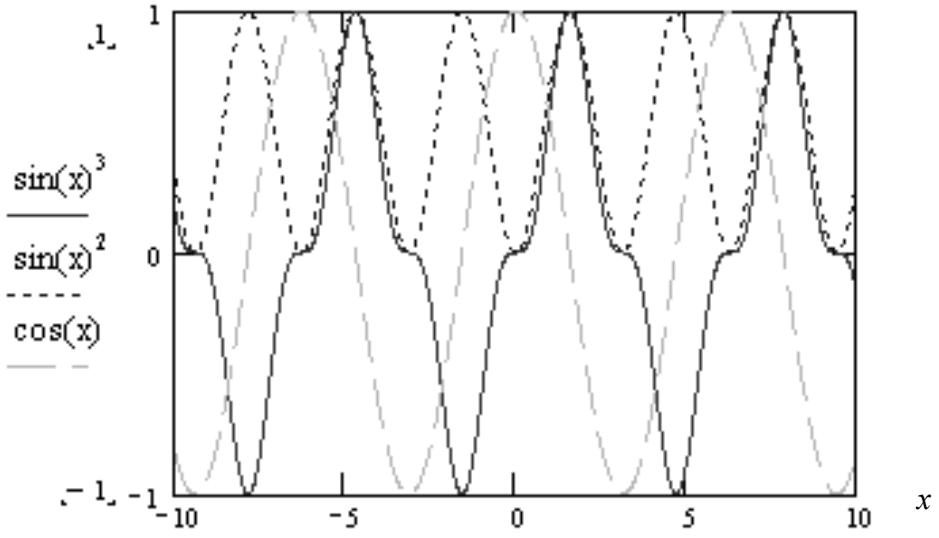


Рис. 4.2

4. Простейшие приемы форматирования графиков:

– графики можно перемещать по полю окна документа и изменять их размеры;

– изменять различные параметры графиков можно при помощи окна форматирования, которое появляется, если дважды щелкнуть мышью на графике, либо щелкнуть один раз, если график выделен;

– форматирование можно также осуществлять с помощью контекстного меню, которое появляется при щелчке на графике правой кнопкой мыши.

5. Построение графиков поверхностей

Пусть необходимо построить график функции $z(x, y) = x^2 + y^2$.

– определить функцию $z(x, y)$;

– ввести шаблон трехмерного графика  ;

– на единственное поле ввода под шаблоном ввести z;

– для завершения построения графика щелкнуть левой кнопкой мыши вне пределов графика (рис. 4.3).

6. Вращение трехмерного графика:

– построенную поверхность поместить с помощью контекстного меню графика в призму (*box*) (рис. 4.4);

– поместить курсор мыши в область графика, нажать левую кнопку и, удерживая ее, двигать мышь в том или ином направлении. Фигура вместе с осями координат и призмой, в которой она находится, будет вращаться.

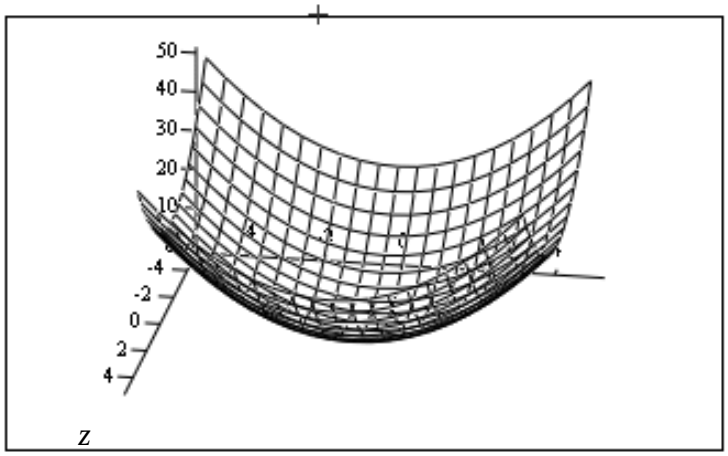


Рис. 4.3

Если при вращении фигуры удерживать нажатой клавишу `Ctrl`, можно удалять или приближать объект к наблюдателю. Если при вращении фигуры удерживать нажатой клавишу `Shift`, то после отпускания левой кнопки можно наблюдать анимированную картину вращения объекта в заданном предварительно направлении. Для остановки вращения надо щелкнуть левой кнопкой мыши.

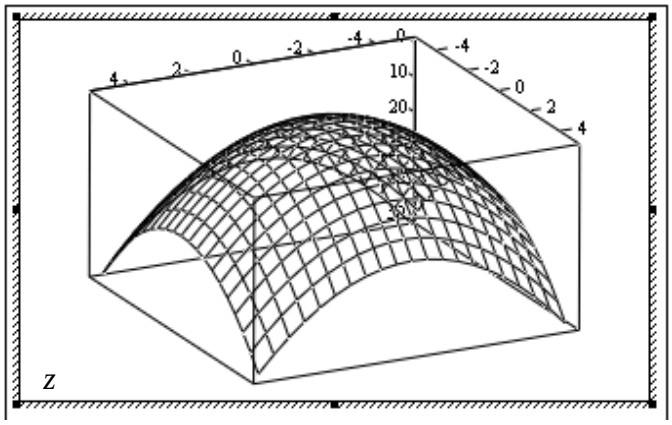


Рис. 4.4

7. Построение нескольких поверхностей на одном графике

Необходимо построить график двух трехмерных поверхностей. Пусть заданы две функции двух переменных z1(*x*,*y*) и z2(*x*,*y*):

z1(*x*,*y*):= $x^2 + y^2$ z2(*x*,*y*):= – ($x^2 + y^2$).

– определить ряд необходимых функций, описывающих поверхности;

– ввести через запятую имена этих функций в поле ввода шаблона трехмерного графика;

– для завершения построения графика щелкнуть левой кнопкой мыши вне пределов графика (рис. 4.5).

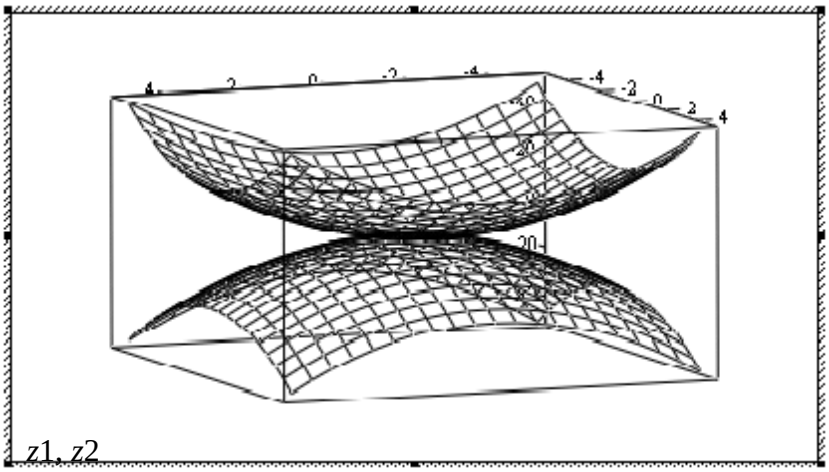


Рис. 4.5

Задача 1. Многомерные вычисления в Mathcad.

Функция $y(x,t) := if\left(x = 0{,}1, \frac{\sin(x)}{x}\right) \cdot if\left(t = 0{,}1, \frac{\sin(t)}{t}\right)$

Данные для циклов $N := 20$ $M := 20$ $h := \pi/5$.

Изменения аргумента $n := 0 \ldots N$.

Изменения аргумента: $m := 0 \ldots M$.

Задача 2. Расчет функции

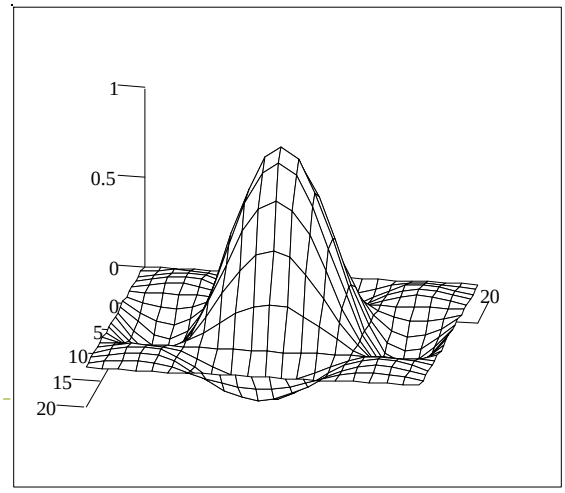
$$M_{n,m} := y\left[\left(n - \frac{N}{2}\right) \cdot h, \left(m - \frac{N}{2}\right) \cdot h\right]$$

M =

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	0.011	0.02	0.022	0.016	0	-0.024	-0.052
2	0	0.02	0.036	0.041	0.03	0	-0.044	-0.095
3	0	0.022	0.041	0.047	0.034	0	-0.051	-0.109
4	0	0.016	0.03	0.034	0.024	0	-0.036	-0.079
5	0	0	0	0	0	0	0	0
6	0	-0.024	-0.044	-0.051	-0.036	0	0.055	0.118
7	0	-0.052	-0.095	-0.109	-0.079	0	0.118	0.255
8	0	-0.079	-0.143	-0.164	-0.118	0	0.177	0.382
9	0	-0.097	-0.177	-0.202	-0.146	0	0.219	0.472
10	0	-0.104	-0.189	-0.216	-0.156	0	0.234	0.505
11	0	-0.097	-0.177	-0.202	-0.146	0	0.219	0.472
12	0	-0.079	-0.143	-0.164	-0.118	0	0.177	0.382
13	0	-0.052	-0.095	-0.109	-0.079	0	0.118	0.255

Задача 3.

Поверхностный график (рис. 4.6).

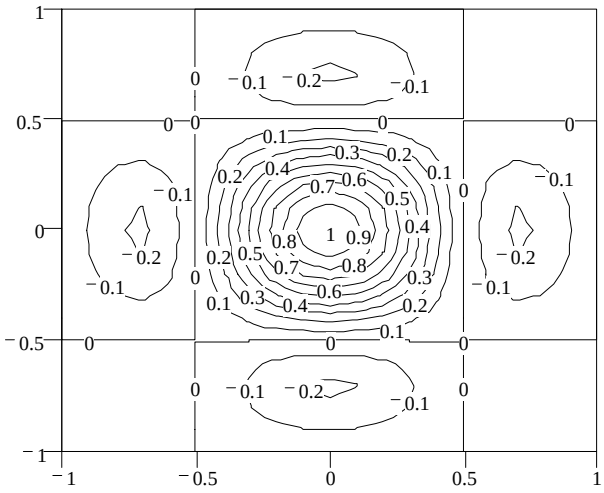


M

Рис. 4.6

Задача 4.

Контурный график (рис. 4.7).



M

Рис. 4.7

5.6. Работа с векторами и матрицами

1. Создание матриц и векторов

Команда `Matrices` (матрицы) меню `Insert` (Вставка) обеспечивает вывод шаблона для ветора или матрицы. Шаблон содержит поля ввода элементов вектора или матрицы. Поля ввода можно активизировать щелчком мыши. С помощью клавиш перемещения курсора можно ввести все элементы вектора или матрицы.

Каждый элемент матрицы характеризуется индексированной переменной, а его положение в матрице определяется двумя индексами – номе-

ром строки и номером столбца. Для задания индексированной переменной прежде всего надо ввести имя переменной, а затем перейти к набору индексов нажатием клавиши [. Сначала указывается индекс строки, а затем индекс столбца.

Вырожденная в одну строку или в один столбец матрица является вектором. Его элементы – индексированные переменные с одним индексом.

Нижняя граница индексов задается значением системной переменной `ORIGIN`. Обычно ее значение задают равным 0 или 1.

2. Векторные и матричные операторы

Примем следующие обозначения: V – вектор, M – матрица, Z – скалярная величина.

Оператор	**Клавиши**	**Описание**
V1+V2	V1+V2	Сложение двух векторов V1 и V2
V1-V2	V1-V2	Вычитание двух векторов V1 и V2
Оператор	**Клавиши**	**Описание**
-V	-V	Смена знака у элементов вектора V
-M	-M	Смена знака у элементов матрицы M
V-Z	V-Z	Вычитание из всех элементов вектора V скаляра Z
Z·V,V·Z	Z*V, V*Z	Умножение вектора V на скаляр Z
Z·M, M·Z	Z*M, M*Z	Умножение матрицы M на скаляр Z
V1·V2	V1*V2	Скалярное умножение двух векторов
M·V	M*V	Умножение матрицы на вектор
M1·M2	M1*M2	Умножение двух матриц
V/Z	V/Z	Деление всех элементов вектора V на скаляр Z
M/Z	M/Z	Деление матрицы M на скаляр Z
M^{-1}	M ^ -1	Обращение матрицы M
M^n	M ^ n	Возведение матрицы M в степень n
|V|	|V	Вычисление модуля вектора V
|M|	|M	Вычисление определителя матрицы M
V^T	V Ctrl !	Транспонирование вектора V
M^T	M Ctrl !	Транспонирование матрицы M
V1×V2	V1 Ctrl * V2	Векторное умножение двух векторов
ΣV	Alt $ V	Вычисление суммы элементов вектора V
$\vec{V}$	V Ctrl —	Векторизация вектора V
$\vec{M}$	M Ctrl —	Векторизация матрицы M
$M^{<n>}$	M Ctrl ^ n	Выделение n-ного столбца матрицы

V_n	V [n	Выделение n-ного элемента вектора
$M_{m,\,n}$	M [(m, n)	Выделение элемента (m, n) матрицы

Все представленные в таблице операторы могут вызываться из палитры матричных операций.

5.7. Решение уравнений

5.7.1. Решение одного уравнения

Начальное значение переменной $x := 3$.

Решаемое уравнение $x^3 - e^3 = 0$.

$$z := \text{root}\left(x^3 - e^x, x\right).$$

Найденное решение $z = 1.857$.

5.7.2. Нахождение корней полинома

Полином $x^3 - 10x + 2$.

Вектор коэффициентов, начинающийся с константы. Сюда должны войти все коэффициенты, в том числе равные нулю.

$$v := \begin{pmatrix} 2 \\ -10 \\ 0 \\ 1 \end{pmatrix}.$$

Найденное решение $\text{polyroots}(v) = \begin{pmatrix} -3.258 \\ 0.201 \\ 3.057 \end{pmatrix}$.

5.7.3. Решение систем линейных уравнений

1) метод обратной матрицы

Решить систему уравнений

$$2 \cdot x1 + 7 \cdot x2 + 3 \cdot x3 = 6$$
$$3 \cdot x1 + 5 \cdot x2 + 2 \cdot x3 = 4$$
$$9 \cdot x1 + 4 \cdot x2 + 10 \cdot x3 = 2$$

Матрица коэффициентов системы линейных уравнений

$$A := \begin{pmatrix} 2 & 7 & 3 \\ 3 & 5 & 2 \\ 9 & 4 & 10 \end{pmatrix}.$$

Вектор свободных членов $B := \begin{pmatrix} 6 \\ 4 \\ 2 \end{pmatrix}$.

Решение системы $X := A^{-1} \cdot B$.

Результаты решения $X = \begin{pmatrix} -1.182 \\ 0.909 \\ 0 \end{pmatrix}$.

2) с помощью функций Given и Find

Задание начальных условий: $x1 := 1,\ x2 := 1,\ x3 := 1.$

Ввод функции ***Given*** Given

$$2 \cdot x1 + 7 \cdot x2 + 3 \cdot x3 = 6$$

Ввод системы уравнений $3 \cdot x1 + 5 \cdot x2 + 2 \cdot x3 = 4$

$$9 \cdot x1 + 4 \cdot x2 + 10 \cdot x3 = 2$$

Получение результата с помощью функции ***Find***

$$\text{Find}(x1, x2, x3) = \begin{pmatrix} -1.182 \\ 0.909 \\ 0 \end{pmatrix}.$$

5.7.4. Решение нелинейных уравнений

1) с помощью функций Given и Find

Задание начальных условий: $x := 1.$

Ввод функции ***Given*** Given

Ввод уравнения $x^3 + 0.2 \cdot x^2 - 1.2 = 0$ z := Find(x).

Получение результата с помощью функции ***Find*** z = 1.

2) приближенное решение нелинейного уравнения с помощью функций Given и Minerr

Задание начальных условий: $x := 1.$

Ввод функции ***Given*** Given

Ввод уравнения $x^3 + 0.2 \cdot x^2 - 1.2 = 0$ z := Minerr(x).

Получение результата с помощью функции ***Minerr*** z = 1.

4.7.5. Приближенное решение системы нелинейных уравнений

Начальные значения переменных: $x := 0.$

Начало вычислительного блока Given

$$(x^2 + 1)^2 + (y^2 + 1)^2 = 5.5$$

Решаемая система уравнений $x + y = 0.95$

$$z := \text{Minerr}(x, y)$$

Найденное решение $z = \begin{pmatrix} -0.106 \\ 1.056 \end{pmatrix}$.

5.8. Выполнение символьных вычислений

Символьными называются такие вычисления, результаты которых представляются в аналитическом виде, то есть в виде формул. В частном случае результат может быть числом. Команды, относящиеся к работе символьного процессора, содержатся в меню Simbolics (Символика). Чтобы выполнить символьную операцию, надо выделить то выражение, над которым она производится. Для ряда операций следует не только указать выражение, к которому относится операция, но и наметить переменную, относительно которой выполняется символьная операция. Для выполнения символьной операции нужно выбрать меню Символика→Величина→Символически (Simbolics→Evaluate→Simbolically).

Символьные операции:

1. Опция Evaluate/Simbolically – вычислить символически.
2. Опция Evaluate/Floating Point – вычислить с плавающей точкой.
3. Опция Evaluate/Complex – вычислить в комплексной форме.
4. Опция Simplify – упростить.
5. Опция Expand – развернуть (разложить по степеням).
6. Опция Factor – факторизовать (разложить на множители).
7. Опция Collect – группировать по подвыражению.
8. Опция Polinomial Coefficients – полиномиальные коэффициенты.
9. Опция Variable/Solve – решить для переменной.
10. Опция Variable/Substitute – подстановка подвыражения переменной.
11. Опция Variable/Differentiate – дифференцировать по переменной.
12. Опция Variable/Integrate – интегрировать по переменной.
13. Опция Expand to Series – разложить в ряд Тейлора.
14. Опция Convert to Partial Fraction – преобразовать в элементарные дроби.
15. Операции с матрицами (транспонирование, обращение, определитель).
16. Преобразования Фурье, Лапласа, z-.
17. Опция Evaluation Style – стиль эволюции.
18. Система SmartMath – чудо-математика.

Рассмотрим примеры вычисление интеграла, дифференциала и предела:

Исходное выражение	Результат операции
$\frac{d}{dx}\sin(x)$	$\cos(x)$
$\int_a^b x^2 dx$	$\frac{1}{3}b^3 - \frac{1}{3}a^3$
$\lim_{x\to 0}(1+x)^{1/x}$	$\exp(1)$

Развёртка / свёртка

Пример 1.

Функция $(a+x)^3$.

Её развёртка $a^3 + 3a^2x + 3ax^2 + x^3$.

Её развёртка в SmartMath (значение a использовано)

$$(a+x)^3 \text{ expand} \rightarrow 8 + 12x + 6x^2 + x^3.$$

Пример 2.

Функция $a^3 + 3a^2x + 3ax^2 + x^3$

Её свёртка $(x+a)^3$.

