Под редакцией
С. В. Симоновича

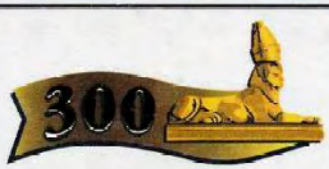
ПИТЕР

ИНФОРМАТИКА

БАЗОВЫЙ КУРС

2-е издание

УЧЕБНИК ДЛЯ ВУЗОВ



для студентов технических специальностей и преподавателей высших учебных заведений

фундаментальный курс, охватывающий все основные разделы информатики

УЧЕБНИК ДЛЯ ВУЗОВ

Под редакцией
С. В. Симоновича

ИНФОРМАТИКА

БАЗОВЫЙ КУРС

2-е издание

Рекомендовано Министерством образования Российской Федерации
в качестве учебного пособия для студентов
высших технических учебных заведений

300.piter.com

Издательская программа

300 лучших учебников для высшей школы
в честь 300-летия Санкт-Петербурга

осуществляется при поддержке Министерства образования РФ

ПИТЕР®

Москва · Санкт-Петербург · Нижний Новгород · Воронеж
Новосибирск · Ростов-на-Дону · Екатеринбург · Самара
Киев · Харьков · Минск · Новосибирск
2005

ББК 32.973.233я7
УДК 681.3(075)
С37

Рецензенты:

Кафедра САПР Московского государственного технического университета им. Н. Э. Баумана
Калин С. В., генеральный директор ЗАО «Открытые технологии '98»

С37 **Информатика. Базовый курс. 2-е издание** / Под ред. С. В. Симоновича. — СПб.: Питер, 2005. — 640 с.: ил.

ISBN 5-94723-752-0

В учебнике рассмотрены основные категории аппаратных и программных средств вычислительной техники. Указаны базовые принципы построения архитектур вычислительных систем. Обеспечено методическое обоснование процессов взаимодействия информации, данных и методов. Приведены эффективные приемы работы с распространенными программными продуктами. Рассмотрены основные средства, приемы и методы программирования.

Книга предназначена для студентов технических вузов, изучающих информационные технологии в рамках дисциплины «Информатика», для преподавательского состава, для слушателей военных учебных заведений, учреждений системы повышения квалификации и для лиц, изучающих средства вычислительной техники самостоятельно.

Рекомендовано Министерством образования Российской Федерации в качестве учебного пособия для студентов высших технических учебных заведений.

ББК 32.973.233я7
УДК 681.3(075)

ISBN 5-94723-752-0

Содержание

Введение 8
Глава 1. Информация и информатика 11
1.1. Информация в материальном мире 11
1.2. Данные 17
1.3. Файлы и файловая структура 31
1.4. Информатика 34
Подведение итогов 36
Вопросы для самоконтроля 37
Глава 2. Вычислительная техника 38
2.1. История развития средств вычислительной техники 38
2.2. Методы классификации компьютеров 42
2.3. Состав вычислительной системы 49
Подведение итогов 60
Вопросы для самоконтроля 61
Глава 3. Устройство персонального компьютера 62
3.1. Базовая аппаратная конфигурация персонального компьютера 62
3.2. Внутренние устройства системного блока 70
3.3. Системы, расположенные на материнской плате 78
3.4. Периферийные устройства персонального компьютера 87
Практическое занятие 94
Глава 4. Функции операционных систем персональных компьютеров 99
4.1. Обеспечение интерфейса пользователя 99
4.2. Обеспечение автоматического запуска 100
4.3. Организация файловой системы 101
4.4. Обслуживание файловой структуры 102

4.5. Управление установкой, исполнением и удалением приложений 107
4.6. Взаимодействие с аппаратным обеспечением 109
4.7. Обслуживание компьютера 110
4.8. Прочие функции операционных систем 113
Подведение итогов 114
Вопросы для самоконтроля 115
Глава 5. Основы работы с операционной системой Windows XP 116
5.1. Основные объекты и приемы управления Windows 116
5.2. Файлы и папки Windows 119
5.3. Операции с файловой структурой 122
5.4. Использование Главного меню 129
5.5. Установка и удаление приложений Windows 129
5.6. Установка оборудования 132
Практическое занятие 134
Исследовательская работа 139
Глава 6. Настройка операционной системы Windows XP 141
6.1. Настройка средств ввода-вывода данных 142
6.2. Настройка элементов оформления Windows XP 143
6.3. Настройка элементов управления Windows XP 147
6.4. Настройка средств автоматизации Windows XP 150
6.5. Настройка шрифтов 156
6.6. Прочие настройки Windows XP 160
6.7. Справочная система Windows XP 162
Практическое занятие 164
Самостоятельная работа 168
Глава 7. Стандартные приложения Windows XP 169
7.1. Стандартные прикладные программы 169
7.2. Принципы внедрения и связывания объектов 181
7.3. Служебные приложения Windows XP 183
7.4. Стандартные средства мультимедиа 187
Практическое занятие 189
Глава 8. Компьютерные сети, Интернет, компьютерная безопасность ... 195
8.1. Компьютерные сети 195
8.2. Интернет. Основные понятия 201
8.3. Подключение к Интернету 213
8.4. Вопросы компьютерной безопасности 215
Практическое занятие 224

Глава 9. Получение информации из Интернета ... 227
9.1. Основные понятия World Wide Web ... 227
9.2. Работа с программой Internet Explorer 6.0 ... 228
9.3. Поиск информации в World Wide Web ... 236
9.4. Отправка и получение сообщений ... 243
Практическое занятие ... 247
Глава 10. Создание простых текстовых документов ... 253
10.1. Общие сведения о текстовом процессоре Microsoft Word ... 253
10.2. Приемы работы с текстами в процессоре Microsoft Word ... 262
10.3. Приемы и средства автоматизации разработки документов ... 274
Практическое занятие ... 279
Глава 11. Создание комплексных текстовых документов ... 285
11.1. Приемы управления объектами Microsoft Word ... 285
11.2. Ввод формул ... 294
11.3. Работа с таблицами ... 296
11.4. Работа с диаграммами ... 299
11.5. Работа с графическими объектами ... 302
Практическое занятие ... 309
Глава 12. Обработка данных средствами электронных таблиц ... 315
12.1. Основные понятия электронных таблиц ... 316
12.2. Содержание электронной таблицы ... 318
12.3. Печать документов Excel ... 323
12.4. Применение электронных таблиц для расчетов ... 325
12.5. Построение диаграмм и графиков ... 328
Практическое занятие ... 330
Глава 13. Работа с базами данных ... 340
13.1. Основные понятия баз данных ... 340
13.2. Формирование баз данных ... 345
13.3. Работа с СУБД Microsoft Access 2002 ... 353
Практическое занятие ... 367
Глава 14. Приемы и методы работы со сжатыми данными ... 375
14.1. Теоретические основы сжатия данных ... 375
14.2. Программные средства сжатия данных ... 379
14.3. Программные средства уплотнения носителей ... 382
Практическое занятие ... 384
Исследовательская работа ... 394

Глава 15. Введение в компьютерную графику ... 398
15.1. Основы представления графических данных ... 398
15.2. Представление графических данных ... 413
Практическое занятие ... 423
15.3. Средства для работы с растровой графикой ... 425
15.4. Средства для работы с векторной графикой ... 432
Практическое занятие ... 437
Исследовательская работа ... 441
Практическое занятие ... 442
Исследовательская работа ... 446
Глава 16. Векторный редактор CorelDraw ... 449
16.1. Особенности CorelDraw ... 449
16.2. Элементы управления ... 450
16.3. Рисование графики ... 458
16.4. Заполнение объектов ... 464
16.5. Операции с текстом ... 469
16.6. Изменение формы объектов ... 472
16.7. Операции с группами ... 475
Пример. Рисование топографической карты ... 482
Практическое занятие ... 482
Глава 17. Автоматизация обработки документов ... 488
17.1. Преобразование документов в электронную форму ... 488
Практическое занятие ... 495
17.2. Автоматизированный перевод документов ... 498
Практическое занятие ... 506
Глава 18. Средства автоматизации научно-исследовательских работ ... 509
18.1. Компьютер как инструмент научной работы ... 509
18.2. Приемы работы с системой Mathcad ... 513
Практическое занятие ... 521
Глава 19. Публикация Web-документов ... 537
19.1. Создание Web-документов ... 537
19.2. Применение языка HTML ... 539
19.3. Работа в редакторе FrontPage ... 552
19.4. Публикация Web-документов ... 557
Практическое занятие ... 558
Исследовательская работа ... 566

Глава 20. Основы программирования 568
20.1. Языки программирования 568
20.2. Системы программирования 578
20.3. Алгоритмическое (модульное) программирование 582
20.4. Структурное программирование 599
20.5. Объектно-ориентированное программирование 605
20.6. Проектирование программ 608
20.7. Пример на Бейсике. Разведение кроликов 616
20.8. Пример на Паскале. Раскрашивание круга 621
20.9. Пример на Си++. Рисование графиков 626
Практические задания по программированию 629
Рекомендуемая литература 631
Алфавитный указатель 633

Введение

Коренное отличие информатики от других технических дисциплин, изучаемых в высшей школе, состоит в том, что ее предмет изучения меняется ускоренными темпами. Сегодня количество компьютеров в мире превышает 500 миллионов единиц, при этом каждая вычислительная система по-своему уникальна. Найти две системы с одинаковыми аппаратными и программными конфигурациями весьма сложно, и потому для эффективной эксплуатации вычислительной техники от специалистов требуется достаточно широкий уровень знаний и практических навыков.

Вместе с тем, в количественном отношении темп численного роста вычислительных систем заметно превышает темп подготовки специалистов, способных эффективно работать с ними. При этом в среднем один раз в полтора года удваиваются основные технические параметры аппаратных средств, один раз в два-три года меняются поколения программного обеспечения и один раз в пять-семь лет меняется база стандартов, интерфейсов и протоколов.

Таким образом, кардинальным отличием информатики от других технических дисциплин является тот факт, что ее предметная область изменяется чрезвычайно динамично. Все, кто причастен к преподаванию информатики в высшей школе, хорошо знают, как часто приходится менять содержание учебных планов, рабочих программ, учебно-методической документации. Далеко не всегда удается обеспечить соответствие материально-технической базы учебного процесса текущему состоянию предметной области. И даже своевременное реагирование на научно-технические достижения не всегда позволяет обеспечить уровень знаний и навыков выпускника, адекватный потребностям сферы материального производства и коммерческого рынка, — настолько динамичны процессы в области информационных технологий.

Ныне информатика сталкивается с парадоксальным фактом. Ее основная задача состоит в преодолении общечеловеческого кризисного явления, называемого «информационным бумом», путем внедрения средств и методов, автоматизирующих операции с данными. Однако даже в собственной предметной области информатика

испытывает такой информационный бум, какого не знает ни одна другая область человеческой деятельности. Например, мировой ассортимент изданий, имеющих прямое отношение к информатике (не считая периодических и электронных), составляет порядка десяти тысяч томов в год и полностью обновляется не реже, чем раз в два года.

Анализируя вышеуказанные особенности информатики, авторы данного пособия приходят к выводу, что для преподавания информатики в сложившихся условиях необходимо расширенное взаимодействие между учебными программами общетехнических и специальных дисциплин и учебной программой курса информатики. Основные принципы, вытекающие из такого подхода, включают непрерывность и системность образования, а также раннюю профессиональную ориентацию.

Непрерывность образования. Практические приемы работы со средствами вычислительной техники закрепляются не только при изучении информатики, но и в течение всего периода обучения. Они используются при проведении учебных занятий по самым разным дисциплинам.

Системность образования. В едином методическом подходе, основанном на системе *задача — средство — методы — приемы*, происходит перекрестное взаимодействие изучаемых дисциплин. Конкретная дисциплина поставляет комплекс *задача — методы*, а информатика обеспечивает комплекс *средства — приемы*.

Ранняя профессиональная ориентация. В системе высшего технического образования действует многоуровневая иерархическая система, основанная на том, что знания студента по общетехническим дисциплинам, как правило, реализуются в практические навыки опосредованно, то есть через дисциплины специального цикла, базирующиеся на общетехнических. Информатика — одна из немногих общетехнических дисциплин, развивающая такие практические навыки, которые востребуются напрямую и немедленно, сразу после включения молодого специалиста в профессиональную деятельность.

Свою работу над книгой авторы подчинили реализации указанных принципов. Этому подчинены структура и содержание пособия. В целом книга состоит из двадцати глав, содержащих достаточно полные сведения о современном состоянии аппаратных и программных средств вычислительной техники.

Главы 1, 2, 8, 15 являются теоретическими и обеспечивают единую методическую базу как для изучения информатики, так и для взаимодействия различных учебных дисциплин на платформе информатики.

Главы 9–14, 16, 18 представляют единую технологическую базу для взаимодействия информатики и других предметных дисциплин. Средства, рассмотренные здесь, могут быть использованы при подготовке домашних заданий, контрольных, курсовых и дипломных работ, при обработке результатов экспериментов, сборе исходной информации для самостоятельных исследований, при выполнении графических работ, математическом моделировании физических и технических процессов и при математическом обосновании разрабатываемых проектов.

Главы 3–7, 10, 12, 13, 16, 17, 19, 20 служат тем же задачам, но являются дополнительным средством ранней профессиональной ориентации. Сведения и навыки, полученные в ходе их изучения, могут быть востребованы немедленно после включения выпускника в практическую деятельность. Эти разделы позволяют обеспечить общую уверенность студента в востребованности его знаний по окончании учебного заведения, независимо от обстоятельств и особенностей конкретного трудоустройства.

Главы, имеющие теоретическое и методообразующее содержание, завершаются списком контрольных вопросов, которые могут обсуждаться на лекционных и семинарских занятиях. Главы, имеющие практическое содержание, завершаются упражнениями и исследовательскими работами. Предполагается, что практические упражнения носят инструктивно-методический характер и выполняются под руководством преподавателя (лаборанта), а исследовательские работы имеют творческий характер и комплексное содержание. Они предназначены для самостоятельной работы и предполагают подготовку итогового отчета. Различие между этими видами занятий отражено в балансе отводимого на них времени.

Исходя из структуры и содержания книги, авторы рассчитывают на то, что она будет полезна следующим категориям читателей:

- студентам технических специальностей вузов, изучающим информатику как самостоятельную дисциплину;
- преподавательскому составу, осуществляющему теоретическую и практическую подготовку студентов по дисциплине «Информатика»;
- преподавателям иных дисциплин, использующим персональные компьютеры в качестве технического средства обучения и (или) средства подготовки учебно-методических материалов (бумажных и электронных) по своей предметной области;
- лицам, самостоятельно изучающим или осваивающим аппаратные и программные средства вычислительной техники.

1.1. Информация в материальном мире

Сигналы и данные

Мы живем в материальном мире. Все, что нас окружает и с чем мы сталкиваемся ежедневно, относится либо к *физическим телам*, либо к *физическим полям*. Из курса физики мы знаем, что состояния абсолютного покоя не существует и физические объекты находятся в состоянии непрерывного движении и изменения, которое сопровождается обменом энергией и ее переходом из одной формы в другую.

Все виды *энергообмена* сопровождаются появлением сигналов, то есть все сигналы имеют в своей основе материальную энергетическую природу. При взаимодействии сигналов с физическими телами в последних возникают определенные изменения свойств — это явление называется *регистрацией сигналов*. Такие изменения можно наблюдать, измерять или фиксировать иными способами — при этом возникают и регистрируются новые сигналы, то есть образуются данные.

Данные — это зарегистрированные сигналы.

Данные и методы

Обратим внимание на то, что данные несут в себе *информацию* о событиях, произошедших в материальном мире, поскольку они являются регистрацией сигналов, возникших в результате этих событий. Однако данные не тождественны информации. Наблюдая излучения далеких звезд, человек получает определенный поток данных, но станут ли эти данные информацией, зависит еще от очень многих обстоятельств. Рассмотрим ряд примеров.

Наблюдая за состязаниями бегунов, мы с помощью механического секундомера регистрируем начальное и конечное положение стрелки прибора. В итоге мы замеряем величину ее перемещения за время забега — это регистрация данных. Однако информацию о времени преодоления дистанции мы пока не получаем. Для того чтобы данные о перемещении стрелки дали информацию о времени забега, необ-

ходимо наличие *метода* пересчета одной физической величины в другую. Надо знать цену деления шкалы секундомера (или знать метод ее определения) и надо также знать, как умножается цена деления прибора на величину перемещения, то есть надо еще обладать математическим методом умножения.

Если вместо механического секундомера используется электронный, суть дела не меняется. Вместо регистрации перемещения стрелки происходит регистрация количества тактов колебаний, произошедших в электронной системе за время измерения. Даже если секундомер непосредственно отображает время в секундах и нам не нужен метод пересчета, то метод преобразования данных все равно присутствует — он реализован специальными электронными компонентами и работает автоматически, без нашего участия.

Прослушивая передачу радиостанции на незнакомом языке, мы получаем данные, но не получаем информацию в связи с тем, что не владеем методом преобразования данных в известные нам понятия. Если эти данные записать на лист бумаги или на магнитную ленту, изменится форма их представления, произойдет новая регистрация и, соответственно, образуются новые данные. Такое преобразование можно использовать, чтобы все-таки извлечь информацию из данных путем подбора метода, адекватного их новой форме. Для обработки данных, записанных на листе бумаги, адекватным может быть метод перевода со словарем, а для обработки данных, записанных на магнитной ленте, можно пригласить переводчика, обладающего своими методами перевода, основанными на знаниях, полученных в результате обучения или предшествующего опыта.

Если в нашем примере заменить радиопередачу телевизионной трансляцией, ведущейся на незнакомом языке, то мы увидим, что наряду с данными мы все-таки получаем определенную (хотя и не полную) информацию. Это связано с тем, что люди, не имеющие дефектов зрения, априорно владеют адекватным методом восприятия данных, передаваемых электромагнитным сигналом в полосе частот видимого спектра с интенсивностью, превышающей порог чувствительности глаза. В таких случаях говорят, что *метод известен по контексту*, то есть данные, составляющие информацию, имеют свойства, однозначно определяющие адекватный метод получения этой информации. (Для сравнения скажем, что слепому «телезрителю» контекстный метод неизвестен и он оказывается в положении радиослушателя, пример с которым был рассмотрен выше.)

Понятие об информации

Несмотря на то что с понятием информации мы сталкиваемся ежедневно, строгого и общепризнанного ее определения до сих пор не существует, поэтому вместо определения обычно используют *понятие об информации*. Понятия, в отличие от определений, не даются однозначно, а вводятся на примерах, причем каждая научная дисциплина делает это по-своему, выделяя в качестве основных компонентов те, которые наилучшим образом соответствуют ее предмету и задачам. При этом типична ситуация, когда понятие об информации, введенное в рамках одной научной дисциплины, может опровергаться конкретными примерами и фактами, полученными в рамках другой науки. Например, представление об информации как о совокупности дан-

ных, повышающих уровень знаний об объективной реальности окружающего мира, характерное для естественных наук, может быть опровергнуто в рамках социальных наук. Нередки также случаи, когда исходные компоненты, составляющие понятие информации, подменяют свойствами информационных объектов, например, когда понятие информации вводят как совокупность данных, которые «могут быть усвоены и преобразованы в знания».

Для информатики как технической науки понятие информации не может основываться на таких антропоцентрических понятиях, как *знание*, и не может опираться только на объективность фактов и свидетельств. Средства вычислительной техники обладают способностью обрабатывать информацию автоматически, без участия человека, и ни о каком знании или незнании здесь речь идти не может. Эти средства могут работать с искусственной, абстрактной и даже с ложной информацией, не имеющей объективного отражения ни в природе, ни в обществе.

В этой работе мы даем новое определение информации, основанное на ранее продемонстрированном факте взаимодействия данных и методов в момент ее образования.

Информация — это продукт взаимодействия данных и адекватных им методов.

Поскольку в такой форме определение информации дается впервые, читатель приглашается для его всесторонней проверки в рамках других известных ему научных дисциплин, а мы рассмотрим пример, в свое время использованный Норбертом Винером для того, чтобы показать, как информация отдельных членов популяции становится информацией общества.

> *Допустим, я нахожусь в лесах вдвоем со смышленым дикарем, который не может говорить на моем языке и на языке которого я тоже не могу говорить. Даже без какого-либо условного языка знаков, известного нам обоим, я могу многое узнать от него. Мне нужно лишь быть особо внимательным в те моменты, когда он обнаруживает признаки волнения или интереса. Тогда я должен посмотреть вокруг, особенно в направлении его взгляда, и запомнить все, что я увижу и услышу. Не пройдет много времени, как я открою, какие предметы представляются важными для него, — не потому, что он сообщил мне о них словами, но потому, что я сам их заметил. Иначе говоря, сигнал, лишенный внутреннего содержания, может приобрести для моего спутника смысл по тому, что наблюдает он в данный момент, и может приобрести для меня смысл по тому, что наблюдаю я в данный момент. Способность дикаря замечать моменты моего особенно активного внимания сама по себе образует язык, возможности которого столь же разнообразны, как и диапазон впечатлений, доступных нам обоим.*
>
> *Н. Винер. Кибернетика*

Анализируя этот пример, мы видим, что здесь речь идет о данных и методах. Прежде всего, здесь автор прямо говорит о целой группе методов, связанных с наблюдением и анализом, и даже приводит вариант конкретного алгоритма, адекватного рамкам его гипотетического эксперимента (*посмотреть, запомнить, открыть...*). Автор неоднократно подчеркивает требование адекватности метода (дикарь должен быть *смышленым*, а наблюдатель должен быть *особо внимательным*), без которого информация может и не образоваться.