Разложение в ряд Тейлора

Функция $\frac{\sin(x)}{x}$.

Её разложение в ряд по x $1 - \frac{1}{6}x^2 + \frac{1}{120}x^4 + O(x^5)$.

Отброшен остаточный член $F2(x) := 1 - \frac{1}{6}x^2 + \frac{1}{120}x^4$.

5.9. Нахождение экстремумов функций

Для нахождения экстремумов функций многих переменных существует две альтернативные возможности. Первая заключается в использовании блока ***given*** и функции ***minerr.*** Определим функцию двух переменных

$$G(x,y) := 25 - x^2 - y^2.$$

Зададимся целью найти ее экстремум в области x = [–5,5] y = [–5,5]. Оценим по графику положение экстремума. Заносим в матрицу **M** значения функции в узловых точках:

$$i := 0\ldots10,\quad j := 0\ldots10,\quad M_{(i,j)} := G(i-5, j-5).$$

На заданном интервале функция не превосходит 25 (рис. 4.8). Зададим начальные приближения для поиска экстремума:

$$x := 1,\quad y := 1.$$

Записываем блок уравнений или неравенств. Число уравнений и неравенств в блоке ***given – Find*** должно быть больше и равно числа искомых величин. Если уравнений и неравенств не хватает, то можно просто продублировать одно и то же уравнение или вписать какое–либо тождество, например, 2 = 2.

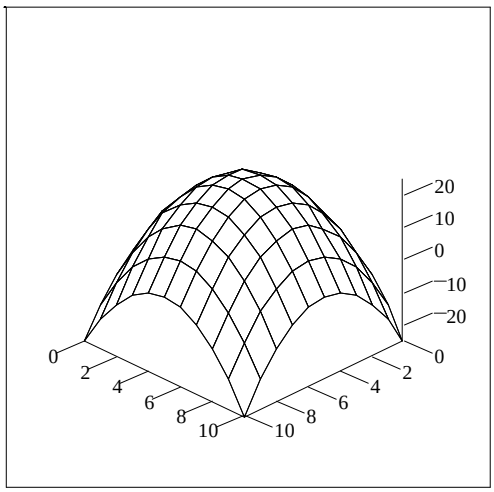
$$\text{Given}\quad G(x,y) = 26\quad x \geq -5\quad x \leq 5\quad y \geq -5\quad y \leq 5$$

$$\text{Minerr}(x,y) = \begin{pmatrix} 4.584\cdot10^{-6} \\ 4.02\cdot10^{-5} \end{pmatrix}.$$

Функция ***Minerr*** ищет приближенной решение для системы уравнений и неравенств, записанных в блоке. В данном случае мы получили, что системе уравнений наилучшим образом соответствует точка [0,0]. (Поскольку по умолчанию точность вычислений составляет 0.001, мы округлили результат до 0).

Из графика видно, что значение 26 больше самого большого значения функции в окрестностях точки [1,1], то есть точное решение найти нельзя и функция ***Minerr*** подбирает такое значение *x*, при котором функция ближе всего к значению 26.

Вторая возможность – поиск нулей первой производной, то есть стандартный математический подход. Для этого можно использовать блок ***given – Find.*** Функция ***Find*** ищет точное решение системы уравнений и неравенств, записанных после слова ***given***



M

Рис. 4.8

$$\text{Given}\quad \frac{d}{dx}G(x,y) = 0\quad \frac{d}{dy}G(x,y) = 0$$

$$\text{Find}(x,y) = \begin{pmatrix} 1.059\cdot10^{-7} \\ 1.059\cdot10^{-7} \end{pmatrix}.$$

Результаты, полученные различными методами, совпадают, время счета

мало в обоих случаях.

Дополнительная возможность поиска экстремумов с помощью функций ***Minimize*** и ***Maximize***, которые могут быть использованы как сами по себе, так и совместно с блоком Given. Аргументы функций: имя функции, экстремум которой ищется, и список ее аргументов.

Определяем функцию двух переменных $f(x,y) := 25 - x2 - y2$ и задаем начальные приближения $x := 1, \quad y := 1$.

Задаем область поиска максимума внутри блока ***Given***:

$$\text{Given} \quad 0 \le x \le 5 \quad 0 \le y \le 5.$$

Находим максимум функции в заданной области:

$$P := \text{Maximize}(f, x, y) \qquad P = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

В случае функции одной переменной задаем функцию $g(x) := x^4 - x^2$ и начальное приближение $x := 1$.

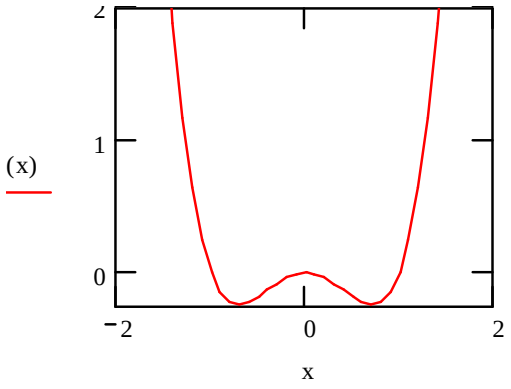


Рис. 4.9

Находим максимум Maximize $(g,x) = 0$.

Для нахождения максимального или минимального значения функции на некотором интервале, то необходимо определить этот интервал в блоке Given:

$$\text{Given} \quad 0 < x \le 2 \quad \text{Maximize}(g,x) = 2$$

На приведенном графике (рис. 4.9) видно, что первый из найденных максимумов соответствовал случаю, когда производная обращается в ноль; второй максимум лежит на границе интервала $x := -2, -1.9..2$.

5.10. Аппроксимация функций

Аппроксимацией (приближением) функции $f(x)$ называется нахождение такой функции $g(x)$ (***аппроксимирующей функции***), которая была бы близка заданной. Критерии близости функций $f(x)$ и $g(x)$ могут быть различные. В том случае, когда приближение строится на дискретном наборе точек, аппроксимацию называют ***точечной*** или ***дискретной.***

В том случае, когда аппроксимация проводится на непрерывном множестве точек (отрезке), она называется ***непрерывной*** или ***интегральной***. Примером такой аппроксимации может служить разложение функции в ряд Тейлора, т. е. замена некоторой функции степенным многочленом.

Наиболее часто встречающимся видом точечной аппроксимации является ***интерполяция*** (в широком смысле).

Пусть задан дискретный набор точек x_i $(i = 0,1,...,n)$, называемых ***узлами интерполяции***, причем среди этих точек нет совпадающих, а также значения функции y_i в этих точках. Требуется построить функцию $g(x)$, проходящую через все заданные узлы. Таким образом, критерием близости функции является $g(x_i) = y_i$.

В качестве функции $g(x)$ обычно выбирается полином, который называют ***интерполяционным полиномом***.

В том случае, когда полином един для всей области интерполяции, говорят, что интерполяция ***глобальная***.

В тех случаях, когда между различными узлами полиномы различны, говорят о ***кусочной*** или ***локальной интерполяции***.

Найдя интерполяционный полином, можно вычислить значения функции $f(x)$ между узлами (провести ***интерполяцию в узком смысле слова***), а также определить значение функции $f(x)$ даже за пределами заданного интервала (провести ***экстраполяцию***). Следует иметь в виду, что точность экстраполяции обычно очень невелика.

5.10.1. Локальная интерполяция. Линейная интерполяция

При линейной интерполяции Mathcad соединяет существующие точки данных прямыми линиями. Это выполняют функцией *linterp*, описанной ниже.

Функция *linterp(vx,vy,x)* – использует векторы данных *VX* и *VY*, чтобы возвратить линейно интерполируемое значение *y*, соответствующее третьему аргументу *x*. Аргументы *VX* и *VY* должны быть векторами одинаковой длины. Вектор *VX* должен содержать вещественные значения, расположенные в порядке возрастания.

Эта функция соединяет точки данных отрезками, создавая таким образом ломаную. Интерполируемое значение для конкретного *x* есть ордината *y* соответствующей точки ломаной. Для значений *x*, расположенных перед первой точкой в *VX*, Mathcad продолжает ломаную прямой линией, проходящей через первые две точки данных. Для значений *x*, расположен-

ных за последней точкой в *VX*, Mathcad продолжает ломаную прямой линией, проходящей через последние две точки данных.

Для получения наилучших результатов значение *x* должно находиться между самым большим и самым маленьким значениями в векторе *VX*.

Пример:

Функция задана таблично. Найти значения этой функции приуказанных значения аргумента *x* (рис. 4.10): $x = 0{,}512$; $x = 0{,}535$.

$$VX := \begin{pmatrix} 0.51 \\ 0.52 \\ 0.53 \\ 0.54 \\ 0.55 \\ 0.56 \\ 0.57 \end{pmatrix} \qquad VY := \begin{pmatrix} 1.6651 \\ 1.6820 \\ 1.6989 \\ 1.7160 \\ 1.7333 \\ 1.7507 \\ 1.7683 \end{pmatrix}$$

linterp(*VX*,*VY*,0,512) = 1,668 linterp(*VX*,*VY*,0,535) = 1,707

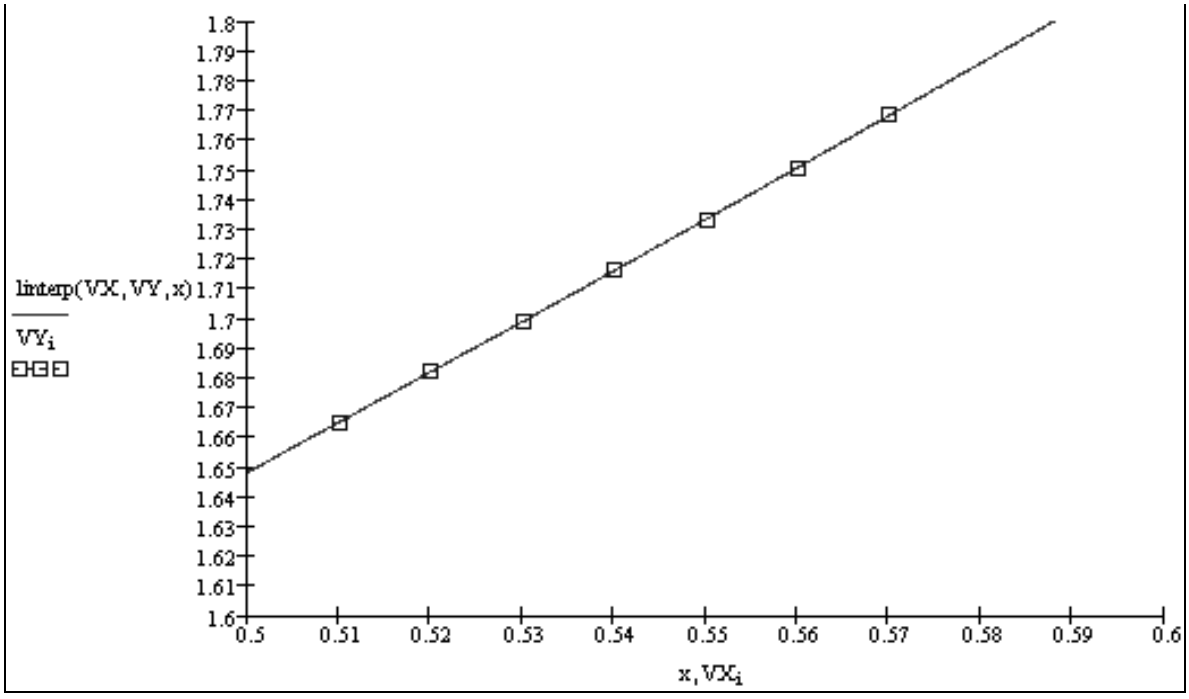


Рис. 4.10

5.10.2. Кубическая сплайн-интерполяция

Кубическая сплайн-интерполяция позволяет провести кривую через набор точек таким образом, что первые и вторые производные кривой не-

прерывны в каждой точке. Эта кривая образуется путем создания ряда кубических полиномов, проходящих через наборы из трех смежных точек. Кубические полиномы потом состыковываются друг с другом, чтобы образовать одну кривую.

Чтобы провести кубический сплайн через набор точек, нужно:

– создать векторы *VX* и *VY*, содержащие координаты *x* и *y*, через которые нужно провести кубический сплайн. Элементы *VX* должны быть расположены в порядке возрастания;

– вычислить вектор *VS: = cspline(VX, VY)*; вектор *VS* содержит вторые производные интерполяционной кривой в рассматриваемых точках;

– чтобы найти интерполируемое значение в произвольной точке, скажем *X0*, вычислить *interp*(*VS, VX, VY, X0*), где *VS, VX, VY* – векторы, описанные ранее.

Можно сделать то же самое, вычисляя *interp(cspline (VX, VY), VX, VY, X0).*

Пример:

Функция задана таблично. Найти значения этой функции приуказанных значения аргумента *x* (рис. 4.11): $x = 0{,}512$; $x = 0{,}535$.

$$VX := \begin{pmatrix} 0.51 \\ 0.52 \\ 0.53 \\ 0.54 \\ 0.55 \\ 0.56 \\ 0.57 \end{pmatrix} \qquad VY := \begin{pmatrix} 1.6651 \\ 1.6820 \\ 1.6989 \\ 1.7160 \\ 1.7333 \\ 1.7507 \\ 1.7683 \end{pmatrix} \qquad VS := cspline(VX, VY)$$

+

linterp(*VS*,*VX*,*VY*,0,512) = 1,668 linterp(*VS*,*VX*,*VY*,0,535) = 1,707

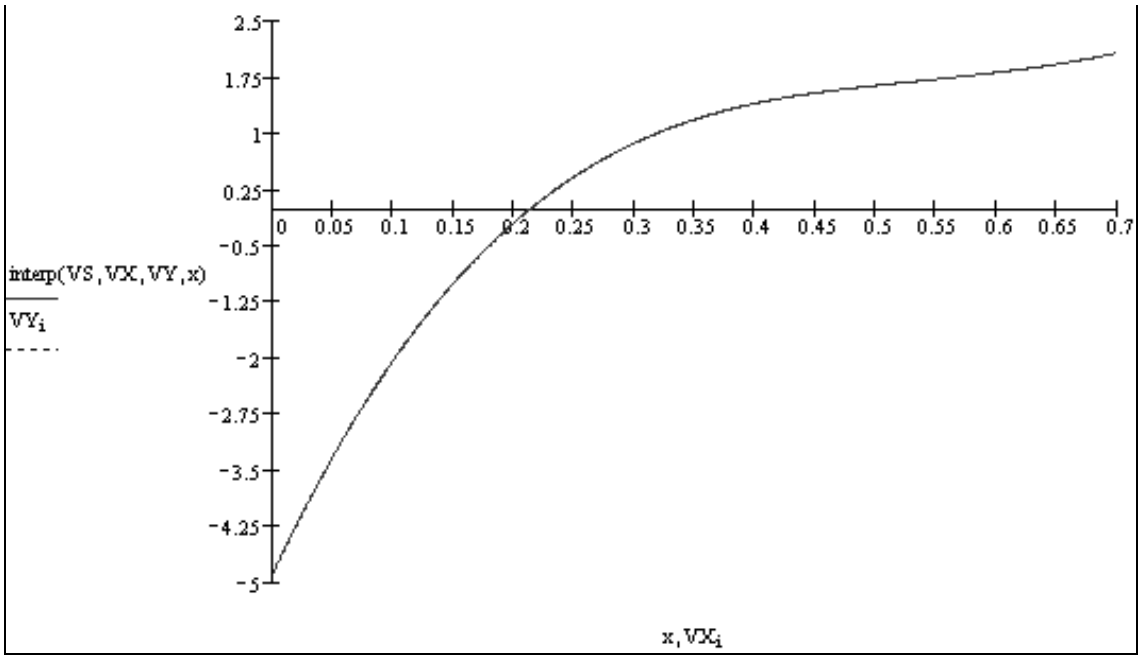


Рис. 4.11

В дополнение к *cspline* в MathCAD имеются две другие кубические сплайн-функции: *pspline(VX, VY)*, *lspline(VX, VY)*. Все эти сплайн-функции возвращают вектор коэффициентов вторых производных, который называется *VS*. Он используется в функции *interp*. Аргументы *VX* и *VY* должны быть веществennозначными векторами одинаковой длины. Значения в *VX* должны быть вещественны и расположены в порядке возрастания.

Функция *lspline* генерирует кривую сплайна, которая приближается к параболе в граничных точках.

Функция *pspline* генерирует кривую сплайна, которая приближается к параболе в граничных точках.

Функция генерирует кривую сплайна, которая может быть кубическим полиномом в граничных точках.

Функция *interp*(*VS*, *VX*, *VY*, *X0*) – возвращает интерполируемое значение *y*, соответствующее аргументу *x*. Вектор *VS* вычисляется на основе векторов данных *VX*, *VY* одной из функций *lspline*, *pspline* или *cspline*.

Интерполируемое значение для конкретного *x* есть ордината *y* соответствующей точки сплайна.

5.10.3. Глобальная интерполяция

При глобальной интерполяции ищется единый полином для всего интервала. Если среди узлов $\{x_i, y_i\}$ нет совпадающих, то такой полином

будет единственным, и его степень не будет превышать n. Запишем систему уравнений для определения коэффициентов полинома:

$$c_0 + c_1 x_0 + c_2 x_0^2 + \ldots + c_{n-1} x_0^{n-1} = y_0,$$
$$c_0 + c_1 x_1 + c_2 x_1^2 + \ldots + c_{n-1} x_1^{n-1} = y_1,$$
$$\vdots$$
$$c_0 + c_1 x_1 + c_2 x_{n-1}^2 + \ldots + c_{n-1} x_{n-1}^{n-1} = y_{n-1}.$$

Определим матрицу коэффициентов системы уравнений

$$i := 0..n-1 \quad j := 0..n-1 \quad a_{(j,i)} := (x_j)^i.$$

Решим систему уравнений матричным методом $c := a^{-1} y$.

Определим интерполяционный полином $L(z) := \sum_i c_i z^i$.

Представим результаты на графике (рис. 4.12).

Вычислим значения интерполяционного полинома в заданных точках и сравним их с точными значениями:

X_k	$L(X_k)$	$\sin(X_k)$
-0.5	-0.594	-0.479
1.111	0.901	0.896
2.333	0.72	0.723
4.574	-1.036	-0.99

Коэффициенты интерполяционного полинома следующие: $c = \begin{bmatrix} 0 \\ 1.134 \\ -0.177 \\ -0.127 \\ 0.022 \end{bmatrix}$.

Внимание! Из-за накопления вычислительной погрешности (ошибок округления) при большом числе узлов ($n > 10$) возможно резкое ухудшение результатов интерполяции. Кроме того, для целого ряда функций глобальная интерполяция полиномом вообще не дает удовлетворительного результата. Рассмотрим в качестве примера две таких функции. Для этих функций точность интерполяции с ростом числа узлов не увеличивается, а уменьшается. Первым примером является функция $f(x) := \frac{1}{1 + 25x^2}$.

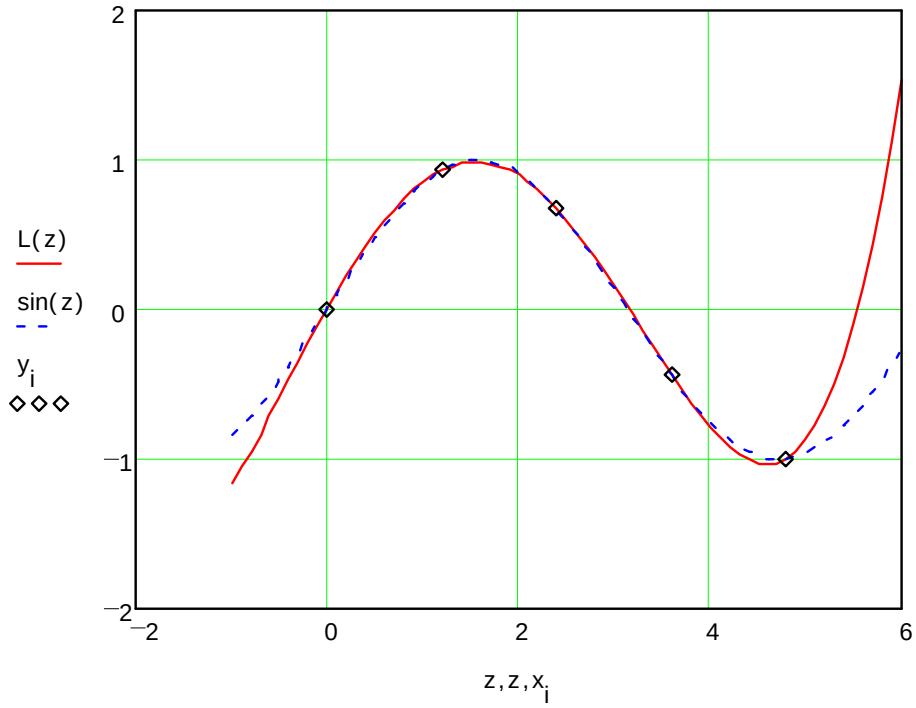


Рис. 4.12

Построим для нее интерполяционный полином на интервале [–1;1], используя 9 точек: $n := 9 \quad i := 0..n-1 \quad j := 0..n-1$.

$$x_j := -1 + \frac{j}{n-1} \cdot 2 .$$

$$a_{(j,i)} := (x_j)^i \quad y_j := f(x_j) \quad c := a^{-1} \cdot y \quad L(z) := \sum_i c_i z^i .$$

Представим результаты на графике (рис. 4.13) z := – 1, –0.99..1.

Второй пример – функция $g(x) := |x|$ (рис. 4.14). Найдем интерполяционный полином, используя заданные точки.

$$y_j := g(x_j) \quad c := a^{-1} \cdot y \quad L(z) := \sum_i c_i z^i .$$

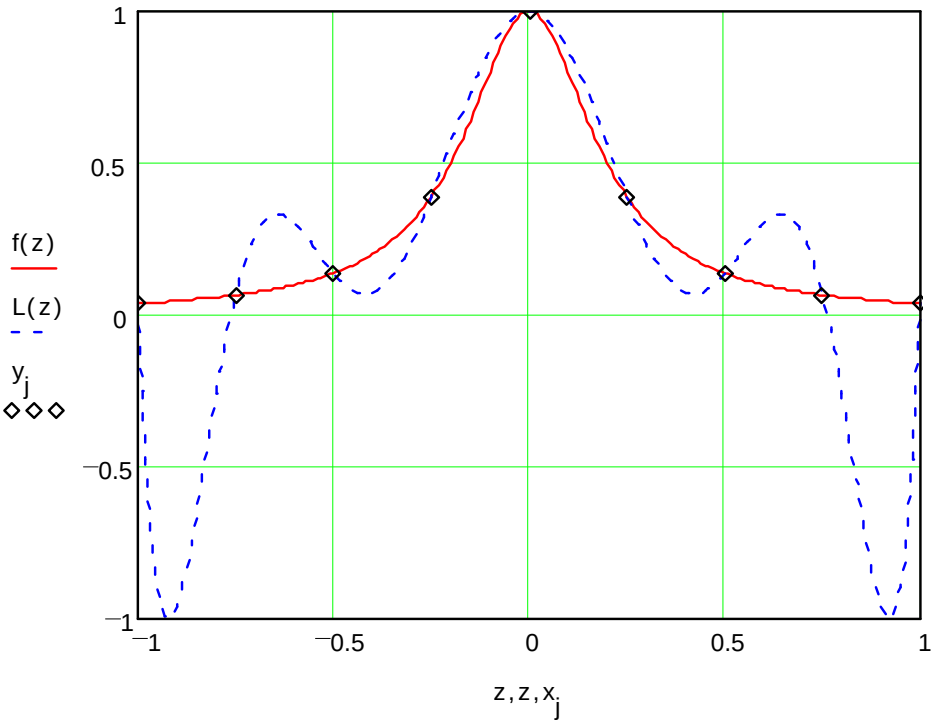


Рис. 4.13

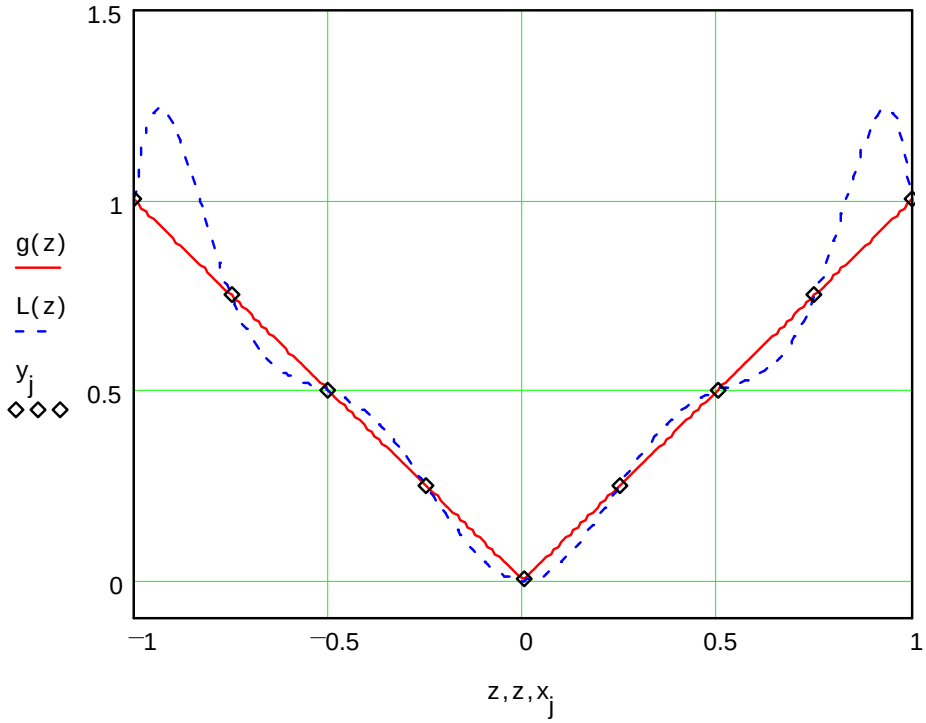


Рис. 4.14

Убедитесь самостоятельно, что при увеличении числа узлов интерполяции, результаты интерполирования вблизи концов интервала ухудшаются.

5.10.4. Метод наименьших квадратов

Наиболее распространенным методом аппроксимации экспериментальных данных является метод наименьших квадратов. Метод позволяет использовать аппроксимирующие функции произвольного вида и относится к группе глобальных методов. Простейшим вариантом метода наименьших квадратов является аппроксимация прямой линией (полиномом первой степени). Этот вариант метода наименьших квадратов носит также название линейной регрессии.

Критерием близости в методе наименьших квадратов является требование минимальности суммы квадратов отклонений от аппроксимирующей функции до экспериментальных точек:

$$\Phi = \sum_{i=1}^{n} \left(y_i - f(x_i) \right)^2 \to \min .$$

5.10.5. Аппроксимация линейной функцией

Применим метод наименьших квадратов для аппроксимации экспериментальных данных.

Читаем данные из файлов *datax* и *datay*:

x := READPRN(data$x$) y := READPRN(data$y$).

Имя файла следует заключать в кавычки и записывать его по правилам MS DOS, например, READPRN("c:\mylib\datax.prn").

Определяем количество прочитанных данных (число экспериментальных точек): n := last(x) i := 0…n.

Используем встроенные функции ***slope*** и ***intercept*** для определения коэффициентов линейной регрессии (аппроксимация данных прямой линией). Функция ***slope*** определяет угловой коэффициент прямой, а функция ***intercept*** – точку пересечения графика с вертикальной осью:

A := intercept (x, y), B := slope (x, y).

Определяем аппроксимирующую функцию: $f1(z) := A + B$.

Коэффициенты линейной регрессии – A = – 3.539, B = 1.81.

Mathcad 2000 предлагает для этих же целей использовать функцию ***line***: $\text{line}(x, y) = \begin{pmatrix} -3.539 \\ 1.81 \end{pmatrix}$.

Вычислим стандартное отклонение (рис. 4.15).

$$S1 := \sqrt{\left[\frac{1}{(n-2)} \cdot \sum_i \left(f1(x_i) - y_i \right)^2 \right]}$$ S1 := 2.093 z := 0, 0.1..10 .

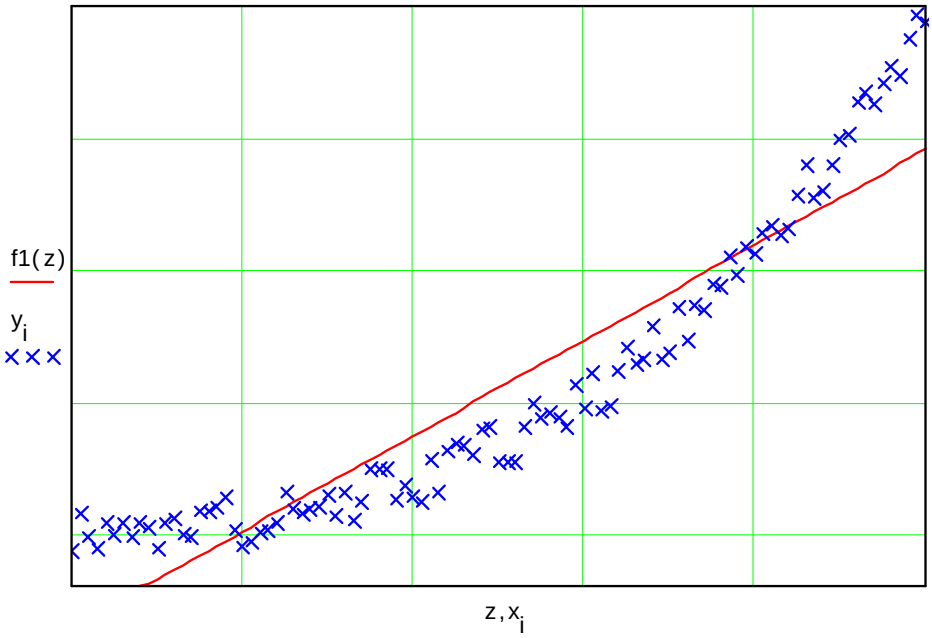


Рис. 4.15

5.10.6. 4.10.6. Аппроксимация полиномами

Теперь попытаемся подобрать полиномы второй и третьей степени, в качестве аппроксимирующей функции. Для этих целей служат встроенные функции ***regress*** и ***interp***. Очевидно, что если в качестве аппроксимирующей функции брать полином степени на единицу меньше числа точек, то задача сведется к задаче глобальной интерполяции и полученный полином будет точно проходить через все заданные узлы.

Вводим степени полиномов: $k2 := 2, \quad k3 := 3.$

Функция ***regress*** является вспомогательной, она подготавливает данные, необходимые для работы функции ***interp***. Вектор ***vs*** содержит, в том числе, и коэффициенты полинома: $vs2 := \text{regress}(x,y,k2)$, $vs3 := \text{regress}(x,y,k3)$.

Функция ***interp*** возвращает значение полинома в точке ***z***. Определив новые функции $f2$, $f3$, получаем возможность находить значение полинома в любой заданной точке:

$$f2(z) := \text{interp}(vs2,x,y,z) \quad f3(z) := \text{interp}(vs3,x,y,z)$$

$$\text{coeffs2} := \text{submatrix}(vs2,3,\text{length}(vs2) - 1,0,0)$$

$$\text{coeffs3} := \text{submatrix}(vs3,3,\text{length}(vs3) - 1,0,0).$$

Коэффициенты:

$$(\text{coeffs2})^T = (0.701 \quad -0.76 \quad 0.257),$$

$$(\text{coeffs3})^T = (-0.122 \quad 0.253 \quad 2.377 \cdot 10^{-3} \quad 0.017),$$

$$S2 := \sqrt{\left[\frac{1}{(n-2)} \cdot \sum_i \left(f2(x_i) - y_i\right)^2\right]} \quad S2 := 0.67,$$

$$S3 := \sqrt{\left[\frac{1}{(n-2)} \cdot \sum_i \left(f3(x_i) - y_i\right)^2\right]} \quad S3 := 0.58.$$

Стандартные отклонения почти не отличают друг от друга, коэффициент при четвертой степени z невелик, поэтому дальнейшее увеличение степени полинома нецелесообразно и достаточно ограничиться только второй степенью.

К сожалению функция ***regress*** имеется далеко не во всех версиях Mathcad. Однако, провести полиномиальную регрессию можно и без использования этой функции. Для этого нужно определить коэффициенты нормальной системы и решить полученную систему уравнений, например, матричным методом.

Теперь попытаемся аппроксимировать экспериментальные данные полиномами степени m и $m1$, не прибегая к помощи встроенной функции ***regress***.

$$m := 2 \qquad t := 0..m \qquad j := 0..m,$$
$$m1 := 3 \quad t1 := 0..m1 \quad j1 := 0..m1.$$

Вычисляем элементы матрицы коэффициентов нормальной системы

$$p_{t,j} := \sum_i (x_i)^{t+j}, \quad p1_{t1,j1} := \sum_i (x_i)^{t1+j1}$$

и столбец свободных членов

$$b_j := \sum_i y_i (x_i)^j, \quad b1_{j1} := \sum_i y_i (x_i)^{j1}$$

Находим коэффициенты полинома, решая систему матричным методом,

$$a := p^{-1} \cdot b, \qquad a1 := p1^{-1} \cdot b1.$$

Определяем аппроксимирующие функции (рис. 4.16, 4.17):

$$f2(z) := \sum_t (a_t) \cdot z^t, \qquad f3(z) := \sum_{t1} (a1_{t1}) \cdot z^{t1}.$$

Коэффициенты полиномов следующие:

$$a = \begin{pmatrix} 0.701 \\ -0.76 \\ 0.257 \end{pmatrix}, \qquad a1 = \begin{bmatrix} -0.122 \\ 0.253 \\ 2.377 \cdot 10^{-3} \\ 0.017 \end{bmatrix}.$$

Вычислим стандартное отклонение:

$$S2 := \sqrt{\left[\frac{1}{(n-2)} \cdot \sum_i \left(f2(x_i) - y_i\right)^2\right]} \quad S2 := 0.671,$$

$$S3 := \sqrt{\left[\frac{1}{(n-2)} \cdot \sum_i \left(f3(x_i) - y_i\right)^2\right]} \quad S3 := 0.581.$$

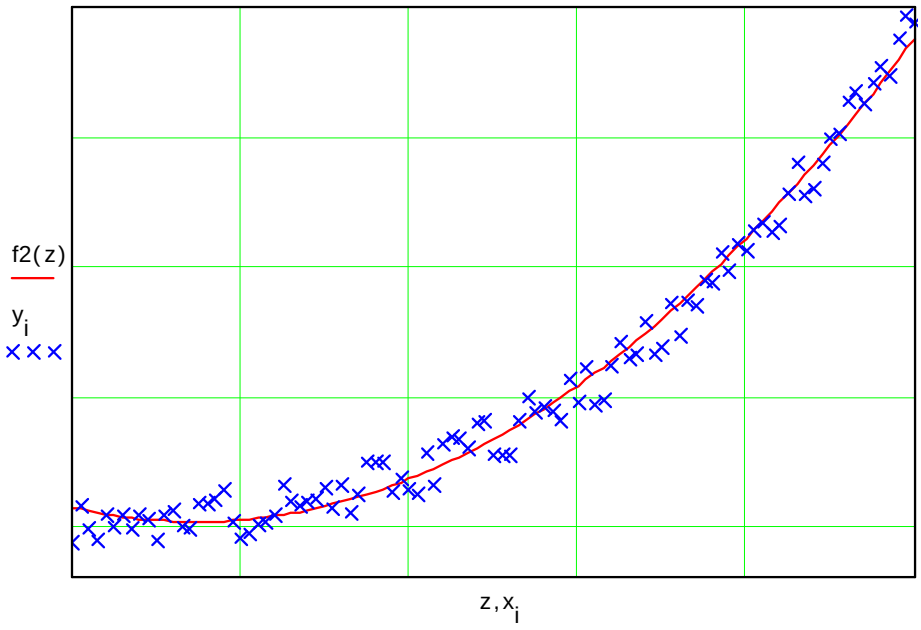


Рис. 4.16

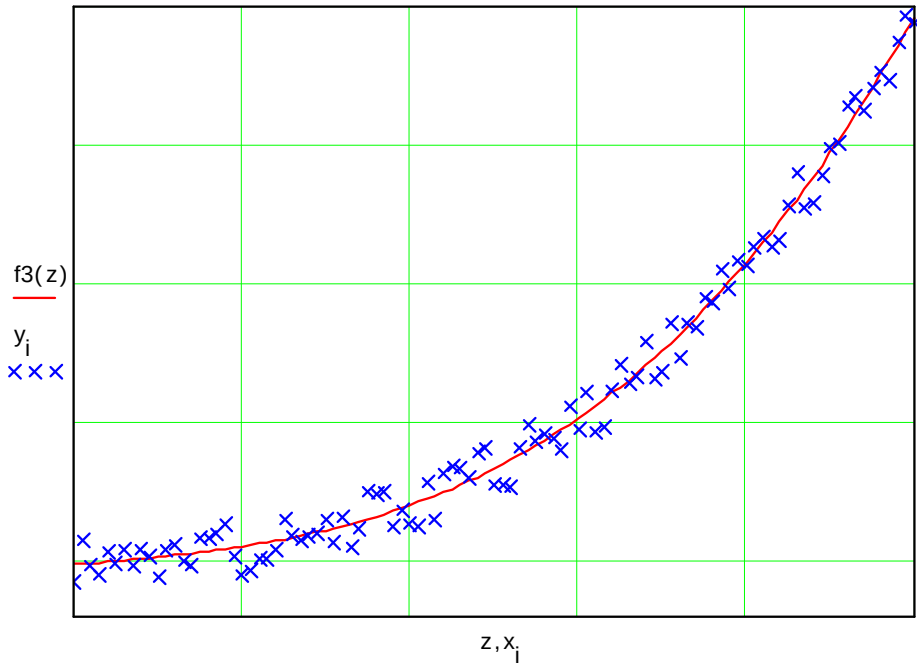


Рис. 4.17

5.10.7. Аппроксимация линейной комбинацией функций

Mathcad предоставляет пользователям встроенную функцию ***linfit*** для аппроксимации данных по методу наименьших квадратов линейной комбинацией произвольных функций.

Функция ***linfit*** имеет три аргумента:

– вектор ***x*** – *x*-координаты заданных точек,

– вектор ***y*** – *y*-координаты заданных точек,

– функция ***F*** – содержит набор функций, который будет использоваться для построения линейной комбинации.