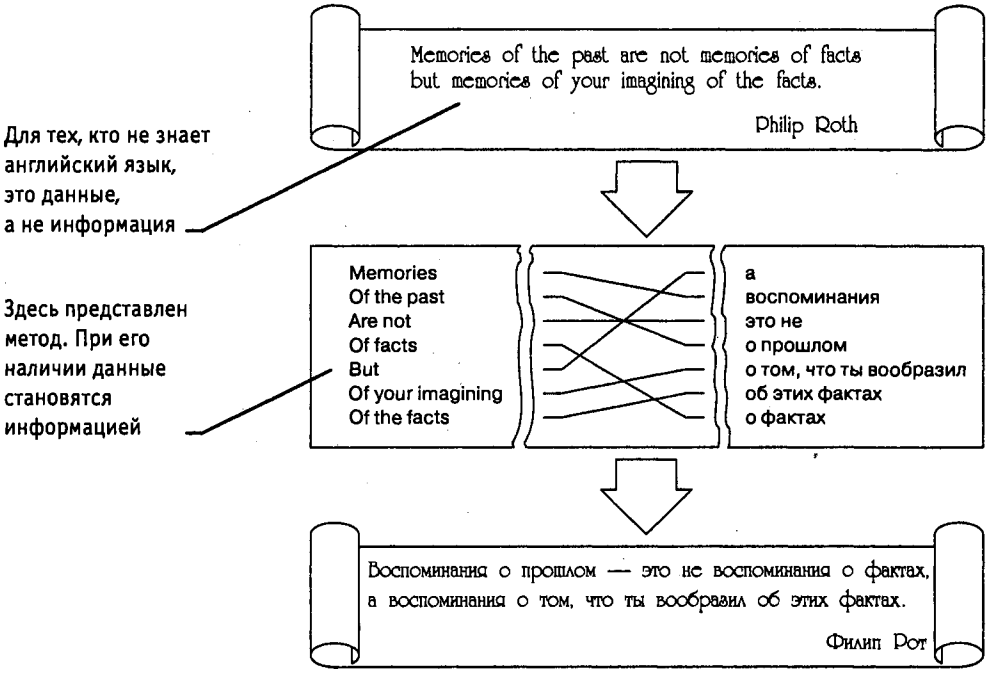


Рис. 1.1. Связь между данными и информацией

Диалектическое единство данных и методов в информационном процессе

Рассмотрим данное выше определение информации и обратим внимание на следующие обстоятельства.

1. *Динамический характер информации.* Информация не является статичным объектом — она динамически меняется и существует только в момент взаимодействия данных и методов. Все прочее время она пребывает в состоянии данных. Таким образом, информация существует только в момент протекания *информационного процесса*. Все остальное время она содержится в виде данных.
2. *Требование адекватности методов.* Одни и те же данные могут в момент потребления поставлять разную информацию в зависимости от степени адекватности взаимодействующих с ними методов. Например, для человека, не владеющего китайским языком, письмо, полученное из Пекина, дает только ту информацию, которую можно получить методом наблюдения (количество страниц, цвет и сорт бумаги, наличие незнакомых символов и т. п.). Все это информация, но это не вся информация, заключенная в письме. Использование более адекватных методов даст иную информацию.
3. *Диалектический характер взаимодействия данных и методов.* Обратим внимание на то, что данные являются *объективными*, поскольку это результат регистрации объективно существовавших сигналов, вызванных изменениями в материальных

телах или полях. В то же время, методы являются *субъективными.* В основе искусственных методов лежат алгоритмы (упорядоченные последовательности команд), составленные и подготовленные людьми (субъектами). В основе естественных методов лежат биологические свойства субъектов информационного процесса. Таким образом, *информация возникает и существует в момент диалектического взаимодействия объективных данных и субъективных методов.*

Такой дуализм известен своими проявлениями во многих науках. Так, например, в основе важнейшего вопроса философии о первичности материалистического и идеалистического подходов к теории познания лежит не что иное, как двойственный характер информационного процесса. В обоснованиях обоих подходов нетрудно обнаружить упор либо на объективность данных, либо на субъективность методов. Подход к информации как к объекту особой природы, возникающему в результате диалектического взаимодействия объективных данных с субъективными методами, позволяет во многих случаях снять противоречия, возникающие в философских обоснованиях ряда научных теорий и гипотез.

Свойства информации

Итак, информация является динамическим объектом, образующимся в момент взаимодействия объективных данных и субъективных методов. Как и всякий объект, она обладает свойствами (объекты различимы по своим свойствам). Характерной особенностью информации, отличающей ее от других объектов природы и общества, является отмеченный выше дуализм: на свойства информации влияют как свойства данных, составляющих ее содержательную часть, так и свойства методов, взаимодействующих с данными в ходе информационного процесса. По окончании процесса свойства информации переносятся на свойства новых данных, то есть свойства методов могут переходить на свойства данных.

Можно привести немало разнообразных свойств информации. Каждая научная дисциплина рассматривает те свойства, которые ей наиболее важны. С точки зрения информатики наиболее важными представляются следующие свойства: объективность, полнота, достоверность, адекватность, доступность и актуальность информации.

Объективность и субъективность информации. Понятие объективности информации является относительным. Это понятно, если учесть, что методы являются субъективными. Более объективной принято считать ту информацию, в которую методы вносят меньший субъективный элемент. Так, например, принято считать, что в результате наблюдения фотоснимка природного объекта или явления образуется более объективная информация, чем в результате наблюдения рисунка того же объекта, выполненного человеком. В ходе информационного процесса степень объективности информации всегда понижается. Это свойство учитывают, например, в правовых дисциплинах, где по-разному обрабатываются показания лиц, непосредственно наблюдавших события или получивших информацию косвенным путем (посредством умозаключений или со слов третьих лиц). В не меньшей степени объективность информации учитывают в исторических дисциплинах. Одни и те же события, зафиксированные в исторических документах разных стран и народов, выглядят совершенно по-разному. У историков имеются свои методы для тестирова-

ния объективности исторических данных и создания новых, более достоверных данных путем сопоставления, фильтрации и селекции исходных данных. Обратим внимание на то, что здесь речь идет не о повышении объективности данных, а о повышении их достоверности (это совсем другое свойство).

Полнота информации. Полнота информации во многом характеризует *качество информации* и определяет *достаточность* данных для принятия решений или для создания новых данных на основе имеющихся. Чем полнее данные, тем шире диапазон методов, которые можно использовать, тем проще подобрать метод, вносящий минимум погрешностей в ход информационного процесса.

Достоверность информации. Данные возникают в момент регистрации сигналов, но не все сигналы являются «полезными» — всегда присутствует какой-то уровень посторонних сигналов, в результате чего полезные данные сопровождаются определенным уровнем «информационного шума». Если полезный сигнал зарегистрирован более четко, чем посторонние сигналы, достоверность информации может быть более высокой. При увеличении уровня шумов достоверность информации снижается. В этом случае для передачи того же количества информации требуется использовать либо больше данных, либо более сложные методы.

Адекватность информации — это степень соответствия реальному объективному состоянию дела. Неадекватная информация может образовываться при создании новой информации на основе неполных или недостоверных данных. Однако и полные, и достоверные данные могут приводить к созданию неадекватной информации в случае применения к ним неадекватных методов.

Доступность информации — мера возможности получить ту или иную информацию. На степень доступности информации влияют одновременно как доступность данных, так и доступность адекватных методов для их интерпретации. Отсутствие доступа к данным или отсутствие адекватных методов обработки данных приводят к одинаковому результату: информация оказывается недоступной. Отсутствие адекватных методов для работы с данными во многих случаях приводит к применению неадекватных методов, в результате чего образуется неполная, неадекватная или недостоверная информация.

Актуальность информации — это степень соответствия информации текущему моменту времени. Нередко с актуальностью, как и с полнотой, связывают коммерческую ценность информации. Поскольку информационные процессы растянуты во времени, то достоверная и адекватная, но устаревшая информация может приводить к ошибочным решениям. Необходимость поиска (или разработки) адекватного метода для работы с данными может приводить к такой задержке в получении информации, что она становится неактуальной и ненужной. На этом, в частности, основаны многие современные системы шифрования данных с *открытым ключом*. Лица, не владеющие ключом (методом) для чтения данных, могут заняться поиском ключа, поскольку алгоритм его работы доступен, но продолжительность этого поиска столь велика, что за время работы информация теряет актуальность и, соответственно, связанную с ней практическую ценность.

1.2. Данные

Носители данных

Данные — диалектическая составная часть информации. Они представляют собой зарегистрированные сигналы. При этом физический метод регистрации может быть любым: механическое перемещение физических тел, изменение их формы или параметров качества поверхности, изменение электрических, магнитных, оптических характеристик, химического состава и (или) характера химических связей, изменение состояния электронной системы и многое другое. В соответствии с методом регистрации данные могут храниться и транспортироваться на носителях различных видов.

Самым распространенным носителем данных, хотя и не самым экономичным, по-видимому, является бумага. На бумаге данные регистрируются путем изменения оптических характеристик ее поверхности. Изменение оптических свойств (изменение коэффициента отражения поверхности в определенном диапазоне длин волн) используется также в устройствах, осуществляющих запись лазерным лучом на пластмассовых носителях с отражающим покрытием (*CD-ROM*). В качестве носителей, использующих изменение магнитных свойств, можно назвать магнитные ленты и диски. Регистрация данных путем изменения химического состава поверхностных веществ носителя широко используется в фотографии. На биохимическом уровне происходит накопление и передача данных в живой природе.