Задаем функцию F (аппроксимирующая функция ищется в виде $a\frac{1}{x+1} + b\cdot x^2$:

$$F(x) := \begin{pmatrix} \frac{1}{x+1} \\ x^2 \end{pmatrix}, \quad S := \text{linfit}(x, y, F), \quad S = \begin{pmatrix} -0.802 \\ 0.176 \end{pmatrix}.$$

Определяем аппроксимирующую функцию (рис. 4.18): $f4(x) := F(x) \cdot S$.

Вычисляем дисперсию:

$$S4 := \sqrt{\left[\frac{1}{(n-2)}\sum_i (f4(x_i) - y_i)^2\right]}, \quad S4 = 0.975.$$

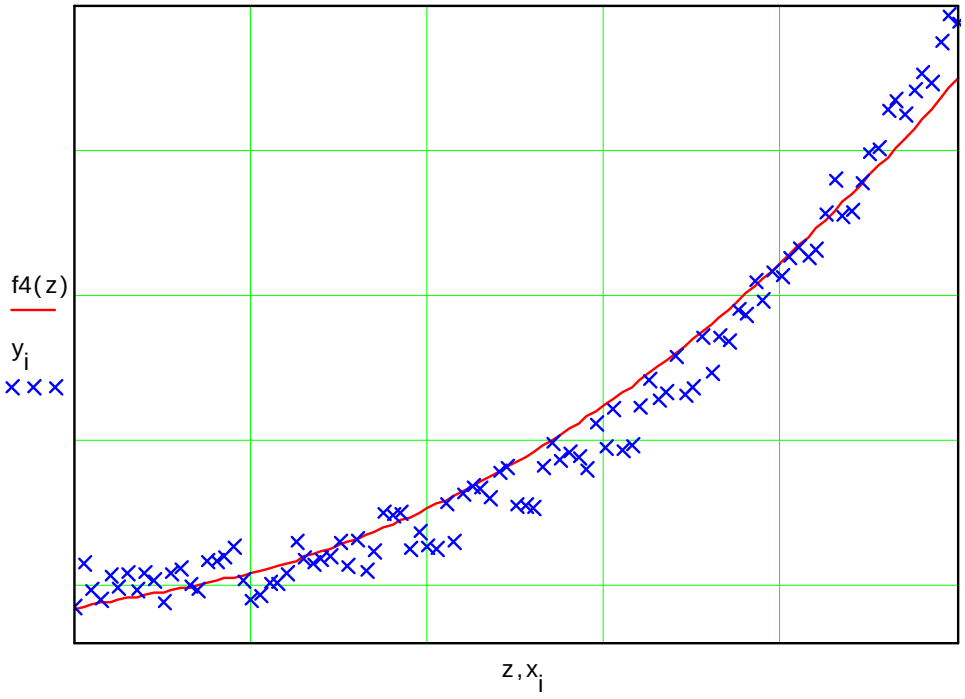


Рис. 4.18

5.10.8. Аппроксимация функцией произвольного вида

Теперь построим аппроксимирующую функцию дробно-рационального типа $\mathrm{f}(x)=\dfrac{ax^2}{b+x}$. Для этого воспользуемся функцией ***genfit***. Функция имеет следующие параметры:

– ***x, y*** – векторы, содержащие координаты заданных точек;

– ***F*** – функция, задающая искомую функциональную *n*-параметрическую зависимость и частные производные этой зависимости по параметрам;

– ***v*** – вектор, задающий начальные приближения для поиска параметров.

$$F(z,u):=\begin{bmatrix}\dfrac{u_0\cdot z^2}{u_1+z}\\ \dfrac{z^2}{u_1+z}\\ \dfrac{u_0\cdot z^2}{(u_1+z)^2}\end{bmatrix},\quad v:=\begin{pmatrix}-1\\-15\end{pmatrix},\quad S:=\mathrm{genfit}(x,y,v,F).\quad S=\begin{pmatrix}-2.146\\-20.85\end{pmatrix}$$

Поскольку нулевой элемент функции *F* содержит искомую функцию, определяем функцию следующим образом (рис. 4.19): $f5(z):=F(z,S)_0$.

Вычисляем среднее квадратичное отклонение:

$$S5:=\sqrt{\left[\frac{1}{(n-2)}\sum_i\left(f5(x_i)-y_i\right)^2\right]},\quad S5=0.581.$$

Функция ***genfit*** имеется не во всех реализациях Mathcad. Возможно, однако, решить задачу, проведя линеаризацию.

Заданная функциональная зависимость может быть линеаризована введением переменных $z=\dfrac{1}{y}$ и $t=\dfrac{1}{x}$. Тогда $z=\dfrac{1}{a}+b\dfrac{t}{a}$.

Определим матрицы коэффициентов нормальной системы [8]:

$$\mathrm{e}:=\begin{bmatrix}\sum_i\left[(x_i)^3\cdot y_i\right]\\ \sum_i\left[x_i\cdot(y_i)^2\right]\end{bmatrix}\qquad \mathrm{o}:=\begin{bmatrix}\sum_i(x_i)^4-\left[\sum_i(x_i)^2\cdot y_i\right]\\ \sum_i\left[(x_i)^2\cdot y_i\right]-\left[\sum_i(y_i)^2\right]\end{bmatrix}$$

Находим коэффициенты функции, решая систему матричным методом,

$$d := o^{-1} e \qquad d = \begin{pmatrix} -1.218 \\ -15.517 \end{pmatrix}.$$

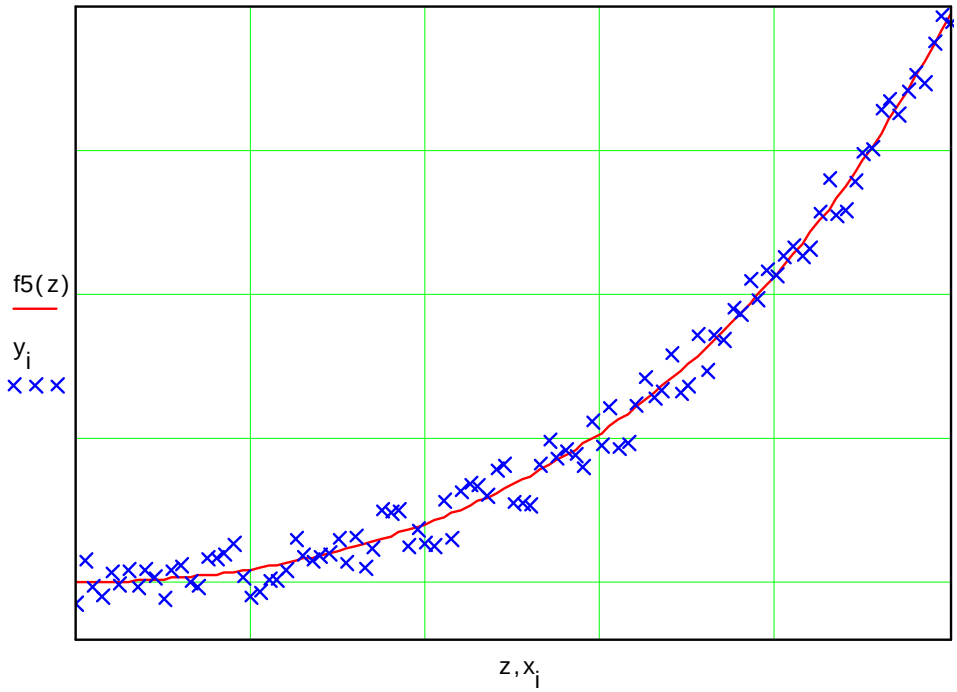


Рис. 4.19

Определяем функцию (рис. 4.20): $f5(z) := \dfrac{z^2 d_0}{d_1 + z}$.

Вычислим стандартное отклонение:

$$S5 := \sqrt{\left[\frac{1}{(n-2)} \sum_i (f5(x_i) - y_i)^2\right]}, \quad S5 = 0.827.$$

Обратите внимание! Мы получили другие коэффициенты! Вспомните, задача на нахождение минимума нелинейной функции, особенно нескольких переменных, может иметь несколько решений. Стандартное отклонение больше, чем в случае аппроксимации полиномами, поэтому следует остановить свой выбор на аппроксимации полиномом. Представим результаты аппроксимации на графике (см. рис. 4.20).

В тех случаях, когда функциональная зависимость оказывается достаточно сложной, может оказаться, что самый простой способ нахождения коэффициентов – минимизация функционала Ф.

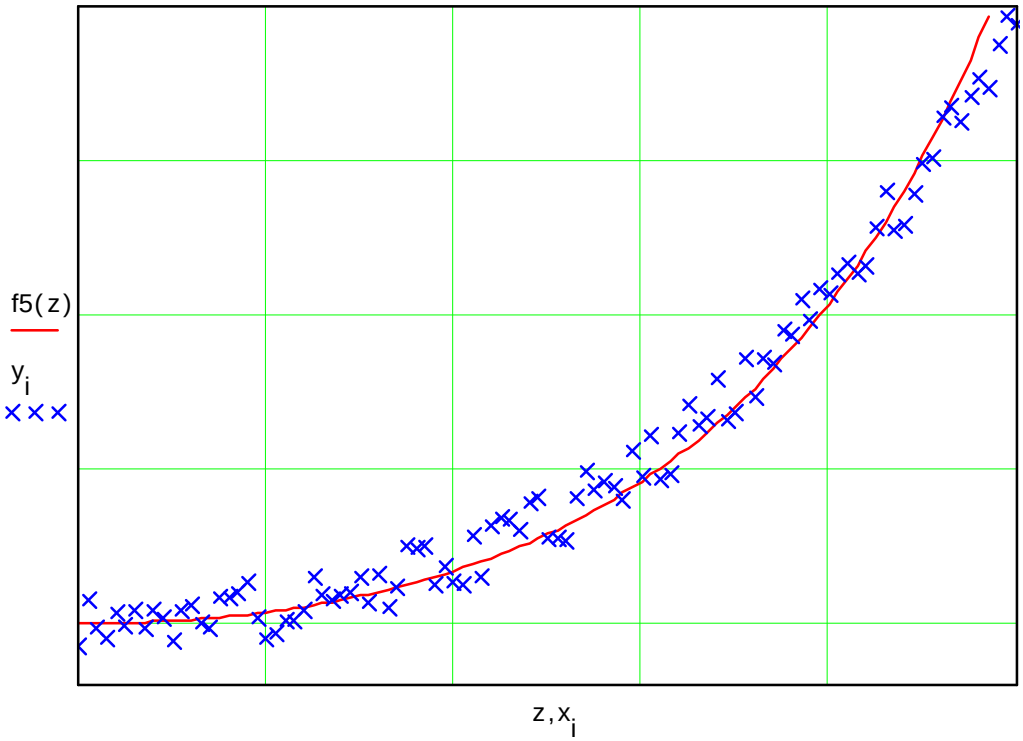


Рис. 4.20

5.11. Вычисление определенных интегралов

Для вычисления определенного интеграла необходимо выбрать знак интеграла из палитры или набрать его нажатием клавиши &. После этого следует вписать пределы интегрирования, подынтегральную функцию и переменную интегрирования. Mathcad успешно справляется с большинством интегралов, в том числе с несобственными. Точность вычислений регулируется встроенной переменной TOL. По умолчанию ее значение установлено TOL := 10^{-3}.

Приведем несколько примеров успешного вычисления несобственных интегралов, интеграла от быстро осциллирующей функции и интеграла от ступенчатой функции:

$$\int_{-\infty}^{-1} \frac{1}{x^2}\,dx = 1, \qquad \int_{0}^{\infty} x \cdot \mathrm{e}^{-x^2}\,dx = 0.5,$$

$$\int_{10}^{20} \sin(x^3)\,dx = 1.822 \cdot 10^{-3}, \qquad \int_{-1}^{1} \Phi(x)\,dx = 1.$$

Здесь $\Phi(x) = \begin{cases} 0, & \text{если } x < 0 \\ 1, & \text{если } x \geq 0 \end{cases}$.

Зависимость результата от заданной точности вычислений:

$$\int_{-\infty}^{\infty} \frac{1}{1+x^2}\,dx = 3.14159265369356$$

$$\text{TOL} := 10^{-6}, \quad \int_{-\infty}^{\infty} \frac{1}{1+x^2}\,dx = 3.14159265358979$$

Для этого примера результат может быть получен также в символьном виде. Для этого вместо знака равенства необходимо нажать знак символического равенства `Ctrl+`

$$\int_{-\infty}^{\infty} \frac{1}{1+x^2}\,dx \to \pi.$$

В то же время в некоторых случаях несобственные интегралы вычисляются неправильно:

$$\int_{-1}^{1} \frac{1}{x^2}\,dx = 1.376 \cdot 10^3.$$

Хотя очевидно, что $\int_{-1}^{1} \frac{1}{x^2}\,dx \to \infty$.

5.12. Решение дифференциальных уравнений

5.12.1. Обыкновенные дифференциальные уравнения

Пусть необходимо найти решение уравнения $y' = f(x, y)$ с начальным условием $y(x_0) = y_0$. Такая задача называется ***задачей Коши***. Разложим искомую функцию $y(x)$ в ряд вблизи точки x_0 и ограничимся первыми двумя членами разложения $y(x) = y(x_0) + y'(x)(x - x_0) + \ldots$. Учтя уравнение и обозначив $x - x_0 = h$, получаем $y(x) = y(x_0) + f(x_0, y_0)\Delta x$. Эту формулу можно применять многократно, находя значения функции во все новых и новых точках:

$$y_{i+1} = y_i + f(x_i, y_i)h.$$

Такой метод решения обыкновенных дифференциальных уравнений называется методом Эйлера. Геометрически метод Эйлера означает, что на каждом шаге мы аппроксимируем решение (интегральную кривую) отрезком касательной, проведенной к графику решения в начале интервала. Точность метода невелика и имеет порядок *h*. Говорят, что метод Эйлера – метод первого порядка, то есть его точность растет линейно с уменьшением шага *h*. Существуют различные модификации метода Эйлера, позво-

ляющие увеличить его точность. Все они основаны на том, что производную, вычисленную в начале интервала, заменяют на среднее значение производной на данном интервале.

Оценку значения производной можно улучшить, увеличивая число вспомогательных шагов. На практике наиболее распространенным методом решения обыкновенных дифференциальных уравнений ***является метод Рунге – Кутты*** четвертого порядка. Для оценки значения производной в этом методе используется четыре вспомогательных шага. Формулы метода Рунге – Кутты следующие:

$$k_1^i = hf(x_i, y_i),$$

$$k_2^i = hf(x_i + \frac{h}{2}, y_i + \frac{k_1^i}{2}),$$

$$k_3^i = hf(x_i + \frac{h}{2}, y_i + \frac{k_2^i}{2}),$$

$$k_4^i = hf(x_i + h, y_i + k_3^i),$$

$$\Delta y_i = \frac{1}{2}(k_1^i + 2k_2^i + 2k_3^i + k_4^i),$$

$$y_{i+1} = y_i + \Delta y_i.$$

Перечисленные методы можно применять и для решения систем дифференциальных уравнений. Поскольку многие дифференциальные уравнения высших порядков могут быть сведены заменой переменных к системе дифференциальных уравнений первого порядка, рассмотренные методы могут быть использованы и для решения дифференциальных уравнений выше первого порядка.

Еще один тип задач, часто встречающихся на практике, – краевые задачи. Пусть имеется дифференциальное уравнение второго порядка $y'' = f(x, y, y')$. Решение уравнения требуется найти на интервале [0,1], причем известно, что $y(0) = y_0$, $y(1) = y_1$. Понятно, что произвольный интервал $[a,b]$ заменой переменных $t = \frac{x-a}{b-a}$ может быть сведен к единичному. Для решения краевой задачи обычно применяют ***метод стрельб***. Пусть $y'(0) = k$, где k – некоторый параметр. Для некоторого пробного значения k может быть решена задача Коши, например, методом Рунге – Кутты. Полученное решение будет зависеть от значения параметра $y = y(x;k)$.

5.12.2. Метод Эйлера для дифференциальных уравнений первого порядка

Решим задачу Коши для дифференциального уравнения первого порядка $y' = f(x, y)$ методом Эйлера.

Пусть правая часть уравнения равна $f(x, y) \equiv x \cdot y$.

Зададим границы изменения $\boldsymbol{x}$: $x_{\min} \equiv 0 \quad x_{\max} \equiv 1$.

Зададим число точек и величину шага: $n \equiv 10 \quad h = \dfrac{(x_{\max} - x_{\min})}{n}$.

Зададим начальные условия: $y_0 \equiv 1 \quad x_0 \equiv x_{\min}$.

Вычислим $\boldsymbol{x}$ и $\boldsymbol{y}$ по формулам Эйлера $j \equiv 1..n$, $x_j \equiv x_{\min} + j \cdot h$, $y_j \equiv y_{j-1} + f(x_{j-1}, y_{j-1}) \cdot h$.

Представим результат графически (рис. 4.21) и сравним его с аналитическим решением $z := 0, 0.1..1$, $y1(z) := \exp\left(\dfrac{z^2}{2}\right)$, $k \equiv 0..n$.

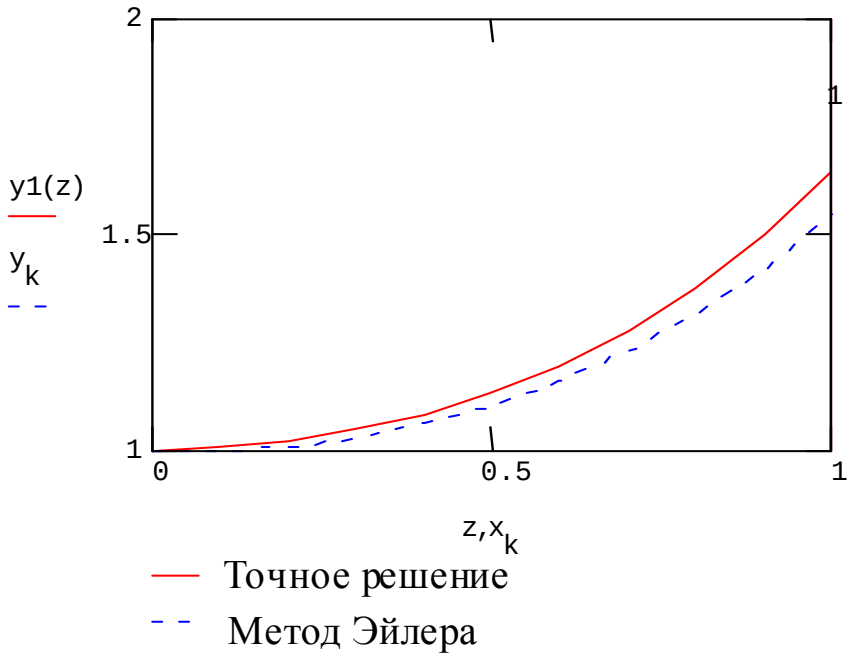


Рис. 4.21

Точное аналитическое решение и решение, полученное численно, отличаются в точке $x = 1$ на $y1(1) = -y_n = 0.102$.

То есть относительная ошибка составляет $\dfrac{y1(1) - y_n}{y1(1)} = 6.163\%$.

5.12.3. Решение дифференциальных уравнений методом Рунге – Кутты

Функция rkfixed(y, $x1$, $x2$, n, F) возвращает матрицу решений методом Рунге – Кутты дифференциального уравнения, правая часть которого записана в символьном векторе F на интервале от $x1$ до $x2$ при фиксированном числе шагов n. Начальные условия – в векторе y.

Необходимо решить уравнение $y' = 2xy^3 - 1$; $x1 = 0$; $x2 = 1$; $y0 = 0$; $n = 8$.

Интервал поиска решения $x1 := 0$ $x2 := 1$.

Число шагов $n := 8$.

Вектор начальных условий $y0 := 0$.

Решаемое уравнение $F(x, y) := (2xy^3) - 1$.

$z := \text{rkfixed}(y0, x1, x2, n, F)$

Найденное решение

$$z = \begin{pmatrix} 0 & 0 \\ 0.125 & -0.125 \\ 0.25 & -0.25 \\ 0.375 & -0.378 \\ 0.5 & -0.513 \\ 0.625 & -0.668 \\ 0.75 & -0.871 \\ 0.875 & -1.225 \\ 1 & -2.652 \end{pmatrix}.$$

5.12.4. Решение систем дифференциальных уравнений

Для решения дифференциальных уравнений Mathcad имеет ряд встроенных функций, в частности, функцию ***rkfixed***, реализующую метод Рунге – Кутты четвертого порядка с фиксированным шагом. Фактически эта функция предназначена для решения систем дифференциальных уравнений первого порядка:

$$y_1' = f_1(x, y_1, y_2, \ldots, y_n),$$
$$y_2' = f_2(x, y_1, y_2, \ldots, y_n),$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots,$$
$$y_n' = f_n(x, y_1, y_2, \ldots, y_n).$$

Функция ***rkfixed(y, x1, x2, npoints, D)*** возвращает матрицу. Первый столбец этой матрицы содержит точки, в которых получено решение, а остальные столбцы – решения и его первые $n-1$ производные.

Аргументы функции:

– ***y*** – вектор начальных значений (***n*** элементов).

– ***x1*** и ***x2*** – границы интервала, на котором ищется решение дифференциального уравнения.

– ***npoints*** – число точек внутри интервала ***(x1, x2)***, в которых ищется решение. Функция ***rkfixed*** возвращает матрицу, состоящую из ***1 + npoints*** строк.

– ***D*** – вектор, состоящий из n элементов, который содержит первые производные искомой функции.

В качестве примера рассмотрим решение системы Вольтерра – Лотки. Эта система описывает динамику численности хищников и жертв на замкнутом ареале и является одной из базовых моделей экологии.

$$\frac{dN_1}{dt} = N_1\left(\varepsilon_1 - \gamma_2 N_2\right),$$
$$\frac{dN_2}{dt} = N_2\left(\varepsilon_2 - \gamma_1 N_1\right).$$

Для решения систем дифференциальных уравнений используются функция ***rkfixed.***

Внимание! В этом примере установлено значение ***ORIGIN = 1***, то есть нумерация элементов массива начинается с 1, а не с 0, как это принято в Mathcad по умолчанию.

Пусть в начальный момент времени число хищников $N_1 = 5$ и число жертв $N_2 = 10$.

Задаем вектор начальных значений $N := \begin{pmatrix} 5 \\ 10 \end{pmatrix}$, параметры системы $\varepsilon := \begin{pmatrix} 0.1 \\ 0.3 \end{pmatrix}$, $y := \begin{pmatrix} 0.03 \\ 0.04 \end{pmatrix}$, интервал времени и количество точек, в которых будет вычислено решение, $t_{\max} := 200$, $\text{npoints} := 400$ и вектор правых частей системы. (Поскольку исходная система не зависит явно от времени ***t***, функция ***D*** также не зависит от времени явно хотя и содержит его в числе своих аргументов.)

$$D(t, N) := \begin{bmatrix} N_1 \cdot (\varepsilon_1 - \gamma_2 \cdot N_2) \\ -N_2 \cdot (\varepsilon_2 - \gamma_1 \cdot N_1) \end{bmatrix}.$$

Решаем систему с помощью встроенной функции

$$Z := \text{rkfixed}(N, 0, t_{max}, \text{npoints}, D), \quad k := 1...\text{npoints}.$$

Представим на графике результаты расчета – *зависимость численности популяций от времени* (рис. 4.22) и *зависимость числа жертв от числа хищников* (рис. 4.23).

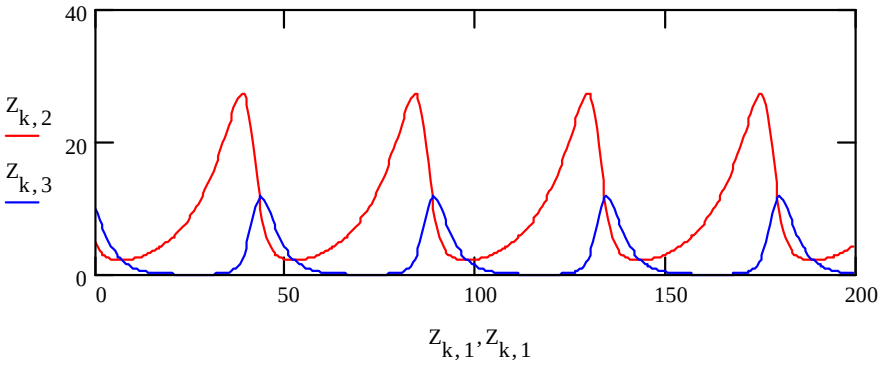


Рис. 4.22

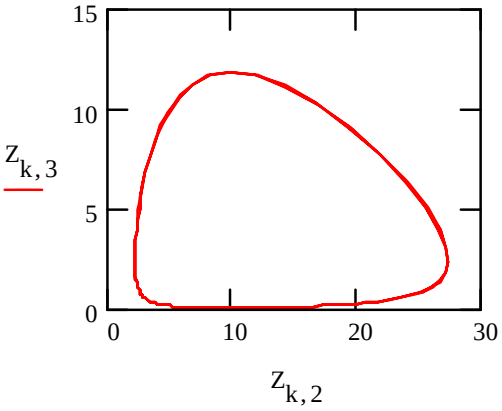


Рис. 4.23

Можно использовать обозначения $\left(z^{<i>}\right)_k$ или $z_{k,i}$ – это одно и то же.

Поскольку дифференциальное уравнение порядка выше первого может быть преобразовано к системе дифференциальных уравнений первого порядка, функция ***rkfixed*** может быть использована и для решения дифференциальных уравнений.

5.12.5. Решение дифференциальных уравнений второго порядка

В качестве примера решим задачу о гармоническом осцилляторе, для которого известно аналитическое решение, и легко может быть оценена точность вычислений. Дифференциальное уравнение второго порядка

$$y'' + 2\beta y' + \omega^2 y = 0$$

преобразуем к системе из двух дифференциальных уравнений первого порядка

$$y' = x,$$

$$x' = -2\beta x - \omega^2 y.$$

Пусть декремент затухания $\beta := 0.0$ и циклическая частота $\omega := 1$.

Зададим начальные условия $y := \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

Начальной координате соответствует ***y0***, а ***y1*** – начальной скорости. Зададим теперь матрицу ***D***. С учетом того, что искомая величина соответствует нулевому элементу массива ***y***, ее первая производная – первому, а вторая – второму, имеем

$$D(t, y) := \begin{pmatrix} y_1 \\ -2\beta y_1 - \omega^2 y_0 \end{pmatrix}, \quad Z := \text{rkfixed}(y, 0, 5\pi, 100, D).$$

Представим результаты расчета на графике (рис. 4.24) и сравним их с аналитическим решением $f(x) := y_1 \exp(-\beta \cdot x) \cdot \cos(\omega \cdot x)$, $k \equiv 0..100$ $x := 0, 0.1..5\pi$.

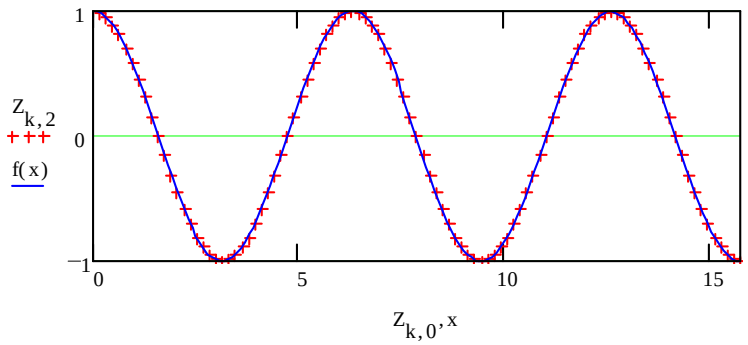


Рис. 4.24

Для контроля точности вычислений нарисуем фазовую траекторию (зависимость смещения от скорости) (рис. 4.25). Для гармонического осциллятора фазовая траектория должна иметь вид эллипса.

Mathcad имеет еще две функции для решения задачи Коши. Это функции ***Rkadapt*** и ***Bulstoer***. Эти функции имеют те же самые аргументы и возвращают решения в такой же форме, что и функция ***rkfixed***. Первая из этих функций использует метод Рунге – Кутты с переменным шагом, что позволяет повысить точность вычислений и сократить их объем, если искомое решение имеет области, где ее значения меняются быстро, и области плавного изме-

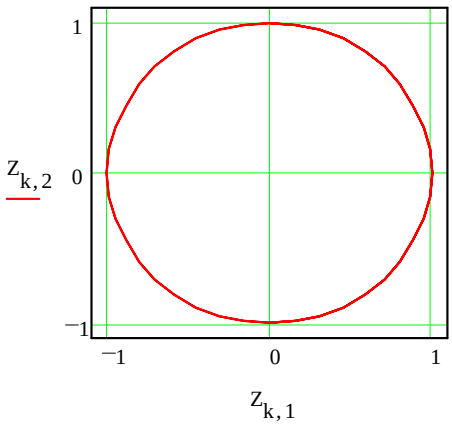


Рис. 4.25

нения. Функция ***Rkadapt*** будет варьировать величину шага в зависимости от скорости изменения решения.

Функция ***Bulstoer*** реализует иной численный метод – метод Булирша – Штёра. Ее следует применять, если известно, что решение является гладкой функцией.

5.12.6. Решение краевой задачи

Пусть требуется найти решение дифференциального уравнения $y'' + 2\beta y + \omega_0^2 y = 0$ при условиях $y(0) = 1$ и $y(5\pi) = 0$.

При значениях параметров $\beta := 0.1,\ \omega_0 := 1.$

Для решения краевой задачи имеется встроенная функция ***sbval***, реализующая метод стрельб и позволяющая свести краевую задачу к задаче Коши. Функция ***sbval*** имеет следующие параметры:

– ***v*** – вектор, содержащий начальные приближения для недостающих начальных условий;

– ***xmin***, ***xmax*** – границы интервала, на котором ищется решение;

– ***D(x,y)*** – вектор-функция, содержащая правые части системы дифференциальных уравнений первого порядка, эквивалентной исходному уравнению, размер вектора ***n*** совпадает со степенью старшей производной дифференциального уравнения;

– ***load(xmin,v)*** – вектор-функция, элементы которой соответствуют ***n*** значениям функций на левой границе интервала. Часть этих значений известна, а для части заданы начальные приближения в векторе ***v***. Их уточненные значения будут найдены в процессе вычисления

– ***score(xmax,y)*** – вектор-функция, имеющая то же число элементов, что и ***v***. Каждое значение является разностью между начальными значениями в конечной точке интервала и соответствующей оценки для решения. Этот вектор показывает, на сколько близко найденное решение к истинному.

Наша задача сводится к системе двух дифференциальных уравнений первого порядка:

$$y' = z,$$

$$z' = -2\beta z - \omega_0^2 y.$$

Поэтому функция D имеет вид $D(t, y) := \begin{pmatrix} y_1 \\ -2\beta y_1 - {\omega_0}^2 y_0 \end{pmatrix}.$

Задаем граничные условия: $x_{\min} = 0,\ x_{\max} = 5\pi.$

Задача Коши для дифференциального уравнения второго порядка содержит два начальных условия. Нам известно только одно. Начальное приближение для недостающего значения задаем в векторе ***v***, который в нашем случае состоит только из одного элемента. Несмотря на это, индекс 0 должен быть обязательно указан, чтобы подчеркнуть векторный характер этой величины: $v_0 := 0.1$.

На левой границе интервала нам известно значение y $\left(y(0)=1\right)$ и задано начальное приближение для y' $\left(y'=0\right)$. Это значение записано в v_0. Задаем вектор-функцию ***load***. Ее нулевой элемент – начальное значение для y_0, первый – для y_1.

$$\text{load}(x_{\min}, v) := \begin{pmatrix} 1 \\ v_0 \end{pmatrix}, \quad \text{score}(x_{\max,y}) := y_0 - 0,$$

$$S := \text{sbval}(v, x_{\min}, x_{\max}, D, \text{load}, \text{score}), \quad S = 12.511.$$

Теперь, когда нам стало известно недостающее начальное условие в задаче Коши, можно воспользоваться, например, функцией ***rkfixed*** (рис. 4.26):

$$n := 500 \qquad i := 0..n \qquad y := \begin{pmatrix} 1 \\ S_0 \end{pmatrix} \qquad Z := \text{rkfixed}(y, x_{\min}, x_{\max}, 500, D).$$

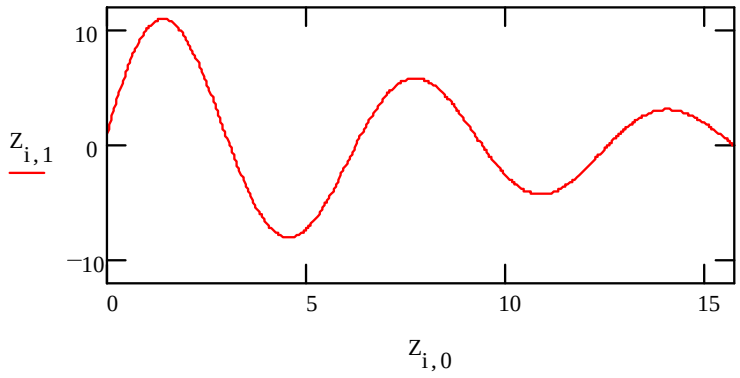


Рис. 4.26

5.12.7. 4.12.7. Решение обыкновенных дифференциальных уравнений в Mathcad

Mathcad предлагает новый способ для решения обыкновенных дифференциальных уравнений, разрешенных относительно старшей производной. Для этих целей служит уже известный нам блок ***given*** совместно с функцией ***odesolve***. Дифференциальное уравнение совместно с начальными или граничными условиями записывается в блоке ***Given***. Производные

можно обозначать как штрихами (Ctrl+F7), так и с помощью знака производной $\frac{d}{dx}$. Приведем пример использования функции для решения задачи Коши (рис. 4.27):

$\beta := 0.1$

Given

$x''(t) + 2\beta x'(t) + x(t) = 0$

$x(0) = 1$

$x'(0) = 1$

$x := \text{odesolve}(t, 5\pi, 0.1)$

$t := 0, 0.1..5\pi$

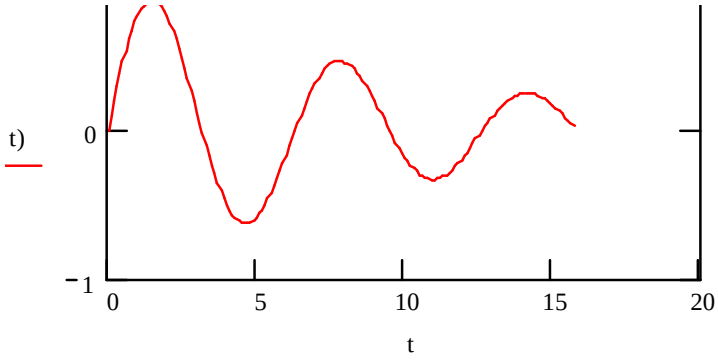


Рис. 4.27

У искомой функции явно указан аргумент, знак производной стоит перед скобкой. Функция ***odesolve*** имеет три аргумента. Первый аргумент – независимая переменная, вторая – граница интервала, на котором ищется решение, последний аргумент – шаг, с которым ищется решение. Последний аргумент может быть опущен.

Следующий пример демонстрирует решение краевой задачи. Показан другой способ записи производных, используется ***odesolve*** функция с двумя аргументами (рис. 4.28):

Given

$\frac{d^2}{dt^2}x(t) + 2\beta\frac{d}{dt}x(t) + x(t) = 0$

$x(0) = 1$

$x(5\pi) = 0.1$

$x := \text{odesolve}(t, 5\pi)$

$t := 0, 0.1..5\pi$

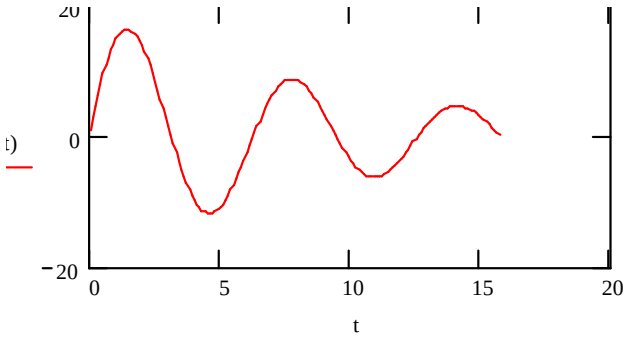


Рис. 4.28

5.13. Решение уравнений в частных производных

Одним из методов решения дифференциальных уравнений в частных производных является ***метод сеток***. Идея метода заключается в следующем. Для простоты, ограничимся случаем только функции двух переменных, и будем полагать, что решение уравнения ищется на квадратной об-

ласти единичного размера. Разобьем область сеткой. Шаг сетки по оси x и по оси y, вообще говоря, может быть разный. По определению частная производная равна

$$\frac{\partial u(x,y)}{\partial x} = \lim_{\Delta x \to 0} \frac{u(x+\Delta x, y) - u(x)}{\Delta x} \approx \frac{u(x+\Delta x, y) - u(x)}{\Delta x}.$$

Если рассматривать функцию только в узлах сетки, то частную производную можно записать в форме

$$\frac{\partial u(x,y)}{\partial x} \approx \frac{u_{i+1,j} - u_{i,j}}{h},$$

где узел (i, j) соответствует точке (x, y). Полученное выражение называется ***правой конечной разностью***. Название связано с тем, что для вычисления производной в точке используются значения функций в этой точке и точке, лежащей правее. Очевидно, что сходное выражение можно было бы получить, используя точку, лежащую слева

$$\frac{\partial u(x,y)}{\partial x} \approx \frac{u_{i,j} - u_{i-1,j}}{h}.$$

Такое выражение называется ***левой конечной разностью***. Можно получить центральную конечную разность, найдя среднее этих выражений.

Теперь получим выражения для вторых производных

$$\frac{\partial^2 u(x,y)}{\partial x^2} \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}.$$

В данном случае для нахождения производной использованы симметричные точки. Однако, очевидно, можно было бы использовать точки с несимметричным расположением.

5.13.1. Уравнения гиперболического типа

В качестве примера рассмотрим решение волнового уравнения (уравнения гиперболического типа).

$$\frac{\partial^2 U}{\partial x^2} = \frac{1}{v^2} \frac{\partial^2 U}{\partial t^2}.$$

Уравнение будем решать методом сеток. Запишем уравнение в конечных разностях

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} = \frac{1}{v^2} \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j}}{\tau^2}.$$

Полученное уравнение позволяет выразить значение функции u в момент времени $j+1$ через значения функции в предыдущие моменты времени

$$u_{i,j+1} = v^2\left(\frac{\tau}{h}\right)^2 (u_{i+1,j} - 2u_{i,j} + u_{i-1,j}) + 2u_{i,j} - u_{i,j-1}.$$

Такая разностная схема называется ***явной***, так как искомая величина получается в явном виде. Она устойчива, если $\tau \le h/v$.