Носители данных интересуют нас не сами по себе, а постольку, поскольку свойства информации весьма тесно связаны со свойствами ее носителей. Любой носитель можно характеризовать параметром *разрешающей способности* (количеством данных, записанных в принятой для носителя единице измерения) и *динамическим диапазоном* (логарифмическим отношением интенсивности амплитуд максимального и минимального регистрируемого сигналов). От этих свойств носителя нередко зависят такие свойства информации, как полнота, доступность и достоверность. Например, мы можем рассчитывать на то, что в базе данных, размещаемой на компакт-диске, проще обеспечить полноту информации, чем в аналогичной по назначению базе данных, размещенной на гибком магнитном диске, поскольку в первом случае плотность записи данных на единице длины дорожки намного выше. Для обычного потребителя доступность информации в книге заметно выше, чем той же информации на компакт-диске, поскольку не все потребители обладают необходимым оборудованием. И наконец, известно, что визуальный эффект от просмотра слайда в проекторе намного больше, чем от просмотра аналогичной иллюстрации, напечатанной на бумаге, поскольку диапазон яркостных сигналов в проходящем свете на два-три порядка больше, чем в отраженном.

Задача преобразования данных с целью смены носителя относится к одной из важнейших задач информатики. В структуре стоимости вычислительных систем устройства для ввода и вывода данных, работающие с носителями информации, составляют до половины стоимости аппаратных средств.

Операции с данными

В ходе информационного процесса данные преобразуются из одного вида в другой с помощью методов. Обработка данных включает в себя множество различных

операций. По мере развития научно-технического прогресса и общего усложнения связей в человеческом обществе трудозатраты на обработку данных неуклонно возрастают. Прежде всего это связано с постоянным усложнением условий управления производством и обществом. Второй фактор, также вызывающий общее увеличение объемов обрабатываемых данных, тоже связан с научно-техническим прогрессом, а именно с быстрыми темпами появления и внедрения новых носителей данных, средств их хранения и доставки.

В структуре возможных операций с данными можно выделить следующие основные:

- *сбор данных* — накопление информации с целью обеспечения достаточной полноты для принятия решений;
- *формализация данных* — приведение данных, поступающих из разных источников, к одинаковой форме, чтобы сделать их сопоставимыми между собой, то есть повысить их уровень доступности;
- *фильтрация данных* — отсеивание «лишних» данных, в которых нет необходимости для принятия решений; при этом должен уменьшаться уровень «шума», а достоверность и адекватность данных должны возрастать;
- *сортировка данных* — упорядочение данных по заданному признаку с целью удобства использования; повышает доступность информации;
- *архивация данных* — организация хранения данных в удобной и легкодоступной форме; служит для снижения экономических затрат по хранению данных и повышает общую надежность информационного процесса в целом;
- *защита данных* — комплекс мер, направленных на предотвращение утраты, воспроизведения и модификации данных;
- *транспортировка данных* — прием и передача (доставка и поставка) данных между удаленными участниками информационного процесса; при этом источник данных в информатике принято называть *сервером*, а потребителя — *клиентом*;
- *преобразование данных* — перевод данных из одной формы в другую или из одной структуры в другую. Преобразование данных часто связано с изменением типа носителя: например книги можно хранить в обычной бумажной форме, но можно использовать для этого и электронную форму, и микрофотопленку. Необходимость в многократном преобразовании данных возникает также при их транспортировке, особенно если она осуществляется средствами, не предназначенными для транспортировки данного вида данных. В качестве примера можно упомянуть, что для транспортировки цифровых потоков данных по каналам телефонных сетей (которые изначально были ориентированы только на передачу аналоговых сигналов в узком диапазоне частот) необходимо преобразование цифровых данных в некое подобие звуковых сигналов, чем и занимаются специальные устройства — *телефонные модемы*.

Приведенный здесь список типовых операций с данными далеко не полон. Миллионы людей во всем мире занимаются созданием, обработкой, преобразованием и транспортировкой данных, и на каждом рабочем месте выполняются свои специфи-

ческие операции, необходимые для управления социальными, экономическими, промышленными, научными и культурными процессами. Полный список возможных операций составить невозможно, да и не нужно. Сейчас нам важен другой вывод: *работа с информацией может иметь огромную трудоемкость и ее надо автоматизировать.*

Кодирование данных двоичным кодом

Для автоматизации работы с данными, относящимися к различным типам, очень важно унифицировать их форму представления — для этого обычно используется прием *кодирования*, то есть выражение данных одного типа через данные другого типа. Естественные человеческие *языки* — это не что иное, как системы кодирования понятий для выражения мыслей посредством речи. К языкам близко примыкают *азбуки* (системы кодирования компонентов языка с помощью графических символов). История знает интересные, хотя и безуспешные попытки создания «универсальных» языков и азбук. По-видимому, безуспешность попыток их внедрения связана с тем, что национальные и социальные образования естественным образом понимают, что изменение системы кодирования общественных данных непременно приводит к изменению общественных методов (то есть норм права и морали), а это может быть связано с социальными потрясениями.

Та же проблема универсального средства кодирования достаточно успешно реализуется в отдельных отраслях техники, науки и культуры. В качестве примеров можно привести систему записи математических выражений, телеграфную азбуку, морскую флажковую азбуку, систему Брайля для слепых и многое другое.

Своя система существует и в вычислительной технике — 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*).

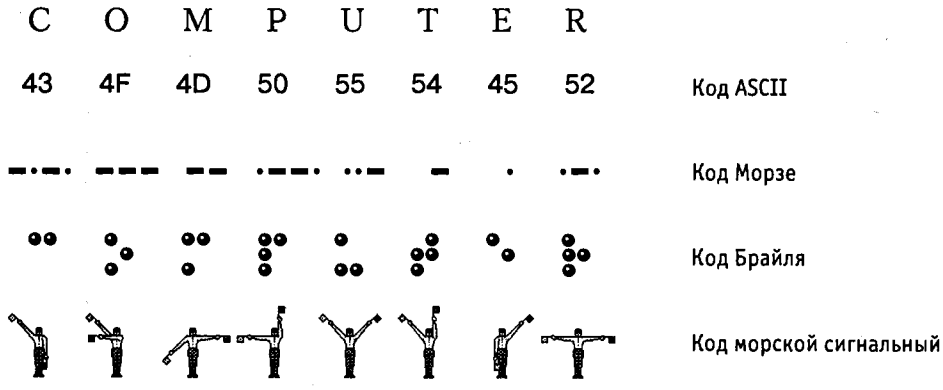


Рис. 1.2. Примеры различных систем кодирования

Одним битом могут быть выражены два понятия: 0 или 1 (*да* или *нет*, *черное* или *белое*, *истина* или *ложь* и т. п.). Если количество битов увеличить до двух, то уже можно выразить четыре различных понятия:

00 01 10 11

Тремя битами можно закодировать восемь различных значений:

000 001 010 011 100 101 110 111

Увеличивая на единицу количество разрядов в системе двоичного кодирования, мы увеличиваем в два раза количество значений, которое может быть выражено в данной системе, то есть общая формула имеет вид:

$$N = 2^m, \quad \text{где:}$$

N — количество независимых кодируемых значений;

m — разрядность двоичного кодирования, принятая в данной системе.

Кодирование целых и действительных чисел

Целые числа кодируются двоичным кодом достаточно просто — достаточно взять целое число и делить его пополам до тех пор, пока в остатке не образуется ноль или единица. Совокупность остатков от каждого деления, записанная справа налево вместе с последним остатком, и образует двоичный аналог десятичного числа.

$$19 : 2 = 9 + 1$$
$$9 : 2 = 4 + 1$$
$$4 : 2 = 2 + 0$$
$$2 : 2 = 1$$

Таким образом, $19_{10} = 1011_2$.

Для кодирования целых чисел от 0 до 255 достаточно иметь 8 разрядов двоичного кода (8 бит). Шестнадцать бит позволяют закодировать целые числа от 0 до 65535, а 24 бита — уже более 16,5 миллионов разных значений.

Для кодирования действительных чисел используют 80-разрядное кодирование. При этом число предварительно преобразуется в *нормализованную форму*:

$$3{,}1415926 = 0{,}31415926 \cdot 10^1$$
$$300\,000 = 0{,}3 \cdot 10^6$$
$$123\,456\,789 = 0{,}123456789 \cdot 10^{10}$$

Первая часть числа называется *мантиссой*, а вторая — *характеристикой*. Большую часть из 80 бит отводят для хранения мантиссы (вместе со знаком) и некоторое фиксированное количество разрядов отводят для хранения характеристики (тоже со знаком).

Кодирование текстовых данных

Если каждому символу алфавита сопоставить определенное целое число (например, порядковый номер), то с помощью двоичного кода можно кодировать и текстовую

информацию. Восьми двоичных разрядов достаточно для кодирования 256 различных символов. Этого хватит, чтобы выразить различными комбинациями восьми битов все символы английского и русского языков, как строчные, так и прописные, а также знаки препинания, символы основных арифметических действий и некоторые общепринятые специальные символы, например символ «§».

Технически это выглядит очень просто, однако всегда существовали достаточно веские организационные сложности. В первые годы развития вычислительной техники они были связаны с отсутствием необходимых стандартов, а в настоящее время вызваны, наоборот, изобилием одновременно действующих и противоречивых стандартов. Для того чтобы весь мир одинаково кодировал текстовые данные, нужны единые таблицы кодирования, а это пока невозможно из-за противоречий между символами национальных алфавитов, а также противоречий корпоративного характера.

Для английского языка, захватившего де-факто нишу международного средства общения, противоречия уже сняты. Институт стандартизации США (*ANSI — American National Standard Institute*) ввел в действие систему кодирования *ASCII* (*American Standard Code for Information Interchange — стандартный код информационного обмена США*). В системе *ASCII* закреплены две таблицы кодирования — *базовая* и *расширенная*. Базовая таблица закрепляет значения кодов от 0 до 127, а расширенная относится к символам с номерами от 128 до 255.

Первые 32 кода базовой таблицы, начиная с нулевого, отданы производителям аппаратных средств (в первую очередь производителям компьютеров и печатающих устройств). В этой области размещаются так называемые *управляющие коды*, которым не соответствуют никакие символы языков, и, соответственно, эти коды не выводятся ни на экран, ни на устройства печати, но ими можно управлять тем, как производится вывод прочих данных.

Начиная с кода 32 по код 127 размещены коды символов английского алфавита, знаков препинания, цифр, арифметических действий и некоторых вспомогательных символов. Базовая таблица кодировки *ASCII* приведена в таблице 1.1.

Таблица 1.1. Базовая таблица кодировки ASCII

32	пробел	48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	

Аналогичные системы кодирования текстовых данных были разработаны и в других странах. Так, например, в СССР в этой области действовала система кодирования КОИ-7 (*код обмена информацией, семизначный*). Однако поддержка производителей оборудования и программ вывела американский код *ASCII* на уровень международного стандарта, и национальным системам кодирования пришлось «отступить» во вторую, расширенную часть системы кодирования, определяющую значения кодов со 128 по 255. Отсутствие единого стандарта в этой области привело к множественности одновременно действующих кодировок. Только в России можно указать три действующих стандарта кодировки и еще два устаревших.