Зададим начальные условия: смещение струны ***U*** в начальный и последующий моменты времени описывается синусоидальной функцией:

$$n := 20 \quad j := 0..n \quad i := 0..100$$

$$U_{i,0} := \sin\left(\pi\frac{i}{50}\right) \quad U_{i,1} := U_{i,0}.$$

Совпадение смещений при j = 0 и j = 1 соответствует нулевой начальной скорости.

Зададим граничные условия: на концах струны смещение равно 0 в любой момент времени: $U_{0,j} := 0$, $U_{100,j} := 0$.

Будем полагать коэффициент $a := 1$, $k := 0.02$, $i := 1..99$, $j := 1..n-1$.

Записываем уравнение в конечных разностях, разрешенное относительно $U_{i,j+1}$

$$U_{i,j+1} := a^2 k(U_{i+1,j} - 2U_{i,j} + U_{i-1,j}) + 2U_{i,j} - U_{i,j-1}.$$

Представляем результат на графике (рис. 4.29)

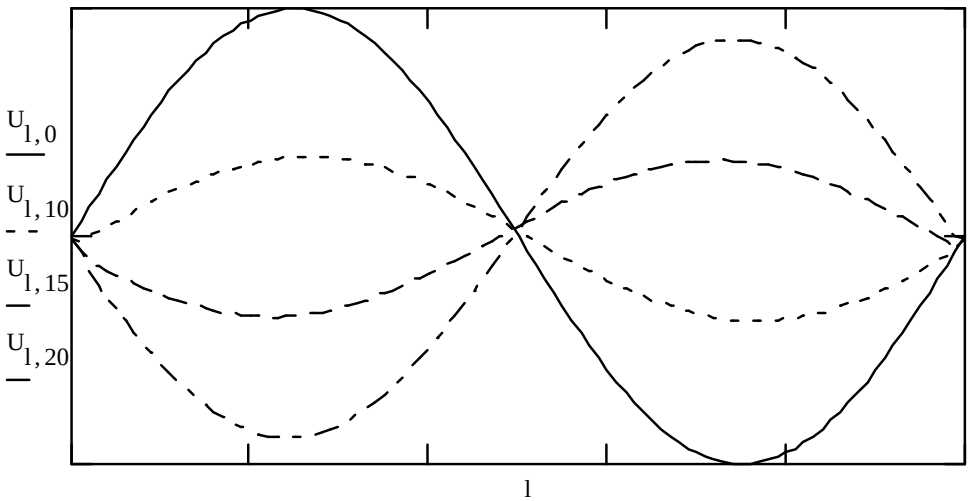


Рис. 4.29

5.13.2. Уравнения параболического типа

Еще один пример использования конечных разностей – уравнение диффузии

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}.$$

Это уравнение параболического типа. Явная разностная схема для этого уравнения имеет вид

$$u_{i,\,j+1} = D\frac{\tau}{h^2}(u_{i+1,\,j} - 2u_{i,\,j} + u_{i-1,\,j}) + u_{i,\,j}. \qquad (2.1)$$

Эта разностная схема устойчива, если $\tau \le \frac{h^2}{2D}$. Для краткости в дальнейшем будем обозначать весь множитель, стоящий перед скобкой, как κ:

Задаем коэффициент $\kappa := 0.15$ и диапазон изменения пространственной и временной координат: $t := 0..29$, $x := 1..49$.

Задаем начальные и граничные условия: $f_{0,x} := 0$, $f_{0,0} := 0$, $f_{0,50} := 0$, $f_{0,25} := 1$.

Уравнение в конечных разностях имеет вид

$$f_{t+1,x} := f_{t,x} + \kappa(f_{t,x-1} - 2f_{t,x} + f_{t,x+1}).$$

Представляем результаты на графике (рис. 4.30) (для большей наглядности изображена только центральная часть) $x := 15…35$.

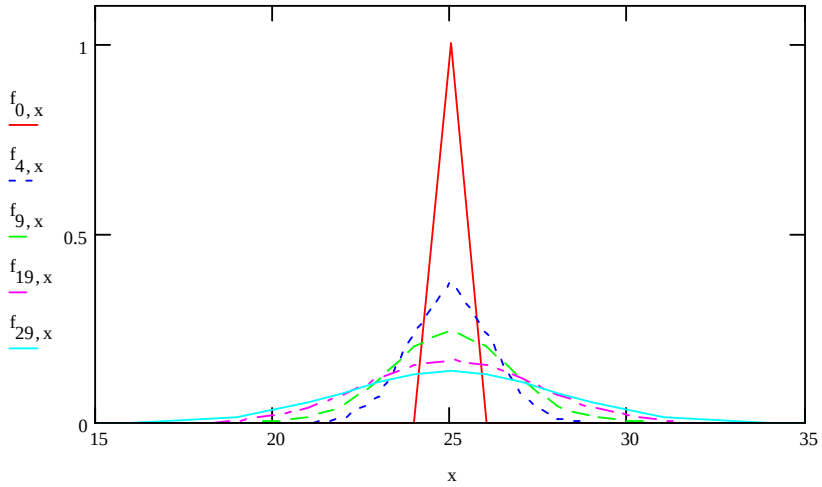


Рис. 4.30

Основное достоинство явных методов – их простота – зачастую сводится на нет достаточно жесткими ограничениями на величину шага. Яв-

ные схемы обычно устойчивы при столь малых шагах по времени, что они становятся непригодными для практических расчетов. Этого существенного недостатка позволяют избежать неявные схемы. Свое название они получили потому, что значения искомой функции на очередном временном шаге не могут быть явно выражены через значения функции на предыдущем шаге.

Рассмотрим применение неявной схемы на примере уравнения теплопроводности

$$\frac{\partial u}{\partial t} = c\frac{\partial^2 u}{\partial x^2} \tag{2.2}$$

Запишем неявную разностную схему для этого уравнения

$$\frac{u_{i,j+1} - u_{i,j}}{\tau} = \frac{c}{h^2}\left(u_{i+1,j+1} - 2u_{i,j+1} + u_{i-1,j+1}\right). \tag{2.3}$$

Здесь первый индекс соответствует пространственной, а второй – временной координате. В отличие от явной схемы, для вычисления в правой части уравнения используются значения функции на том же самом временном шаге. Вводя обозначение $\mu = \frac{c\tau}{h^2}$, уравнение (2.3) можно переписать в виде

$$(1+2\mu)u_{i,j+1} - \mu(u_{i+1,j+1} + u_{i-1,j+1}) = u_{i,j} \tag{2.4}$$

или в матричной форме

$$\begin{pmatrix} 1+2\mu & -\mu & & & & \\ -\mu & 1+2\mu & -\mu & & & \\ & & \cdots & & & \\ & & & 1+2\mu & -\mu & \\ & & & -\mu & 1+2\mu & \end{pmatrix} \begin{pmatrix} u_{1,j+1} \\ u_{2,j+1} \\ \vdots \\ u_{n-1,j+1} \\ u_{n,j+1} \end{pmatrix} = \begin{pmatrix} u_{1,j} + \mu\alpha \\ u_{2,j} \\ \vdots \\ u_{n-1,j} \\ u_{n,j} + \mu\beta \end{pmatrix} \tag{2.5}$$

где $u(0,t) = \alpha,\ u(t,1) = \beta$.

Задаем количество узлов сетки (в данном случае оно одинаково для обеих переменных) $n := 30,\ i := 0\ldots n,\ j := 0\ldots n,\ k := 0\ldots n-1,\ m := 0\ldots n-1.$

Задаем значения параметров $\alpha := 0,\ \beta := 1,\ \mu := 5$ и начальное распределение температуры в области $u_{i,0} := \sin\left(\pi \cdot \frac{i}{n}\right) + \frac{i}{n}.$

Формируем матрицы уравнения (2.5)

$u_{0,j} := \alpha$ $\quad u_{n,j} := \beta$

$A_{i,i} := 1 + 2\mu$ $\quad A_{m,m-1} := -\mu$ $\quad A_{m-1,m} := -\mu$

$\mu\alpha_i := 0$ $\quad \mu\alpha_0 := \mu\alpha$ $\quad \mu\beta_i := 0$ $\quad \mu\beta_0 := \mu\beta$

Находим решение системы (рис. 4.31)

$$u^{\langle j+1\rangle} := A^{-1} \cdot \left(u^{\langle j\rangle} + \mu\alpha + \mu\beta\right).$$

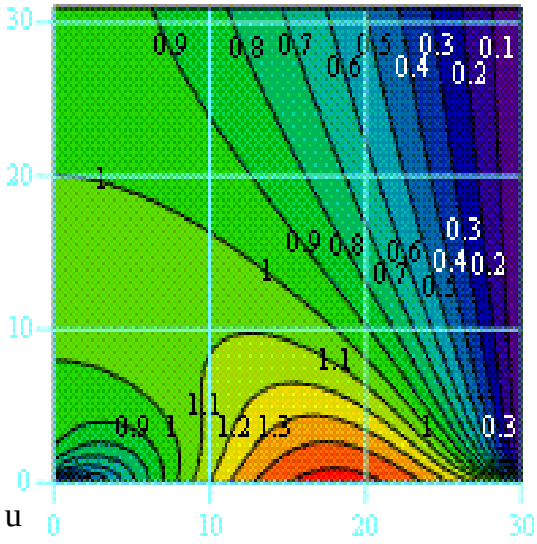


Рис. 4.31

5.13.3. Решение уравнений Лапласа и Пуассона

Для решения уравнений Пуассона $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = F(x, y)$ и Лапласа (частный случай, когда $F(x, y) = 0$) – уравнений эллиптического типа – предназначена функция ***relax(a, b, c, d, e, f, u, rjac)***, реализующая метод релаксации. Фактически эту функцию можно использовать для решения эллиптического уравнения общего вида

$$A\frac{\partial^2 u}{\partial x^2} + 2B\frac{\partial^2 u}{\partial x \partial y} + C\frac{\partial^2 u}{\partial y^2} + a\frac{\partial u}{\partial x} + b\frac{\partial u}{\partial y} + cu = F(x, y),$$

$$D = AC - B^2 > 0,$$

которое может быть сведено к уравнению в конечных разностях

$$a_{i,j}u_{i+1,j} + b_{i,j}u_{i-1,j} + c_{i,j}u_{i,j+1} + d_{i,j}u_{i,j-1} + e_{i,j}u_{i,j} = f_{i,j}.$$

В частности, для уравнения Пуассона коэффициенты

$$a_{i,j} = b_{i,j} = c_{i,j} = d_{i,j} = 1, \quad e_{i,j} = -4.$$

Идея метода релаксации заключается в следующем. Если нет источников (уравнение Лапласа), то значение функции в данном узле на текущем шаге $k+1$ определяется как среднее значение функции в ближайших узлах на предыдущем шаге k

$$u_{i,j}^{k+1} = \frac{1}{4}(u_{i-1,j}^{k} + u_{i+1,j}^{k} + u_{i,j-1}^{k} + u_{i,j+1}^{k}) \tag{2.6}$$

При наличии источников разностная схема имеет вид

$$u_{i,j}^{k+1} = \frac{1}{4}(u_{i-1,j}^{k} + u_{i+1,j}^{k} + u_{i,j-1}^{k} + u_{i,j+1}^{k}) - \frac{h^2}{4} f_{i,j} \tag{2.7}$$

Метод релаксации сходится достаточно медленно, так как фактически он использует разностную схему (2.1) с максимально возможным для двумерного случая шагом $\tau = \frac{h^2}{4}$.

В методе релаксации необходимо задать начальное приближение, то есть значения функции во всех узлах области, а так же граничные условия.

Функция ***relax*** возвращает квадратную матрицу, в которой:

1) расположение элемента в матрице соответствует его положению внутри квадратной области,

2) это значение приближает решение в этой точке.

Эта функция использует метод релаксации для приближения к решению.

Функцию ***relax*** необходимо использовать, если известны значения искомой функции ***u(x, y)*** на всех четырех сторонах квадратной области.

Аргументы:

a, b, c, d, e – квадратные матрицы одного и того же размера, содержащие коэффициенты дифференциального уравнения;

f – квадратная матрица, содержащая значения правой части уравнения в каждой точке внутри квадрата;

u – квадратная матрица, содержащая граничные значения функции на краях области, а также начальное приближение решения во внутренних точках области;

rjac – параметр, управляющий сходимостью процесса релаксации.

Он может быть в диапазоне от 0 до 1, но оптимальное значение зависит от деталей задачи.

Пример. $n := 2^4 \quad i := 0..n \quad j := 0..n$.

Задаем правую часть уравнения Пуассона – два точечных источника:

$$M_{i,j} := 0, \quad M_{6,8} := 10, \quad M_{10,8} := -10.$$

Задаем значения параметров функции ***relax***:

$$a_{i,j} := 1, \quad b := a, \quad c := a, \quad d := a, \quad f := M, \quad e := -4a.$$

Задаем граничные условия и начальное приближение – нули во всех внутренних точках области:

$$u_{i,j} := 0, \quad u_{i,n} := -1, \quad u_{0,j} := 1 - 2\frac{j}{n}, \quad u_{i,0} := 1, \quad u_{n,j} := 1 - 2\frac{j}{n}.$$

Находим решение $z :=$ relax (a, b, c, d, e, f, u, 0.95) и представляем его графически (рис. 4.32, 4.33) в виде поверхности и линий уровней.

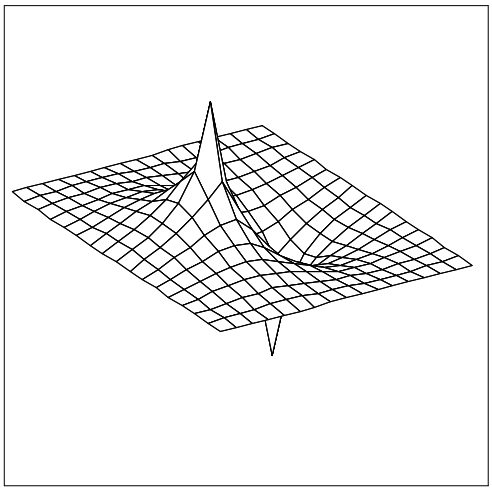


Рис. 4.32

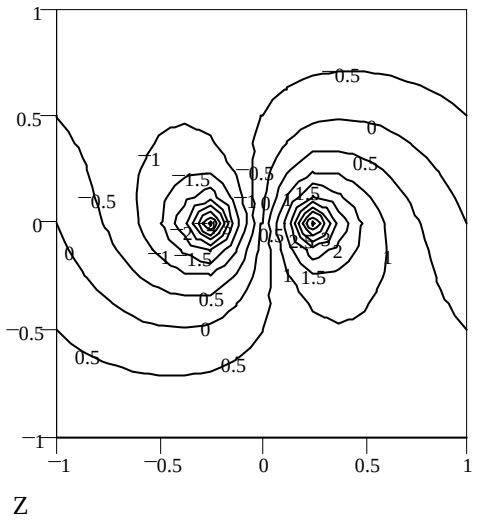


Рис. 4.33

Если граничные условия равны нулю на всех четырех сторонах квадрата, можно использовать функцию ***multigrid***: $z :=$ multigrid (M, 3).

6. КОМПЬЮТЕРНЫЕ СЕТИ

6.1. Назначение компьютерных сетей

При физическом соединении двух или более компьютеров образуется компьютерная *сеть*. В общем случае, для создания компьютерных сетей необходимо специальное аппаратное обеспечение (*сетевое оборудование*) и специальное программное обеспечение (*сетевые программные средства*). Простейшее соединение двух компьютеров для обмена данными называется *прямым соединением*.

Назначение всех компьютерных сетей состоит в обеспечении совместного доступа к общим *ресурсам*. В зависимости от назначения сети в термин *ресурс* можно вкладывать различный смысл. Ресурсы бывают трех типов: *аппаратные*, *программные* и *информационные*. Например, устройство печати (принтер), емкости жестких дисков – это аппаратный ресурс. Когда все участники небольшой компьютерной сети пользуются одним общим принтером, это значит, что они разделяют общий аппаратный ресурс. То же можно сказать и о сети, имеющей один компьютер с увеличенной емкостью жесткого диска (*файловый сервер*), на котором все участники сети хранят свои архивы и результаты работы.

Кроме аппаратных ресурсов компьютерные сети позволяют совместно использовать *программные ресурсы*. Так, например, для выполнения очень сложных и продолжительных расчетов можно подключиться к удаленной большой ЭВМ и отправить вычислительное задание на нее, а по окончании расчетов точно так же получить результат обратно.

Данные, хранящиеся на удаленных компьютерах, образуют *информационный ресурс*. Роль этого ресурса видна наиболее ярко на примере Internet, который воспринимается, прежде всего, как гигантская информационно-справочная система.

При работе в компьютерной сети любого типа одновременно происходит совместное использование всех типов ресурсов. Так, например, обращаясь в Internet за какой-либо информацией, мы используем аппаратные средства, на которых работают программы, обеспечивающие поставку затребованных нами данных.

6.2. Локальные и глобальные сети. Основные понятия

Для обеспечения необходимой совместимости как по аппаратуре, так и по программам в компьютерных сетях действуют специальные стандарты, называемые *протоколами*. Они определяют характер аппаратного взаимодействия компонентов сети (*аппаратные протоколы*) и характер взаимодействия программ и данных (*программные протоколы*). Физически функции поддержки протоколов исполняют аппаратные устройства (*интерфейсы*) и

программные средства (*программы поддержки протоколов*). Программы, выполняющие поддержку протоколов, часто также называют *протоколами*.

Так, например, если два компьютера соединены между собой прямым соединением, то на физическом уровне протокол их взаимодействия определяют конкретные устройства физического порта (параллельного или последовательного) и механические компоненты (разъемы, кабель и т. п.). На более высоком уровне взаимодействие между компьютерами определяют программные средства, управляющие передачей данных. На самом высоком уровне протокол взаимодействия обеспечивают приложения ОС.

В соответствии с используемыми протоколами компьютерные сети принято разделять на *локальные* (*LAN – Local Area Network*) и *глобальные* (*WAN – Wide Area Network*). Компьютеры локальной сети преимущественно используют единый комплект протоколов для всех участников. По территориальному признаку локальные сети отличаются компактностью. Они могут объединять компьютеры одного помещения, этажа, здания, группы компактно расположенных сооружений. Глобальные сети имеют, как правило, увеличенные географические размеры. Они могут объединять как отдельные компьютеры, так и отдельные локальные сети, в том числе и использующие различные протоколы.

Группы сотрудников, работающих над одним проектом в рамках локальной сети, называются *рабочими группами*. В рамках одной локальной сети могут работать несколько рабочих групп. У участников рабочих групп могут быть разные права для доступа к общим ресурсам сети. Совокупность приемов разделения и ограничения прав участников компьютерной сети называется *политикой сети*. Управление сетевыми политиками (их может быть несколько в одной сети) называется *администрированием сети*. Лицо, управляющее организацией работы участников локальной компьютерной сети, называется *системным администратором*.