Так, например, кодировка символов русского языка, известная как кодировка *Windows-1251*, была введена «извне» — компанией *Microsoft*, но, учитывая широкое распространение операционных систем и других продуктов этой компании в России, она глубоко закрепилась и нашла широкое распространение (таблица 1.2). Эта кодировка используется на большинстве локальных компьютеров, работающих на платформе *Windows*.

Таблица 1.2. Кодировка Windows 1251

128 Ђ	144 ђ	160	176 °	192 А	208 Р	224 а	240 р
129 Ѓ	145 ‘	161 Ў	177 ±	193 Б	209 С	225 б	241 с
130 ‚	146 ’	162 ў	178 І	194 В	210 Т	226 в	242 т
131 ѓ	147 “	163 Ј	179 і	195 Г	211 У	227 г	243 у
132 „	148 ”	164 ¤	180 ґ	196 Д	212 Ф	228 д	244 ф
133 …	149 •	165 Ґ	181 µ	197 Е	213 Х	229 е	245 х
134 †	150 –	166 ¦	182 ¶	198 Ж	214 Ц	230 ж	246 ц
135 ‡	151 —	167 §	183 ·	199 З	215 Ч	231 з	247 ч
136	152	168 Ё	184 ё	200 И	216 Ш	232 и	248 ш
137 ‰	153 ™	169 ©	185 №	201 Й	217 Щ	233 й	249 щ
138 Љ	154 љ	170 Є	186 є	202 К	218 Ъ	234 к	250 ъ
139 ‹	155 ›	171 «	187 »	203 Л	219 Ы	235 л	251 ы
140 Њ	156 њ	172 ¬	188 ј	204 М	220 Ь	236 м	252 ь
141 Ќ	157 ќ	173 -	189 Ѕ	205 Н	221 Э	237 н	253 э
142 Ћ	158 ћ	174 ®	190 ѕ	206 О	222 Ю	238 о	254 ю
143 Џ	159 џ	175 Ї	191 ї	207 П	223 Я	239 п	255 я

Другая распространенная кодировка носит название КОИ-8 (*код обмена информацией, восьмизначный*) — ее происхождение относится ко временам действия Совета Экономической Взаимопомощи государств Восточной Европы (таблица 1.3). Сегодня кодировка КОИ-8 имеет широкое распространение в компьютерных сетях на территории России и в российском секторе Интернета.

Международный стандарт, в котором предусмотрена кодировка символов русского алфавита, носит название кодировки *ISO* (*International Standard Organization — Международный институт стандартизации*). На практике данная кодировка используется редко (таблица 1.4).

На компьютерах, работающих в операционных системах *MS-DOS*, могут действовать еще две кодировки (кодировка *ГОСТ* и кодировка *ГОСТ-альтернативная*). Первая из них считалась устаревшей даже в первые годы появления персональной вычислительной техники, но вторая используется и по сей день (см. таблицу 1.5).

Таблица 1.3. Кодировка КОИ-8

128	144 ▒	160 ─	176 ├	192 ю	208 п	224 Ю	240 П
129 │	145 ▒	161 Ё	177 ├	193 а	209 я	225 А	241 Я
130 ┌	146 ▒	162 ┌	178 ┤	194 б	210 р	226 Б	242 Р
131 ┐	147 ┌	163 ё	179 Ё	195 ц	211 с	227 Ц	243 С
132 └	148 ■	164 ┌	180 ┤	196 д	212 т	228 Д	244 Т
133 ┘	149 •	165 ┌	181 ┤	197 е	213 у	229 Е	245 У
134 ├	150 √	166 ┐	182 ┬	198 ф	214 ж	230 Ф	246 Ж
135 ┤	151 ≈	167 ┐	183 ┬	199 г	215 в	231 Г	247 В
136 ┬	152 ≤	168 ┐	184 ┬	200 х	216 ь	232 Х	248 Ь
137 ┴	153 ≥	169 └	185 ┴	201 и	217 ы	233 И	249 Ы
138 ┼	154	170 └	186 ┴	202 й	218 з	234 Й	250 З
139 ■	155 ⌠	171 └	187 ┴	203 к	219 ш	235 К	251 Ш
140 ▄	156 °	172 ┘	188 ┼	204 л	220 э	236 Л	252 Э
141 ▌	157 ²	173 ┘	189 ┼	205 м	221 щ	237 М	253 Щ
142 ▐	158 ·	174 ┘	190 ┼	206 н	222 ч	238 Н	254 Ч
143 ▀	159 ÷	175 ├	191 ё	207 о	223 ъ	239 О	255 Ъ

Таблица 1.4. Кодировка ISO

	160	176 А	192 Р	208 а	224 р	240 №
	161 Ё	177 Б	193 С	209 б	225 с	241 ё
	162 Ђ	178 В	194 Т	210 в	226 т	242 ђ
	163 Ѓ	179 Г	195 У	211 г	227 у	243 ѓ
	164 Є	180 Д	196 Ф	212 д	228 ф	244 є
	165 S	181 Е	197 Х	213 е	229 х	245 s
	166 I	182 Ж	198 Ц	214 ж	230 ц	246 і
В *ISO* не определены	167 Ї	183 З	199 Ч	215 з	231 ч	247 ї
	168 J	184 И	200 Ш	216 и	232 ш	248 j
	169 Љ	185 Й	201 Щ	217 й	233 щ	249 љ
	170 Њ	186 К	202 Ъ	218 к	234 ъ	250 њ
	171 Ћ	187 Л	203 Ы	219 л	235 ы	251 ћ
	172 Ќ	188 М	204 Ь	220 м	236 ь	252 ќ
	173 -	189 Н	205 Э	221 н	237 э	253 §
	174 Ў	190 О	206 Ю	222 о	238 ю	254 ў
	175 Џ	191 П	207 Я	223 п	239 я	255 џ

Таблица 1.5. ГОСТ-альтернативная кодировка

128 А	144 Р	160 а	176 ▒	192 └	208 ┴	224 р	240 Ё
129 Б	145 С	161 б	177 ▒	193 Г	209 ┬	225 с	241 ё
130 В	146 Т	162 в	178 ▒	194 ┬	210 ┬	226 т	242 Є
131 Г	147 У	163 г	179 │	195 ├	211 └	227 у	243 є
132 Д	148 Ф	164 д	180 ┤	196 ─	212 └	228 ф	244 Ї
133 Е	149 Х	165 е	181 ┤	197 ┼	213 ┌	229 х	245 ї
134 Ж	150 Ц	166 ж	182 ┤	198 ├	214 ┌	230 ц	246 Ў
135 З	151 Ч	167 з	183 ┐	199 ├	215 ┼	231 ч	247 ў
136 И	152 Ш	168 и	184 ┐	200 └	216 ┼	232 ш	248 °
137 Й	153 Щ	169 й	185 ┤	201 ┌	217 ┘	233 щ	249 •
138 К	154 Ъ	170 к	186 │	202 ┴	218 ┌	234 ъ	250 ·
139 Л	155 Ы	171 л	187 ┐	203 ┬	219 ▌	235 ы	251 √
140 М	156 Ь	172 м	188 ┘	204 ├	220 ▄	236 ь	252 №
141 Н	157 Э	173 н	189 ┘	205 ─	221 ▐	237 э	253 ¤
142 О	158 Ю	174 о	190 ┘	206 ┼	222 ▐	238 ю	254 ■
143 П	159 Я	175 п	191 ┐	207 ┴	223 ▀	239 я	255

В связи с изобилием систем кодирования текстовых данных, действующих в России, возникает задача межсистемного преобразования данных — это одна из распространенных задач информатики.

Универсальная система кодирования текстовых данных

Если проанализировать организационные трудности, связанные с созданием единой системы кодирования текстовых данных, то можно прийти к выводу, что они вызваны ограниченным набором кодов (256). В то же время очевидно, что если, например, кодировать символы не восьмиразрядными двоичными числами, а числами с большим количеством разрядов, то и диапазон возможных значений кодов станет намного больше. Такая система, основанная на 16-разрядном кодировании символов, получила название *универсальной — UNICODE*. Шестнадцать разрядов позволяют обеспечить уникальные коды для 65 536 различных символов — этого поля достаточно для размещения в одной таблице символов большинства языков планеты.

Несмотря на тривиальную очевидность такого подхода, простой механический переход на данную систему долгое время сдерживался из-за недостаточных ресурсов средств вычислительной техники (в системе кодирования *UNICODE* все текстовые документы автоматически становятся вдвое длиннее). Во второй половине 90-х годов технические средства достигли необходимого уровня обеспеченности ресурсами, и сегодня мы наблюдаем постепенный перевод документов и программных средств на универсальную систему кодирования. Для индивидуальных пользователей это еще больше добавило забот по согласованию документов, выполненных в разных системах кодирования, с программными средствами, но это надо понимать как трудности переходного периода.

Кодирование графических данных

Если рассмотреть с помощью увеличительного стекла черно-белое графическое изображение, напечатанное в газете или книге, то можно увидеть, что оно состоит из мельчайших точек, образующих характерный узор, называемый *растром* (рис. 1.3).

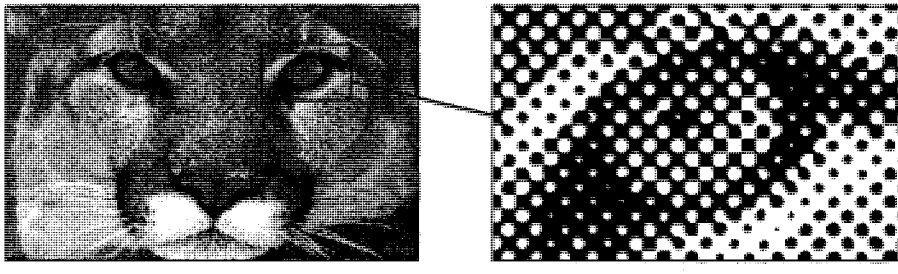


Рис. 1.3. Растр — это метод кодирования графической информации, издавна принятый в полиграфии

Поскольку линейные координаты и индивидуальные свойства каждой точки (яркость) можно выразить с помощью целых чисел, то можно сказать, что растровое кодирование позволяет использовать двоичный код для представления графических данных. Общепринятым на сегодняшний день считается представление черно-белых

иллюстраций в виде комбинации точек с 256 градациями серого цвета, и, таким образом, для кодирования яркости любой точки обычно достаточно восьмиразрядного двоичного числа.

Для кодирования цветных графических изображений применяется *принцип декомпозиции* произвольного цвета на основные составляющие. В качестве таких составляющих используют три основные цвета: красный (*Red, R*), зеленый (*Green, G*) и синий (*Blue, B*). На практике считается (хотя теоретически это не совсем так), что любой цвет, видимый человеческим глазом, можно получить путем механического смешения этих трех основных цветов. Такая система кодирования называется системой *RGB* по первым буквам названий основных цветов.

Если для кодирования яркости каждой из основных составляющих использовать по 256 значений (восемь двоичных разрядов), как это принято для полутоновых черно-белых изображений, то на кодирование цвета одной точки надо затратить 24 разряда. При этом система кодирования обеспечивает однозначное определение 16,5 млн различных цветов, что на самом деле близко к чувствительности человеческого глаза. Режим представления цветной графики с использованием 24 двоичных разрядов называется *полноцветным* (*True Color*).

Каждому из основных цветов можно поставить в соответствие дополнительный цвет, то есть цвет, дополняющий основной цвет до белого. Нетрудно заметить, что для любого из основных цветов дополнительным будет цвет, образованный суммой пары остальных основных цветов. Соответственно, дополнительными цветами являются: голубой (*Cyan, C*), пурпурный (*Magenta, M*) и желтый (*Yellow, Y*). Принцип декомпозиции произвольного цвета на составляющие компоненты можно применять не только для основных цветов, но и для дополнительных, то есть любой цвет можно представить в виде суммы голубой, пурпурной и желтой составляющей. Такой метод кодирования цвета принят в полиграфии, но в полиграфии используется еще и четвертая краска — черная (*Black, K*). Поэтому данная система кодирования обозначается четырьмя буквами *CMYK* (черный цвет обозначается буквой *K*, потому, что буква *B* уже занята синим цветом), и для представления цветной графики в этой системе надо иметь 32 двоичных разряда. Такой режим тоже называется *полноцветным* (*True Color*).

Если уменьшить количество двоичных разрядов, используемых для кодирования цвета каждой точки, то можно сократить объем данных, но при этом диапазон кодируемых цветов заметно сокращается. Кодирование цветной графики 16-разрядными двоичными числами называется режимом *High Color*.

При кодировании информации о цвете с помощью восьми бит данных можно передать только 256 цветовых оттенков. Такой метод кодирования цвета называется *индексным*. Смысл названия в том, что, поскольку 256 значений совершенно недостаточно, чтобы передать весь диапазон цветов, доступный человеческому глазу, код каждой точки растра выражает не цвет сам по себе, а только его номер (*индекс*) в некоей справочной таблице, называемой *палитрой*. Разумеется, эта палитра должна прикладываться к графическим данным — без нее нельзя воспользоваться методами воспроизведения информации на экране или бумаге (то есть, воспользоваться, конечно,

можно, но из-за неполноты данных полученная информация не будет адекватной: листва на деревьях может оказаться красной, а небо — зеленым).

Кодирование звуковой информации

Приемы и методы работы со звуковой информацией пришли в вычислительную технику наиболее поздно. К тому же, в отличие от числовых, текстовых и графических данных, у звукозаписей не было столь же длительной и проверенной истории кодирования. В итоге методы кодирования звуковой информации двоичным кодом далеки от стандартизации. Множество отдельных компаний разработали свои корпоративные стандарты, но если говорить обобщенно, то можно выделить два основных направления.

Метод FM (*Frequency Modulation*) основан на том, что теоретически любой сложный звук можно разложить на последовательность простейших гармонических сигналов разных частот, каждый из которых представляет собой правильную синусоиду, а следовательно, может быть описан числовыми параметрами, то есть кодом. В природе звуковые сигналы имеют непрерывный спектр, то есть являются аналоговыми. Их разложение в гармонические ряды и представление в виде дискретных цифровых сигналов выполняют специальные устройства — *аналогово-цифровые преобразователи* (*АЦП*). Обратное преобразование для воспроизведения звука, закодированного числовым кодом, выполняют *цифро-аналоговые преобразователи* (*ЦАП*). При таких преобразованиях неизбежны потери информации, связанные с методом кодирования, поэтому качество звукозаписи обычно получается не вполне удовлетворительным и соответствует качеству звучания простейших электромузыкальных инструментов с окрасом, характерным для электронной музыки. В то же время, данный метод кодирования обеспечивает весьма компактный код, и потому он нашел применение еще в те годы, когда ресурсы средств вычислительной техники были явно недостаточны.

Метод таблично-волнового (*Wave-Table*) синтеза лучше соответствует современному уровню развития техники. Если говорить упрощенно, то можно сказать, что где-то в заранее подготовленных таблицах хранятся образцы звуков для множества различных музыкальных инструментов (хотя не только для них). В технике такие образцы называют *сэмплами*. Числовые коды выражают тип инструмента, номер его модели, высоту тона, продолжительность и интенсивность звука, динамику его изменения, некоторые параметры среды, в которой происходит звучание, а также прочие параметры, характеризующие особенности звука. Поскольку в качестве образцов используются «реальные» звуки, то качество звука, полученного в результате синтеза, получается очень высоким и приближается к качеству звучания реальных музыкальных инструментов.

Основные структуры данных

Работа с большими наборами данных автоматизируется проще, когда данные *упорядочены*, то есть образуют заданную структуру. Существует три основных типа структур данных: *линейная*, *иерархическая* и *табличная*. Их можно рассмотреть на примере обычной книги.

Если разобрать книгу на отдельные листы и перемешать их, книга потеряет свое назначение. Она по-прежнему будет представлять набор данных, но подобрать адекватный метод для получения из нее информации весьма непросто. (Еще хуже дело будет обстоять, если из книги вырезать каждую букву отдельно, — в этом случае вряд ли вообще найдется адекватный метод для ее прочтения.)

Если же собрать все листы книги в правильной последовательности, мы получим простейшую структуру данных — *линейную*. Такую книгу уже можно читать, хотя для поиска нужных данных ее придется прочитать подряд, начиная с самого начала, что не всегда удобно.

Для быстрого поиска данных существует *иерархическая структура*. Так, например, книги разбивают на части, разделы, главы, параграфы и т. п. Элементы структуры более низкого уровня входят в элементы структуры более высокого уровня: разделы состоят из глав, главы из параграфов и т. д.

Для больших массивов поиск данных в иерархической структуре намного проще, чем в линейной, однако и здесь необходима *навигация*, связанная с необходимостью просмотра. На практике задачу упрощают тем, что в большинстве книг есть вспомогательная перекрестная *таблица*, связывающая элементы иерархической структуры с элементами линейной структуры, то есть связывающая разделы, главы и параграфы с номерами страниц. В книгах с простой иерархической структурой, рассчитанных на последовательное чтение, эту таблицу принято называть *оглавлением*, а в книгах со сложной структурой, допускающей выборочное чтение, ее называют *содержанием*.

Линейные структуры (списки данных, векторы данных)

Линейные структуры — это хорошо знакомые нам списки. *Список* — это простейшая структура данных, отличающаяся тем, что каждый элемент данных однозначно определяется своим номером в массиве. Проставляя номера на отдельных страницах рассыпанной книги, мы создаем структуру списка. Обычный журнал посещаемости занятий, например, имеет структуру списка, поскольку все студенты группы зарегистрированы в нем под своими *уникальными* номерами. Мы называем номера *уникальными* потому, что в одной группе не могут быть зарегистрированы два студента с одним и тем же номером.

При создании любой структуры данных надо решить два вопроса: как разделять элементы данных между собой и как разыскивать нужные элементы. В журнале посещаемости, например, это решается так: каждый новый элемент списка заносится с новой строки, то есть разделителем является конец строки. Тогда нужный элемент можно разыскать по номеру строки.

N п/п	Фамилия, Имя, Отчество
1	Аистов Александр Алексеевич
2	Бобров Борис Борисович
3	Воробьева Валентина Владиславовна
...	.....
27	Сорокин Сергей Семенович

Разделителем может быть и какой-нибудь специальный символ. Нам хорошо известны разделители между словами — это пробелы. В русском и во многих европейских языках общепринятым разделителем предложений является точка. В рассмотренном нами классном журнале в качестве разделителя можно использовать любой символ, который не встречается в самих данных, например символ «*». Тогда список выглядел бы так:

Аистов Александр Алексеевич * Бобров Борис Борисович * Воробьева Валентина Владиславовна * ... * Сорокин Сергей Семенович

В этом случае для розыска элемента с номером n надо просмотреть список начиная с самого начала и пересчитать встретившиеся разделители. Когда будет отсчитано $n-1$ разделителей, начнется нужный элемент. Он закончится, когда будет встречен следующий разделитель.

Еще проще можно действовать, если все элементы списка имеют равную длину. В этом случае разделители в списке вообще не нужны. Для розыска элемента с номером n надо просмотреть список с самого начала и отсчитать $a(n-1)$ символ, где a — длина одного элемента. Со следующего символа начнется нужный элемент. Его длина тоже равна a, поэтому его конец определить нетрудно. Такие упрощенные списки, состоящие из элементов равной длины, называют *векторами данных*. Работать с ними особенно удобно.

Таким образом, *линейные структуры данных (списки) — это упорядоченные структуры, в которых адрес элемента однозначно определяется его номером.*

Табличные структуры (таблицы данных, матрицы данных)

С таблицами данных мы тоже хорошо знакомы, достаточно вспомнить всем известную таблицу умножения. Табличные структуры отличаются от списочных тем, что элементы данных определяются *адресом ячейки*, который состоит не из одного параметра, как в списках, а из нескольких. Для таблицы умножения, например, адрес ячейки определяется номерами строки и столбца. Нужная ячейка находится на их пересечении, а элемент выбирается из ячейки.

При хранении табличных данных количество разделителей должно быть больше, чем для данных, имеющих структуру списка. Например, когда таблицы печатают в книгах, строки и столбцы разделяют графическими элементами — линиями вертикальной и горизонтальной разметки (рис. 1.4).

Если нужно сохранить таблицу в виде длинной символьной строки, используют один символ-разделитель между элементами, принадлежащими одной строке, и другой разделитель для отделения строк, например так:

Меркурий*0,39*0,056*0#Венера*0,67*0,88*0#Земля*1,0*1,0*1#Марс*1,51*0,1*2#...

Для розыска элемента, имеющего адрес ячейки (m, n), надо просмотреть набор данных с самого начала и пересчитать внешние разделители. Когда будет отсчитан $m-1$ разделитель, надо пересчитывать внутренние разделители. После того как будет найден $n-1$ разделитель, начнется нужный элемент. Он закончится, когда будет встречен любой очередной разделитель.