Создание локальных сетей характерно для отдельных предприятий или отдельных подразделений предприятий. Если предприятие (или отрасль) занимает обширную территорию, то отдельные локальные сети могут объединяться в глобальные сети. В этом случае локальные сети связывают между собой с помощью любых традиционных каналов связи (кабельных, спутниковых, радиорелейных и т. п.).

Для связи между собой нескольких локальных сетей, работающих по разным протоколам, служат специальные средства, называемые *шлюзами*. Шлюзы могут быть как аппаратными, так и программными. Например, это может быть специальный компьютер (*шлюзовой сервер*) или компьютерная программа (*шлюзовое приложение*). В последнем случае компьютер может выполнять не только функцию шлюза, но и какие-то иные функции, типичные для рабочих станций.

При подключении локальной сети предприятия к глобальной сети важную роль играет понятие *сетевой безопасности*. В частности, должен быть ограничен доступ в локальную сеть для посторонних лиц извне, а также ограничен выход за пределы локальной сети для сотрудников предприятия, не имеющих соответствующих прав. Для обеспечения сетевой безопасности между локальной и глобальной сетью устанавливают так называемые *брандмауэры*. Брандмауэром может быть специальный компьютер или компьютерная программа, препятствующая несанкционированному перемещению данных между сетями.

6.3. Краткая история развития Internet

Internet – это глобальная компьютерная сеть, состоящая из множества соединенных друг с другом меньших по размеру сетей и покрывающая весь земной шар. Internet можно рассматривать в физическом смысле как миллионы компьютеров, связанных друг с другом всевозможными линиями связи, однако такой «физический» взгляд на Internet слишком узок. Лучше рассматривать Internet как некое информационное пространство.

Ранние эксперименты по передаче и приему информации с помощью компьютеров начались еще в 50-х годах и имели лабораторный характер. В США решение о создании первой глобальной сети национального масштаба было принято в 1958 г.

Поводом для создания глобальной компьютерной сети стала разработка Пентагоном глобальной системы раннего оповещения о пусках ракет (*NORAD – North American Aerospare Defence Command*). Станции системы *NORAD* протянулись через север Канады от Аляски до Гренландии, а подземный командный центр расположился вблизи города Колорадо-Спрингс в недрах горы Шайенн. Центр управления был введен в действие в 1964 г., и, собственно, с этого времени можно говорить о работе первой глобальной компьютерной сети, хотя и ведомственной. С середины 60-х годов к ней стали подключаться авиационные, метеорологические и другие военные и гражданские службы.

Курированием работы сети занималась специальная организация – Управление перспективных разработок министерства обороны США (*DARPA – Defense Advanced Research Project Agency*). Основным недостатком централизованной сети была недостаточная устойчивость, связанная с тем, что при выходе из строя какого-либо из узлов полностью выходил из строя и весь сектор, находившийся за ним, а при выходе из строя центра управления выходила из строя вся сеть. Во времена ядерного противостояния сверхдержав этот недостаток был критичным.

Решение проблемы устойчивости и надежности сети было поручено управлению *DARPA*. Основными направлениями исследований стали по-

иск новых протоколов обслуживания сети и новых принципов сетевой архитектуры. Полигоном для испытаний новых принципов стали крупнейшие университетские и научные центры США, между которыми были проложены линии компьютерной связи. Со стороны министерства обороны работы курировались тем же управлением *DARPA*, и первая вневедомственная национальная компьютерная сеть получила название *ARPANET*. Ее внедрение состоялось в 1969 г.

В 70-е годы сеть *ARPANET* медленно развивалась. В основном развитие происходило за счет подключения региональных сетей, воссоздающих общую архитектуру *ARPANET* на более низком уровне (в региональном или локальном масштабе). Основной объявленной задачей *ARPANET* стала координация групп коллективов, работающих над едиными научно-техническими проектами, а основным назначением стал обмен электронной почтой и файлами с научной и проектно-конструкторской документацией. В то же время не прекращались работы над основной необъявленной задачей – разработкой новых сетевых протоколов, способных обеспечить живучесть глобальной сети даже в ядерном конфликте.

Второй датой рождения Internet принято считать 1983 г. В этом году произошли революционные изменения в программном обеспечении компьютерной связи. Проблема устойчивости глобальной сети была решена внедрением протокола *TCP/IP*, лежащего в основе всемирной сети и по нынешний день. Решив, наконец, эту задачу, управление *DARPA* прекратило свое участие в проекте и передало управление сетью Национальному научному фонду (*NSF*). Так в 1983 г. образовалась глобальная сеть *NSFNET*. В середине 80-х годов к ней начали активно подключаться академические и научные сети других стран, например академическая сеть Великобритании *JANET* (*Joined Academic Network*).

Во второй половине 80-х годов произошло деление всемирной сети на домены по принципу принадлежности. Домен *gov* финансировался на средства правительства, домен *sci* – на средства научных кругов, домен *edu* – на средства системы образования, а домен *com* (коммерческий) не финансировался никем, то есть его узлы должны были развиваться за счет собственных ресурсов. Национальные сети других государств стали рассматриваться как отдельные домены, например, *uk* – домен Великобритании, *by* – домен Беларуси, *ru* – домен России.

Когда во второй половине 80-х годов сложилась и заработала система доменных имен (*DNS, Domain Name System*), Национальный научный фонд США утратил контроль над развитием сети. Тогда и появилось понятие «Internet» как саморазвивающейся децентрализованной иерархической структуры.

Стек протоколов TCP/IP. В техническом понимании *TCP/IP* – это не один сетевой протокол, а два протокола, лежащих на разных уровнях (это так называемый *стек протоколов*). Протокол *TCP* – протокол *транспортного уровня*. Он управляет тем, как происходит передача данных. Протокол *IP* – адресный. Он принадлежит *сетевому уровню* и определяет, куда происходит передача.

Протокол TCP. Согласно протоколу *TCP* (*Transmission Control Protocol*), отправляемые данные «нарезаются» на небольшие пакеты, после чего каждый пакет маркируется таким образом, чтобы в нем были данные, необходимые для правильной сборки документа на компьютере получателя.

Протокол IP. Суть адресного протокола *IP* (*Internet Protocol*) состоит в том, что у каждого участника Всемирной сети должен быть свой уникальный адрес (*IP-адрес*). Без этого нельзя говорить о точной доставке *TCP*-пакетов на нужное рабочее место. Этот адрес выражается четырьмя байтами, например, 195.38.46.11. Структура *IP*-адреса организована так, что каждый компьютер, через который проходит какой-либо *TCP*-пакет, может по этим четырем числам определить, кому из ближайших «соседей» надо переслать пакет, чтобы он оказался «ближе» к получателю. В результате конечного числа перебросок *TCP*-пакет достигает адресата.

Слово «ближе» не случайно взято в кавычки, в данном случае оценивается не географическая близость. В расчет принимаются условия связи и пропускная способность линии. Два компьютера, находящиеся на разных континентах, но связанные высокопроизводительной линией космической связи, считаются более «близкими» друг к другу, чем два компьютера из соседних поселков, связанные простым телефонным проводом. Решением вопросов, что считать «ближе», а что «дальше», занимаются специальные средства – *маршрутизаторы*. Роль маршрутизаторов в сети обычно выполняют специализированные компьютеры, но это могут быть и специальные программы, работающие на узловых серверах сети.

6.4. Службы Internet

Когда говорят о работе в Internet или об использовании Internet, то на самом деле речь идет не об Internet в целом, а только об одной или нескольких из его многочисленных служб. В зависимости от конкретных целей и задач клиенты сети используют те службы, которые им необходимы.

В простейшем понимании *служба* – что пара программ, взаимодействующих между собой согласно определенным правилам, называемым *протоколами*. Одна из программ этой пары называется *сервером*, а вторая – *клиентом*. Соответственно, когда говорят о работе служб Internet,

речь идет о взаимодействии серверного оборудования и программного обеспечения и клиентского оборудования и программного обеспечения.

Так, например, для передачи файлов в Internet используется специальный прикладной протокол *FTP* (*File Transfer Protocol*). Соответственно, чтобы получить из Internet файл, необходимо:

– иметь на компьютере программу, являющуюся клиентом *FTP* (*FTP-клиент*);

– установить связь с сервером, предоставляющим услуги *FTP* (*FTP-сервером*).

6.5. Telnet

Исторически одной из ранних является служба удаленного управления компьютером *Telnet*. Подключившись к удаленному компьютеру по протоколу этой службы, можно управлять его работой. Такое управление еще называют *консольным* или *терминальным*. В прошлом эту службу широко использовали для проведения сложных расчетов на удаленных вычислительных центрах. Так, например, если для очень сложных вычислений на персональном компьютере требовались недели непрерывной работы, а на удаленной супер-ЭВМ всего несколько минут, то ПК применяли для удаленного ввода данных в ЭВМ и для приема полученных результатов.

В наши дни в связи с быстрым увеличением мощности ПК необходимость в подобной услуге сократилась, но, тем не менее, службы Telnet в Internet продолжают существовать. Часто протоколы Telnet применяют для дистанционного управления техническими объектами, например: телескопами, видеокамерами, промышленными роботами, автоматизированными складами и даже торговыми автоматами.

6.1.1. 5.5.1. Электронная почта (E-Mail)

Эта служба также является одной из наиболее ранних. Ее обеспечением в Internet занимаются специальные почтовые серверы.

Почтовые серверы получают сообщения от клиентов и пересылают их по цепочке к почтовым серверам адресатов, где эти сообщения накапливаются. При установлении соединения между адресатом и его почтовым сервером происходит автоматическая передача поступивших сообщений на компьютер адресата.

Почтовая служба основана на двух прикладных протоколах: *SMTP* и *POP3*. По первому происходит отправка корреспонденции с компьютера на сервер, а по второму – прием поступивших сообщений. Существует большое разнообразие клиентских почтовых программ. К ним относится,

например, программа *Microsoft Outlook Express*. Из специализированных почтовых программ хорошую популярность имеют программы *The Bat!*, *Eudora*, *Pegasus mail*.

6.1.2. Служба телеконференций (Usenet)

Служба телеконференций похожа на циркулярную рассылку электронной почты, в ходе которой одно сообщение отправляется не одному корреспонденту, а большой группе (такие группы называются *телеконференциями* или *группами новостей*).

Обычное сообщение электронной почты пересылается по узкой цепочке серверов от отправителя к получателю. При этом не предполагается его хранение на промежуточных серверах. Сообщения, направленные на сервер группы новостей, отправляются с него на все серверы, с которыми он связан, если на них данного сообщения еще нет. Далее процесс повторяется.

На каждом из серверов поступившее сообщение хранится ограниченное время (обычно неделю), и все желающие могут в течение этого времени с ним ознакомиться.

Вся система телеконференций разбита на тематические группы. Сегодня в мире насчитывают порядка 50 000 тематических групп новостей. Они охватывают большинство тем, интересующих массы.

Основной прием использования групп новостей состоит в том, чтобы задать вопрос, обращаясь ко всему миру, и получить ответ или совет от тех, кто с этим вопросом уже разобрался. При этом важно следить за тем, чтобы содержание вопроса соответствовало теме данной телеконференции. При отправке сообщений в телеконференции принято указывать свой адрес электронной почты для обратной связи.

Для работы со службой телеконференций существуют специальные клиентские программы. Так, например, приложение *Microsoft Outlook Express*, указанное выше как почтовый клиент, позволяет работать также и со службой телеконференций. Для начала работы надо настроить программу на взаимодействие с сервером групп новостей, оформить «подписку» на определенные группы и периодически, как и электронную почту, получать все сообщения, проходящие по теме этой группы. В данном случае слово «подписка» не предполагает со стороны клиента никаких обязательств или платежей – это просто указание серверу о том, что сообщения по указанным темам надо доставлять, а по прочим – нет. Отменить подписку или изменить ее состав можно в любой удобный момент.

6.1.3. Служба World Wide Web (WWW)

World Wide Web – это единое *информационное пространство*, состоящее из взаимосвязанных электронных документов, хранящихся на *Web-серверах*.

Web-страница. Отдельные документы, составляющие *пространство Web* называют *Web-страницами*. Обычно это комбинированный документ, который может содержать текст, графические иллюстрации, мультимедийные и другие объекты. Для создания Web-страниц используется язык *HTML* (*HyperText Markup Language – язык гипертекстовой разметки*), который при помощи вставленных в документ *тегов* описывает логическую структуру документа, управляет форматированием текста и размещением различных объектов. От обычного текста теги отличаются тем, что заключены в угловые скобки. Большинство тегов используются парами: открывающий тег и закрывающий. Закрывающий тег начинается с символа «/»:

```
<CENTER> Этот текст должен выравниваться по центру экрана </CENTER>
<P ALIGN = "LEFT"> Этот текст выравнивается по левой границе экрана </P>
<P ALIGN = "RIGHT"> Этот текст выравнивается по правой границе экрана </P>
```

Сложные теги имеют кроме ключевого слова дополнительные *атрибуты* и *параметры*, детализирующие способ их применения. Таким образом, Web-документ представляет собой обычный текстовый документ, размеченный тегами *HTML*. Такие документы также называют *HTML-документами* или документами в формате *HTML*. При просмотре *HTML*-документа на экране с помощью *браузера* (см. ниже) теги не отображаются, и пользователь видит только текст, составляющий документ.

Web-узел. Группы тематически объединенных Web-страниц называют *Web-узлами*. Один физический Web-сервер может содержать достаточно много Web-узлов, каждому из которых, как правило, отводится отдельный каталог на жестком диске сервера.

Web-каналы. Обычный Web-узел выдает информацию (запрошенный документ) только в ответ на обращение клиента. Чтобы следить за обновлением опубликованных материалов, пользователь вынужден регулярно обращаться к данному узлу. Современная модель Web-узла позволяет автоматически в заданное время передать обновленную информацию на компьютер зарегистрированного клиента. Такие Web-узлы, способные самостоятельно инициировать поставку информации, называют *каналами*.

Гиперссылки. Отличительной особенностью среды WWW является наличие средств перехода от одного документа к другому, тематически с ним связанному, без явного указания адреса. Связь между документами осуществляется при помощи *гиперссылок*. *Гиперссылка* – это выделенный

фрагмент документа (текст или иллюстрация), с которым ассоциирован адрес другого Web-документа. При использовании гиперссылки (обычно для этого требуется навести на нее указатель мыши и один раз щелкнуть) происходит переход по гиперссылке – открытие Web-страницы, на которую указывает ссылка. Перемещение между Web-документами называют *Web-навигацией*. Механизм гиперссылок позволяет организовать тематическое путешествие по WWW без использования адресов конкретных страниц.

Средства просмотра Web. Программы для просмотра Web-страниц называют *браузерами*. Браузер выполняет отображение документа на экране, руководствуясь тегами, которые автор документа внедрил в его текст. Основные функции браузеров следующие:

– установление связи с Web-сервером, на котором хранится документ, и загрузка всех компонентов комбинированного документа;

– интерпретация тегов языка *HTML*, форматирование и отображение Web-страницы в соответствии с возможностями компьютера, на котором браузер работает;

– предоставление средств для отображения мультимедийных и других объектов, входящих в состав Web-страниц, а также механизма расширения, позволяющего настраивать программу на работу с новыми типами объектов;

– обеспечение автоматизации поиска Web-страниц и упрощение доступа к Web-страницам, посещавшимся ранее;

– предоставление доступа к встроенным или автономным средствам для работы с другими службами Internet.

Адресация документов. Гипертекстовая связь между миллиардами документов, хранящихся на серверах Internet, является основой существования пространства WWW. Однако такая связь не могла бы существовать, если бы каждый документ в этом пространстве не обладал своим уникальным адресом. Адрес любого файла в глобальной сети Internet определяется *унифицированным указателем ресурса – URL*.

Адрес *URL* состоит из трех частей, разделенных точками:

1) указание службы, которая осуществляет доступ к данному ресурсу (обычно обозначается именем протокола, соответствующего данной службе). Так, например, для службы *WWW* таковым является протокол *HTTP* (*HyperText Transfer Protocol – протокол передачи гипертекста*). После имени протокола ставится двоеточие (:) и два знака «/» (слеш):

http://...

2) указание *доменного имени* компьютера (сервера), на котором хранится данный ресурс:

http://*www.abcde.com*…

3) указание полного пути доступа к файлу на данном компьютере. В качестве разделителя используется символ «/»:

http://www.abcde.com/*Files/New/abcdefg.zip*

При щелчке на гиперссылке, содержащей *URL*, браузер посылает запрос для поиска и доставки ресурса, указанного в ссылке. Если по каким-то причинам он не найден, выдается сообщение о том, что ресурс недоступен (возможно, что сервер временно отключен или изменился адрес ресурса).

Преобразование адреса *URL* в цифровую форму *IP*-адреса производит служба имен доменов (*DNS*).

6.1.4. Служба имен доменов (DNS)

Человеку неудобно работать с числовым представлением *IP*-адреса, зато доменное имя запоминается легко, потому что, как правило, это имя содержательное.

Автоматическая работа серверов сети организована с использованием числового адреса. Благодаря ему промежуточные серверы могут осуществлять передачу запросов и ответов в нужном направлении, не зная, где именно находятся отправитель и получатель. Поэтому необходим перевод доменных имен в связанные с ними *IP*-адреса. Этим и занимаются серверы службы имен доменов *DNS* (*Domain Name System*). Запрос на получение одной из страниц сервера *www.abcde.com* (условный адрес) сначала обрабатывается сервером *DNS*, и далее он направляется по *IP*-адресу, а не по доменному имени.

6.1.5. Служба передачи файлов (FTP)

Прием и передача файлов составляют значительный процент от прочих Internet-услуг. Необходимость в передаче файлов возникает, например, при приеме файлов программ, при пересылке крупных документов, а также при передаче архивных файлов, в которых запакованы большие объемы информации.

Служба *FTP* имеет свои серверы в мировой сети, на которых хранятся архивы данных. Со стороны клиента для работы с серверами *FTP* может быть установлено специальное программное обеспечение, хотя в большинстве случаев браузеры *WWW* обладают встроенными возможностями, реализующими простейшие операции протокола *FTP*, например, загрузку файлов с сервера.

Протокол *FTP* работает одновременно с двумя *TCP*-соединениями между сервером и клиентом. По одному соединению идет передача данных, а второе соединение используется как управляющее. Протокол *FTP* также предоставляет серверу средства для идентификации обратившегося клиента.

Этим часто пользуются коммерческие серверы и серверы ограниченного доступа, поставляющие информацию только зарегистрированным клиентам, – они выдают запрос на ввод имени пользователя и связанного с ним пароля. Однако существуют и десятки тысяч *FTP*-серверов с анонимным доступом для всех желающих. В этом случае в качестве имени пользователя надо ввести слово anonymous, а в качестве пароля задать адрес электронной почты. В большинстве случаев программы-клиенты *FTP* делают это автоматически.

6.1.6. ICQ

Эта служба предназначена для поиска сетевого *IP*-адреса человека, подключенного в данный момент к Internet. Необходимость в подобной услуге связана с тем, что большинство пользователей не имеют постоянного *IP*-адреса. Название службы является акронимом выражения *I seek you – я тебя ищу*. Для пользования этой службой надо зарегистрироваться на ее центральном сервере (*http://www.icq.com*) и получить персональный идентификационный номер *UIN* (*Universal Internet Number*). При каждом подключении к Internet программа *ICQ*, установленная на компьютере, определяет текущий *IP*-адрес и сообщает его центральной службе, которая, в свою очередь, оповещает партнеров по контактам. Далее партнеры (если они тоже являются клиентами данной службы) могут установить с нужным пользователем прямую связь. Программа предоставляет возможность выбора режима связи («готов к контакту»; «прошу не беспокоить, но готов принять срочное сообщение»; «закрыт для контакта» и т. п.). После установления контакта происходит прямое общение в режиме реального времени.