Планета	Расстояние до Солнца, а.е.	Относительная масса	Количество спутников
Меркурий	0,39	0,056	0
Венера	0,67	0,88	0
Земля	1,0	1,0	1
Марс	1,51	0,1	2
Юпитер	5,2	318	16

Рис. 1.4. В двумерных таблицах, которые печатают в книгах, применяется два типа разделителей — вертикальные и горизонтальные

Еще проще можно действовать, если все элементы таблицы имеют равную длину. Такие таблицы называют *матрицами*. В данном случае разделители не нужны, поскольку все элементы имеют равную длину и количество их известно. Для розыска элемента с адресом (m, n) в матрице, имеющей M строк и N столбцов, надо просмотреть ее с самого начала и отсчитать $a\,[N(m-1)+(n-1)]$ символ, где a — длина одного элемента. Со следующего символа начнется нужный элемент. Его длина тоже равна a, поэтому его конец определить нетрудно.

Таким образом, табличные структуры данных (*матрицы*) — это упорядоченные структуры, в которых *адрес элемента определяется номером строки и номером столбца, на пересечении которых находится ячейка, содержащая искомый элемент.*

Многомерные таблицы. Выше мы рассмотрели пример таблицы, имеющей два измерения (строка и столбец), но в жизни нередко приходится иметь дело с таблицами, у которых количество измерений больше. Вот пример таблицы, с помощью которой может быть организован учет учащихся.

Номер факультета:	3
Номер курса (на факультете):	2
Номер специальности (на курсе):	2
Номер группы в потоке одной специальности:	1
Номер учащегося в группе:	19

Размерность такой таблицы равна пяти, и для однозначного отыскания данных об учащемся в подобной структуре надо знать все пять параметров (координат).

Иерархические структуры данных

Нерегулярные данные, которые трудно представить в виде списка или таблицы, часто представляют в виде *иерархических структур*. С подобными структурами мы очень хорошо знакомы по обыденной жизни. Иерархическую структуру имеет система почтовых адресов. Подобные структуры также широко применяют в научных систематизациях и всевозможных классификациях (рис. 1.5).

В иерархической структуре адрес каждого элемента определяется путем доступа (маршрутом), ведущим от вершины структуры к данному элементу. Вот, например, как выглядит путь доступа к команде, запускающей программу Калькулятор (стандартная программа компьютеров, работающих в операционной системе *Windows 98*):

Пуск ▸ Программы ▸ Стандартные ▸ Калькулятор.

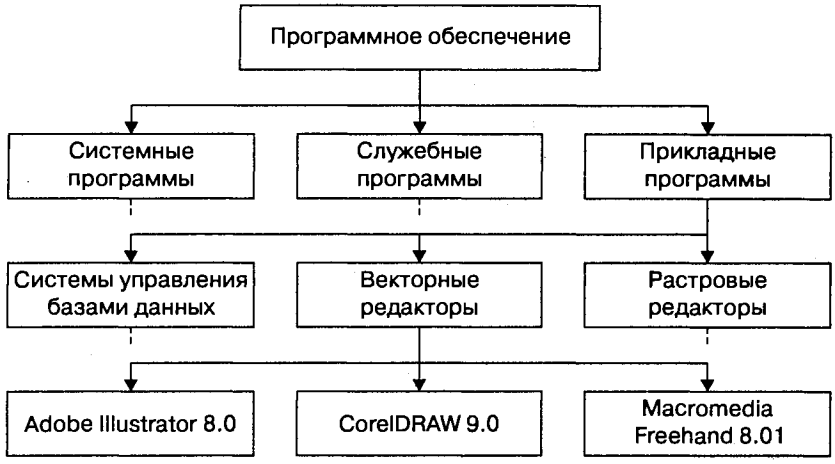


Рис. 1.5. Пример иерархической структуры данных

Дихотомия данных. Основным недостатком иерархических структур данных является увеличенный размер пути доступа. Очень часто бывает так, что длина маршрута оказывается больше, чем длина самих данных, к которым он ведет. Поэтому в информатике применяют методы для регуляризации иерархических структур с тем, чтобы сделать путь доступа компактным. Один из методов получил название *дихотомии*. Его суть понятна из примера, представленного на рис. 1.6.

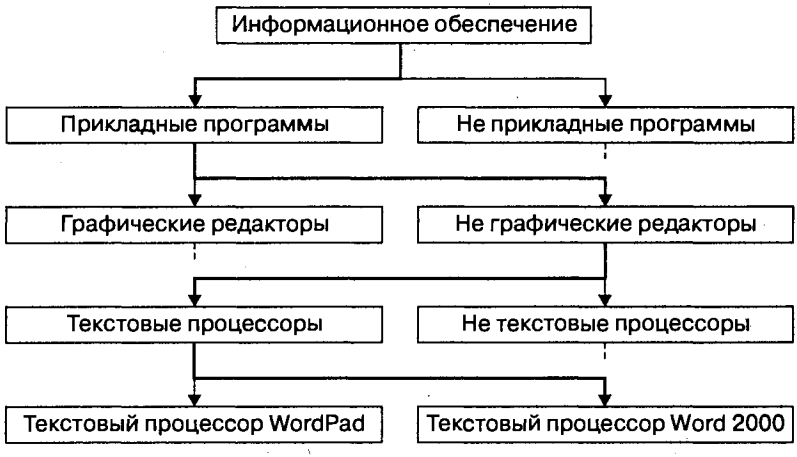


Рис. 1.6. Пример, поясняющий принцип действия метода дихотомии

В иерархической структуре, построенной методом дихотомии, путь доступа к любому элементу можно представить как путь через рациональный лабиринт с поворотами налево (0) или направо (1) и, таким образом, выразить путь доступа в виде компактной двоичной записи. В нашем примере путь доступа к текстовому процессору Word 2000 выразится следующим двоичным числом: 1010.

Упорядочение структур данных

Списочные и табличные структуры являются простыми. Ими легко пользоваться, поскольку адрес каждого элемента задается числом (для списка), двумя числами (для двумерной таблицы) или несколькими числами для многомерной таблицы. Они также легко упорядочиваются. Основным методом упорядочения является *сортировка*. Данные можно сортировать по любому избранному критерию, например: по алфавиту, по возрастанию порядкового номера или по возрастанию какого-либо параметра.

Несмотря на многочисленные удобства, у простых структур данных есть и недостаток — их трудно обновлять. Если, например, перевести студента из одной группы в другую, изменения надо вносить сразу в два журнала посещаемости; при этом в обоих журналах будет нарушена списочная структура. Если переведенного студента вписать в конец списка группы, нарушится упорядочение по алфавиту, а если его вписать в соответствии с алфавитом, то изменятся порядковые номера всех студентов, которые следуют за ним.

Таким образом, *при добавлении произвольного элемента в упорядоченную структуру списка может происходить изменение адресных данных у других элементов*. В журналах успеваемости это пережить нетрудно, но в системах, выполняющих автоматическую обработку данных, нужны специальные методы для решения этой проблемы.

Иерархические структуры данных по форме сложнее, чем линейные и табличные, но они не создают проблем с обновлением данных. Их легко развивать путем создания новых уровней. Даже если в учебном заведении будет создан новый факультет, это никак не отразится на пути доступа к сведениям об учащихся прочих факультетов.

Недостатком иерархических структур является относительная трудоемкость записи адреса элемента данных и сложность упорядочения. Часто методы упорядочения в таких структурах основывают на предварительной *индексации*, которая заключается в том, что каждому элементу данных присваивается свой уникальный индекс, который можно использовать при поиске, сортировке и т. п. Ранее рассмотренный принцип дихотомии на самом деле является одним из методов индексации данных в иерархических структурах. После такой индексации данные легко разыскиваются по двоичному коду связанного с ними индекса.

Адресные данные. Если данные хранятся не как попало, а в организованной структуре (причем любой), то каждый элемент данных приобретает новое свойство (параметр), который можно назвать *адресом*. Конечно, работать с упорядоченными данными удобнее, но за это приходится платить их размножением, поскольку адреса элементов данных — это тоже данные и их тоже надо хранить и обрабатывать.

1.3. Файлы и файловая структура

Единицы представления данных

Существует множество систем представления данных. С одной из них, принятой в информатике и вычислительной технике, двоичным кодом, мы познакомились выше. Наименьшей единицей такого представления является бит (*двоичный разряд*).

Совокупность двоичных разрядов, выражающих числовые или иные данные, образует некий битовый рисунок. Практика показывает, что с битовым представлением удобнее работать, если этот рисунок имеет регулярную форму. В настоящее время в качестве таких форм используются группы из восьми битов, которые называются *байтами*.

Десятичное число	Двоичное число	Байт
1	1	0000 0001
2	10	0000 0010
...	...	...
255	11111111	1111 1111

Понятие о байте как группе взаимосвязанных битов появилось вместе с первыми образцами электронной вычислительной техники. Долгое время оно было *машинно-зависимым*, то есть для разных вычислительных машин длина байта была разной. Только в конце 60-х годов понятие байта стало универсальным и *машиннонезависимым*.

Выше мы видели, что во многих случаях целесообразно использовать не восьми-разрядное кодирование, а 16-разрядное, 24-разрядное, 32-разрядное и более. Группа из 16 взаимосвязанных бит (двух взаимосвязанных байтов) в информатике называется *словом*. Соответственно, группы из четырех взаимосвязанных байтов (32 разряда) называются *удвоенным словом*, а группы из восьми байтов (64 разряда) — *учетверенным словом*. Пока, на сегодняшний день, такой системы обозначения достаточно.

Единицы измерения данных

Существует много различных систем и единиц измерения данных. Каждая научная дисциплина и каждая область человеческой деятельности может использовать свои, наиболее удобные или традиционно устоявшиеся единицы. В информатике для измерения данных используют тот факт, что разные типы данных имеют универсальное двоичное представление и потому вводят свои единицы данных, основанные на нем.

Наименьшей единицей измерения является байт. Поскольку одним байтом, как правило, кодируется один символ текстовой информации, то для текстовых документов размер в байтах соответствует лексическому объему в символах (пока исключение представляет рассмотренная выше универсальная кодировка *UNICODE*).

Более крупная единица измерения — килобайт (Кбайт). Условно можно считать, что 1 Кбайт примерно равен 1000 байт. Условность связана с тем, что для вычислительной техники, работающей с двоичными числами, более удобно представление чисел в виде степени двойки и потому на самом деле 1 Кбайт равен 2^{10} байт (1024 байт). Однако всюду, где это не принципиально, с инженерной погрешностью (до 3 %) «забывают» о «лишних» байтах.

В килобайтах измеряют сравнительно небольшие объемы данных. Условно можно считать, что одна страница неформатированного машинописного текста составляет около 2 Кбайт.