Кроме того, номер *UIN* можно сообщить партнерам по контактам, и тогда служба *ICQ* приобретает характер Internet-пейджера. Зная номер *UIN* партнера, но не зная его текущий *IP*-адрес, можно через центральный сервер службы отправить ему сообщение с предложением установить соединение.

6.6. Поиск информации в Internet

6.1.7. Браузер Microsoft Internet Explorer

Как уже отмечалось выше, приложение, посредством которого выполняется просмотр *WWW*, называется *Web-браузером*. Наиболее известные браузеры – Internet Explorer (компания Microsoft), Netscape Navigator (компания Netscape Communications) и Opera (компания Opera Software).

Помимо *WWW*, браузеры допускают обращение к другим службам Internet (телеконференции Usenet, файловым архивам *FTP* и др.)

Для изучения выберем браузер Internet Explorer из соображений его особой распространенности.

Для запуска браузера Internet Explorer можно использовать ярлык Internet Explorer на Рабочем столе или Главное меню (*Пуск→Программы→Internet Explorer*). Кроме того, программа запускается автоматически при попытке открыть документ Internet или локальный документ в формате *HTML*.

Если соединение с Internet отсутствует, то после запуска программы на экране появится диалоговое окно для управления установкой соединения. При невозможности установить соединение сохраняется возможность просмотра в *автономном режиме* ранее загруженных Web-документов. При наличии соединения, после запуска программы, на экране появится так называемая *начальная* страница, выбранная при настройке программы.

Открытие и просмотр Web-страниц. Просматриваемая Web-страница отображается в рабочей области окна Internet Explorer. По умолчанию воспроизводится все ее содержимое, включая графические иллюстрации и встроенные мультимедийные объекты. Управление просмотром осуществляется при помощи строки меню, панелей инструментов, а также активных элементов, имеющихся в открытом документе, например *гиперссылок*.

Если *URL*-адрес Web-страницы известен, его можно ввести в строке *Адрес* и нажать клавишу `Enter`. Страница с указанным адресом открывается вместо текущей. Наличие средства автозаполнения адресной строки упрощает повторный ввод адресов. Вводимый адрес автоматически сравнивается с адресами ранее просматривавшихся Web-страниц. Все подходящие адреса отображаются в раскрывающемся списке строки *Адрес*. Если нужный адрес есть в списке, его можно выбрать клавишами ↑ и ↓, после чего нажать клавишу `Enter`. При отсутствии нужного адреса ввод продолжают как обычно.

Навигация в Internet чаще выполняется не путем ввода адреса *URL*, а посредством использования *гиперссылок*. При отображении Web-страницы на экране, гиперссылки выделяются цветом (обычно синим) и подчеркиванием. При наведении указателя мыши на гиперссылку он принимает форму кисти руки, а сама гиперссылка при соответствующей настройке браузера изменяет цвет. Адрес *URL*, на который указывает ссылка, отображается в строке состояния. При щелчке на гиперссылке соответствующая Web-страница загружается вместо текущей. Если гиперссылка указывает на произвольный файл, его загрузка происходит по протоколу *FTP*.

На Web-страницах могут также встречаться графические ссылки (гиперссылки, представленные рисунком) и изображения-карты, объединяющие несколько ссылок в рамках одного изображения.

Дополнительные возможности использования гиперссылок предоставляет их контекстные меню. Чтобы открыть новую страницу, не закрывая текущей, применяют команду *Открыть в новом окне*. В результате открывается новое окно браузера. Адрес *URL*, заданный ссылкой, можно поместить в буфер обмена при помощи команды *Копировать ярлык*. Его можно вставить в поле строки *Адрес* или в любой другой документ для последующего использования.

Другие операции, относящиеся к текущей странице и ее элементам, также удобно осуществлять через контекстное меню. Например, рисунок, имеющийся на странице, можно:

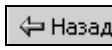
– сохранить как файл (*Сохранить рисунок как*);

– использовать как фоновый рисунок (*Сделать рисунком рабочего стола*) или как активный элемент (*Сохранить как элемент рабочего стола*).

Если рисунок выполняет функции графической ссылки, к нему можно применять как команды, относящиеся к изображению, так и команды, относящиеся к ссылке.

Приемы управления браузером. Необходимость определенных действий в ходе просмотра документов WWW часто диктуется самим ходом работы. В таких случаях удобно использовать кнопки панели инструментов *Обычные кнопки* (рис. 5.1).

Рис. 5.1. Панель инструментов *Обычные кнопки*

Для того чтобы вернуться к странице, которая просматривалась некоторое время назад, используют кнопку *Назад* 

. Чтобы возвратиться на несколько страниц назад, можно использовать присоединенную к ней кнопку раскрывающегося списка. Отменить действия, выполненные при помощи кнопки *Назад*, позволяет кнопка *Вперед*  .

Если процесс загрузки страницы затянулся или необходимость в ней отпала, используют кнопку *Остановить*  . Заново загрузить Web-страницу, если ее загрузка была прервана или содержание документа изменилось, позволяет кнопка *Обновить*  . Чтобы немедленно загрузить начальную страницу, с которой браузер обычно начинает работу, пользуются кнопкой *Домой*  .

Создать новое окно, сохранить открытый документ на своем компьютере, распечатать его, включить или выключить режим автономной работы, а также завершить работу с программой позволяют команды меню *Файл*.

Копирование фрагментов документа в буфер обмена и поиск текста на Web-странице осуществляются при помощи команд меню *Правка*.

Включение и выключение отображения служебных элементов окна (панелей инструментов, дополнительных панелей, строки состояния), выбор шрифта и кодировки символов осуществляются через меню *Вид*.

Ведение списка регулярно посещаемых страниц и быстрый доступ к ним осуществляются через меню *Избранное*. Переход к использованию программ для работы с другими службами Internet, а также настройка браузера осуществляются через меню *Сервис*.

Работа с несколькими окнами. Нередко возникает необходимость открыть новый Web-документ, не закрывая текущий, например, в тех случаях, когда текущий документ содержит список интересных ссылок. Чтобы открыть новое окно программы Internet Explorer, используют команду меню *Файл→Создать→Окно*. Каждое окно отображает свой Web-документ и может использоваться самостоятельно. В частности, списки кнопок *Назад* и *Вперед* обновляются в каждом окне индивидуально. Закрывать окна программы Internet Explorer можно в любом порядке.

Прием файлов из Internet. Гиперссылки, имеющиеся на Web-страницах, могут указывать на документы разных типов. Если браузер не способен отображать файлы определенного типа (например, исполняемые файлы с расширением *.exe*, архивы *.zip* и пр.), инициируется процесс загрузки данного файла на компьютер.

Программа Internet Explorer запускает мастер загрузки файла, на первом этапе работы которого требуется указать, следует ли открыть файл или сохранить его на диске. Открытие файла подразумевает загрузку его в каталог временных файлов и немедленный запуск (если это исполняемый файл) или открытие с помощью программы, которая предназначена для работы с файлами этого типа. Такой подход открывает путь на компьютер для небезопасной информации. Надежнее выбрать сохранение файла на диске. В этом случае требуется выбрать папку, в которой следует сохранить файл.

Ход загрузки файла отображается в специальном окне загрузки. Процесс загрузки файла не препятствует параллельному просмотру Web-страниц или другим операциям в Internet.

Загрузку файла можно прервать в любой момент при помощи кнопки *Отмена*.

6.1.8. Рекомендации по поиску информации

Web-пространство отличается гигантскими размерами. Найти в этом пространстве именно то, что нужно, представляется весьма сложной зада-

чей. Помощь в поиске нужной информации оказывают разнообразные поисковые системы.

Поисковая система представляет собой специализированный Web-узел. Пользователь сообщает поисковой системе данные о содержании искомой Web-страницы, а система выдает ему список гиперссылок на страницы, соответствующие запросу. Наибольшую популярность приобрели это поисковые каталоги и поисковые указатели.

Поисковые каталоги. Поисковые каталоги устроены по тому же принципу, что и тематические каталоги крупных библиотек. На основной странице поискового каталога размещен сокращенный список крупных тематических категорий. Каждая запись в списке категорий – это гиперссылка. Щелчок на ней открывает следующую страницу поискового каталога, на котором данная тема представлена подробнее. Щелчок на названии темы открывает страницу со списком разделов. Продолжая погружение в тему, можно дойти до списка конкретных Web-страниц и выбрать себе тот ресурс, который лучше подходит для решения задачи.

Поисковые машины. В поисковых машинах происходит автоматический просмотр, отбор и сортировка новых Web-страниц без участия человека. Основной принцип работы поисковой машины заключается в поиске Web-ресурсов по *ключевым словам*. Пользователь описывает искомый ресурс с помощью ключевых слов, после чего дает задание на поиск. Поисковая система анализирует данные, хранящиеся в своей базе, и выдает список Web-страниц, соответствующих запросу. Вместе с гиперссылками выдаются краткие сведения о найденных ресурсах, на основании которых пользователь может выбрать нужные ему ресурсы.

Сегодня в мире существует огромное количество поисковых машин. Вершину списка занимают около двух десятков зарубежных систем: Alta Vista (www.atavista.com), Excite (www.excite.com), Fast Search (www.alltheweb.com), Go/Infoseek (www.go.com), Google (www.google.com), HotBot (hotbot.lycos.com), Lycos (www.lycos.com), Netscape Search (search netscape.com), WebCrawler (www.webcrawler.com) и др. В России также имеется несколько поисковых указателей, из которых наиболее крупными и популярными являются следующие: Апорт2000 (www.aport.ru), Яndex (www.yandex.ru) и Rambler (www.rambler.ru).

Рекомендации по приемам эффективного поиска. Для проведения реферативного поиска, когда тема задана достаточно широко, рекомендуется пользоваться поисковыми каталогами, такими, как Yahoo! (www.yahoo.com) или «Атрус» (atrus.aport.ru). Это позволит быстро установить местоположение основных первоисточников. При ознакомлении с первоисточниками следует, прежде всего, уделять внимание понятийной базе. Знание основных понятий и

терминов позволит перейти к углубленному поиску в поисковых указателях с использованием ключевых слов, наиболее точно характеризующих тему.

При наличии первичных сведений по теме поиска, документы можно разыскивать с помощью поисковых машин. При этом следует различать приемы *простого*, *расширенного*, *контекстного* и *специального поиска*.

Под *простым поиском* понимается поиск Web-ресурсов по одному или нескольким ключевым словам. Недостаток простого поиска заключается в том, что обычно он выдает слишком много документов, среди которых трудно выбрать наиболее подходящие.

При использовании *расширенного поиска* ключевые слова связывают между собой операторами логических отношений. Расширенный поиск применяют в тех случаях, когда приемы простого поиска дают слишком много результатов. С помощью логических отношений поисковое задание формируют так, чтобы более точно детализировать задание и ограничить область отбора, например, по дате публикации или по типу данных.

Контекстный поиск – это поиск по точной фразе. Он удобен для реферативного поиска информации, но доступен далеко не во всех поисковых системах.

Специальный поиск применяют при розыске Web-страниц, содержащих ссылки на заданные адреса *URL*, а также содержащих заданные данные в служебных полях, например, в поле заголовка.

Рекомендации по использованию поисковых систем. Для проведения научных поисков рекомендуется пользоваться поисковой системой Northern Light (www.northernlight.com). Эта система имеет один из лучших коэффициентов охвата Web-пространства, и ее администрация прилагает специальные усилия для поддержания актуальности своих указателей. Кроме того, система удачно сочетает свойства поискового каталога и поисковой машины. По наиболее популярным темам в ней можно найти специальные разделы каталожного типа – они называются *Special Editions* и подготавливаются вручную. Дополнительно система предоставляет платные услуги по поставке актуальных научных документов. Они находятся в разделе *Special Collection*.

Самым большим поисковым указателем обладает поисковая система Fast Search (www.alltheweb.com). Всего за один год после запуска эта поисковая система вышла на первое место в мире по объему проиндексированного пространства.

Исторически одной из наиболее популярных считается поисковая система Alta Vista (www.altavista.com).

В России в настоящее время действуют три примерно одинаковых по мощности поисковых указателя: Апорт2000 (www.aport.ru), Rambler

(www.rambler.ru) Яndex (www.yandex.ru). Все они обладают примерно одинаковым «знанием» о ресурсах российского сектора *WWW* и работают достаточно быстро. Систему Апорт2000 удобно использовать в операциях простого поиска. В этой системе приняты специальные меры по устранению дубликатов, удалению неактуальных ссылок, наглядному представлению результатов поиска. Система Rambler по своей сути является не только поисковой, но и выполняет функции удобного Web-портала. Систему Яndex удобно использовать при формировании сложных поисковых заданий, поскольку она обладает наиболее гибким языком для расширенного поиска.

ЭКЗАМЕНАЦИОННЫЕ ВОПРОСЫ

1. Охарактеризуйте основные понятия информатики.
2. Охарактеризуйте виды и свойства информации.
3. Что собой представляют процессы восприятия, сбора, передачи, обработки и накопления информации?
4. Охарактеризуйте этапы развития информатики.
5. Классификация ЭВМ по назначению, уровню специализации, типоразмерам и совместимости.
6. Кодирование данных в ЭВМ двоичным кодом.
7. Системы счисления.
8. Единицы измерения информации.
9. Базовая конфигурация персонального компьютера.
10. Носители информации.
11. Периферийные устройства персонального компьютера.
12. Программное обеспечение средств вычислительной техники.

13. Назначение, классификация и основные функции операционных систем.
14. Общая характеристика операционной системы Windows.
15. Основные элементы графического интерфейса Windows: рабочий стол Windows, значки и ярлыки объектов, окна.
16. Файловая система и файловая структура Windows.
17. Что такое файл? Какие требования предъявляются к именам файлов? Из каких элементов состоит полное имя файла? Что собой представляют атрибуты файла?
18. Перечислите основные операции с файловой структурой. Какие средства существуют в Windows для работы с файловой структурой?
19. Главное меню Windows: его назначение и структура.
20. Обмен данными в Windows.
21. Стандартные приложения Windows.
22. Программы технического обслуживания.
23. Программы-архиваторы.
24. Антивирусные программы.

25. Понятие и основные свойства алгоритма.
26. Способы записи алгоритмов.
27. Разновидности структур алгоритмов.
28. Программирование. Языки программирования высокого уровня.
29. Основные понятия объектно-ориентированного программирования.
30. Основы теории погрешностей.

31. Приближенное решение нелинейных уравнений.
32. Методы решения систем линейных уравнений.
33. Приближенное вычисление определенных интегралов.
34. Приближенное решение дифференциальных уравнений.
35. Интерполирование функций.
36. Регрессия.

37. Пакет Mathcad: общая характеристика, пользовательский интерфейс.
38. Пакет Mathcad: базовые элементы входного языка.
39. Пакет Mathcad: основные виды вычислений, численные и символьные вычисления.
40. Пакет Mathcad: графический анализ данных.
41. Пакет Mathcad: работа с массивами данных (вектора, матрицы).
42. Пакет Mathcad: использование программных фрагментов, программирование основных типов алгоритмов.
43. Пакет Mathcad: ввод и вывод данных различных типов.
44. Пакет Mathcad: решение уравнений.
45. Пакет Mathcad: выполнение символьных вычислений.
46. Пакет Mathcad: нахождение экстремумов функций
47. Пакет Mathcad: аппроксимация функций
48. Пакет Mathcad: вычисление определенных интегралов
49. Пакет Mathcad: решение дифференциальных уравнений
50. Пакет Mathcad: решение уравнений в частных производных.

51. Назначение и возможности текстового процессора Microsoft Word.
52. Режимы просмотра документов в Microsoft Word.
53. Установка параметров страницы. Ввод и редактирование текста.
54. Форматирование текста. Использование стилей при создании документа.
55. Верстка многостраничного документа.
56. Буфер обмена Microsoft Office.
57. Списки в Microsoft Word.
58. Создание и редактирование таблиц.
59. Создание и редактирование художественных надписей (WordArt).
60. Рисунки в Microsoft Word. Печать документа.

61. Назначение и возможности табличного процессора Microsoft Excel.
62. Ввод и редактирование данных. Ввод последовательностей данных в ячейки. Настройка Автозаполнения.

63. Организация вычислений: автоматическое суммирование строк и столбцов; составление формул; формулы с относительными и абсолютными адресами.
64. Диаграммы.
65. Сортировка данных.
66. Подведение итогов.
67. Подбор параметра.
68. Таблицы подстановки данных.

69. Назначение компьютерных сетей.
70. Локальные и глобальные сети. Основные понятия.
71. Службы Internet: Telnet, электронная почта (E-Mail), служба телеконференций (Usenet).
72. Службы Internet: World Wide Web (WWW).
73. Работа с гипертекстовой информацией в сети Internet.
74. Службы Internet: служба имен доменов (DNS), служба передачи файлов (FTP), ICQ.
75. Поиск информации в Internet с помощью поисковых систем.

76. Растровая графика.
77. Векторная и фрактальная графика.
78. Программные средства для работы с графикой.
79. Средства подготовки презентаций.
80. Microsoft PowerPoint как средство создания презентаций, его основные возможности.
81. Воспроизведение презентаций: презентации на экране и в Internet.

ЛИТЕРАТУРА

1. Информатика: Базовый курс / Под ред. С.В. Симоновича. – СПб.: Питер, 2001.
2. Справочное пособие по приближённым методам решения задач высшей математики / Л.И. Бородич, А.И. Герасимович. Н.П. Кеда, И.Н. Мелешко. – Мн.: Высш. Шк., 1986.
3. Офицеров Д.В. и др. Программирование на персональных ЭВМ: Практикум: Учеб. пособие. – Мн.: Выш. Шк., 1993. –256с.
4. Дьяконов, В.П. MathCad 2001: учеб. курс /В.П. Дьяконов. - СПб.: Питер, 2001.
5. Очков, В.Ф. MathCad 8 Pro для студентов и инженеров / В.Ф. Очков. - М., 1999.
6. Основы информатики и вычислительной техники: Учеб.-метод. комплекс для студ. спец. 1-25 01 08, 1-25 01 04 / Сост. и общ. ред. С.Е. Рясовой. – Новополоцк: ПГУ, 2005. – 340 с.
7. Программирование: Учеб.-метод. комплекс для студ. спец. 1-39 01 01 «Радиотехника» / Сост. Л.В. Малухиной, А.С. Барышникова. Под общ. ред. Л.В. Малухиной. – Новополоцк: «УО» ПГУ, 2006. – 288 с.

Учебное издание

Составитель
СПИРИДОНОВ Александр Владимирович

ИНФОРМАТИКА И КОМПЬЮТЕРНАЯ ГРАФИКА

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС

для студентов специальности

48 01 03 «Химическая технология природных энергоносителей и углеродных материалов»

Редакторы

Подписано в печать . Формат 60×84 1/16. Бумага офсетная. Гарнитура Таймс.
Отпечатано на ризографе. Усл. печ. л. 19,72. Уч.-изд. л. 17,31. Тираж 100. Заказ

Издатель и полиграфическое исполнение
Учреждение образования «Полоцкий государственный университет»

ЛИ ЛП №

211440 г. Новополоцк, ул. Блохина, 29