Более крупные единицы измерения данных образуются добавлением префиксов *мега-*, *гига- тера-*; в более крупных единицах пока нет практической надобности.

1 Мбайт = 1024 Кбайт = 10^{20} байт

1 Гбайт = 1024 Мбайт = 10^{30} байт

1 Тбайт = 1024 Гбайт = 10^{40} байт

Особо обратим внимание на то, что при переходе к более крупным единицам «инженерная» погрешность, связанная с округлением, накапливается и становится недопустимой, поэтому на старших единицах измерения округление производится реже.

Единицы хранения данных

При хранении данных решаются две проблемы: как сохранить данные в наиболее компактном виде и как обеспечить к ним удобный и быстрый доступ (если доступ не обеспечен, то это не хранение). Для обеспечения доступа необходимо, чтобы данные имели упорядоченную структуру, а при этом, как мы уже знаем, образуется «паразитная нагрузка» в виде адресных данных. Без них нельзя получить доступ к нужным элементам данных, входящих в структуру.

Поскольку адресные данные тоже имеют размер и тоже подлежат хранению, хранить данные в виде мелких единиц, таких как байты, неудобно. Их неудобно хранить и в более крупных единицах (килобайтах, мегабайтах и т. п.), поскольку неполное заполнение одной единицы хранения приводит к неэффективности хранения.

В качестве единицы хранения данных принят объект переменной длины, называемый *файлом*. *Файл — это последовательность произвольного числа байтов, обладающая уникальным собственным именем.* Обычно в отдельном файле хранят данные, относящиеся к одному типу. В этом случае тип данных определяет *тип файла*.

Проще всего представить себе файл в виде безразмерного канцелярского досье, в которое можно по желанию добавлять содержимое или извлекать его оттуда. Поскольку в определении файла нет ограничений на размер, можно представить себе файл, имеющий 0 байтов (*пустой файл*), и файл, имеющий любое число байтов.

В определении файла особое внимание уделяется имени. Оно фактически несет в себе адресные данные, без которых данные, хранящиеся в файле, не станут информацией из-за отсутствия метода доступа к ним. Кроме функций, связанных с адресацией, имя файла может хранить и сведения о типе данных, заключенных в нем. Для автоматических средств работы с данными это важно, поскольку по имени файла они могут автоматически определить адекватный метод извлечения информации из файла.

Понятие о файловой структуре

Требование уникальности имени файла очевидно — без этого невозможно гарантировать однозначность доступа к данным. В средствах вычислительной техники требование уникальности имени обеспечивается автоматически — создать файл с именем, тождественным уже имеющемуся, не может ни пользователь, ни автоматика.

Хранение файлов организуется в иерархической структуре, которая в данном случае называется *файловой структурой*. В качестве вершины структуры служит имя носителя, на котором сохраняются файлы. Далее файлы группируются в *каталоги* (*папки*), внутри которых могут быть созданы *вложенные каталоги* (*папки*). *Путь доступа к файлу* начинается с имени устройства и включает все имена каталогов (папок), через которые проходит. В качестве разделителя используется символ «\» (обратная косая черта).

Уникальность имени файла обеспечивается тем, что *полным именем файла считается собственное имя файла вместе с путем доступа к нему*. Понятно, что в этом случае на одном носителе не может быть двух файлов с тождественными полными именами.

Пример записи полного имени файла:

<имя носителя>\<имя каталога-1>\...\<имя каталога-N>\<собственное имя файла>

Вот пример записи двух файлов, имеющих одинаковое собственное имя и размещенных на одном носителе, но отличающихся путем доступа, то есть полным именем. Для наглядности имена каталогов (папок) напечатаны прописными буквами.

C:\АВТОМАТИЧЕСКИЕ АППАРАТЫ\ВЕНЕРА\АТМОСФЕРА\Результаты исследований

C:\РАДИОЛОКАЦИЯ\ВЕНЕРА\РЕЛЬЕФ\Результаты исследований

О том, как на практике реализуются файловые структуры, мы узнаем несколько позже, когда познакомимся со средствами вычислительной техники и с понятием *файловой системы*.

1.4. Информатика

Предмет и задачи информатики

Информатика — это техническая наука, систематизирующая приемы создания, хранения, воспроизведения, обработки и передачи данных средствами вычислительной техники, а также принципы функционирования этих средств и методы управления ими.

Из этого определения видно, что информатика очень близка к технологии, поэтому ее предмет нередко называют *информационной технологией*.

Предмет информатики составляют следующие понятия:

- аппаратное обеспечение средств вычислительной техники;
- программное обеспечение средств вычислительной техники;
- средства взаимодействия аппаратного и программного обеспечения;
- средства взаимодействия человека с аппаратными и программными средствами.

Как видно из этого списка, в информатике особое внимание уделяется вопросам *взаимодействия*. Для этого даже есть специальное понятие — *интерфейс*. Методы и средства взаимодействия человека с аппаратными и программными средствами называют *пользовательским интерфейсом*. Соответственно, существуют *аппаратные интерфейсы*, *программные интерфейсы* и *аппаратно-программные интерфейсы*.

Основной задачей информатики является систематизация приемов и методов работы с аппаратными и программными средствами вычислительной техники. *Цель* систематизации состоит в выделении, внедрении и развитии передовых, наиболее эффективных технологий, в автоматизации этапов работы с данными, а также в методическом обеспечении новых технологических исследований.

Информатика — практическая наука. Ее достижения должны проходить подтверждение практикой и приниматься в тех случаях, когда они соответствуют критерию повышения эффективности. В составе основной задачи информатики сегодня можно выделить следующие направления для практических приложений:

- архитектура вычислительных систем (приемы и методы построения систем, предназначенных для автоматической обработки данных);
- интерфейсы вычислительных систем (приемы и методы управления аппаратным и программным обеспечением);
- программирование (приемы, методы и средства разработки компьютерных программ);
- преобразование данных (приемы и методы преобразования структур данных);
- защита информации (обобщение приемов, разработка методов и средств защиты данных);
- автоматизация (функционирование программно-аппаратных средств без участия человека);
- стандартизация (обеспечение совместимости между аппаратными и программными средствами, а также между форматами представления данных, относящихся к различным типам вычислительных систем).

На всех этапах технического обеспечения информационных процессов для информатики ключевым понятием является *эффективность*. Для аппаратных средств под эффективностью понимают отношение производительности оборудования к его стоимости (с учетом стоимости эксплуатации и обслуживания). Для программного обеспечения под эффективностью понимают производительность лиц, работающих с ними (пользователей). В программировании под эффективностью понимают объем программного кода, создаваемого программистами в единицу времени.

В информатике все жестко ориентировано на эффективность. Вопрос, *как сделать ту или иную операцию*, для информатики является важным, но не основным. Основным же является вопрос, *как сделать данную операцию эффективно*.

Истоки и предпосылки информатики

Слово *информатика* происходит от французского слова *Informatique*, образованного в результате объединения терминов *Informacion* (*информация*) и *Automatique* (*автоматика*), что выражает ее суть как науки об автоматической обработке информации. Кроме Франции термин *информатика* используется в ряде стран Восточной Европы. В то же время, в большинстве стран Западной Европы и США используется другой термин — *Computer Science* (*наука о средствах вычислительной техники*).

В качестве источников информатики обычно называют две науки — *документалистику* и *кибернетику*. Документалистика сформировалась в конце XIX века в связи с бурным развитием производственных отношений. Ее расцвет пришелся на 20–30-е годы XX века, а основным предметом стало изучение рациональных средств и методов повышения эффективности документооборота.

Основы близкой к информатике технической науки *кибернетики* были заложены трудами по математической логике американского математика Норберта Винера, опубликованными в 1948 году, а само название происходит от греческого слова (*kybernetic os* — искусный в управлении).

Впервые термин *кибернетика* ввел французский физик Андре Мари Ампер в первой половине XIX веке. Он занимался разработкой единой системы классификации всех наук и обозначил этим термином гипотетическую науку об управлении, которой в то время не существовало, но которая, по его мнению, должна была существовать.

Сегодня предметом кибернетики являются принципы построения и функционирования систем автоматического управления, а основными задачами — методы моделирования процесса принятия решений техническими средствами, связь между психологией человека и математической логикой, связь между информационным процессом отдельного индивидуума и информационными процессами в обществе, разработка принципов и методов искусственного интеллекта. На практике кибернетика во многих случаях опирается на те же программные и аппаратные средства вычислительной техники, что и информатика, а информатика, в свою очередь, заимствует у кибернетики математическую и логическую базу для развития этих средств.

Подведение итогов

Все процессы в природе сопровождаются *сигналами*. Зарегистрированные сигналы образуют *данные*. Данные преобразуются, транспортируются и потребляются с помощью *методов*. При взаимодействии данных и адекватных им методов образуется *информация*. Информация — это динамический объект, образующийся в ходе *информационного процесса*. Он отражает диалектическую связь между объективными данными и субъективными методами. Свойства информации зависят как от свойств данных, так и от свойств методов.

Данные различаются *типами*, что связано с различиями в физической природе сигналов, при регистрации которых образовались данные. В качестве средства хранения и транспортировки данных используются *носители данных*. Для удобства операций с данными их структурируют. Наиболее широко используются следующие структуры: *линейная*, *табличная* и *иерархическая* — они различаются методом адресации к данным. При сохранении данных образуются данные нового типа — *адресные данные*.

Вопросами систематизации приемов и методов создания, хранения, воспроизведения, обработки и передачи данных средствами вычислительной техники занимается техническая наука — *информатика*. С целью унификации приемов и методов работы с данными в вычислительной технике применяется универсальная система кодирования данных, называемая *двоичным кодом*. Элементарной единицей представления

данных в двоичном коде является *двоичный разряд* (*бит*). Другой, более крупной единицей представления данных является *байт*.

Основной единицей хранения данных является *файл*. Файл представляет собой последовательность байтов, имеющую собственное имя. Совокупность файлов образует файловую структуру, которая, как правило, относится к иерархическому типу. *Полный адрес* файла в файловой структуре является уникальным и включает в себя собственное имя файла и путь доступа к нему.

Вопросы для самоконтроля

1. Как вы можете объяснить бытовой термин «переизбыток информации»? Что имеется в виду: излишняя полнота данных; излишняя сложность методов; неадекватность поступающих данных и методов, имеющихся в наличии?
2. Как вы понимаете термин «средство массовой информации»? Что это? Средство массовой поставки данных? Средство, обеспечивающее массовое распространение методов? Средство, обеспечивающее процесс информирования путем поставки данных гражданам, обладающим адекватными методами их потребления?
3. Как вы полагаете, являются ли данные товаром? Могут ли методы быть товаром?
4. На примере коммерческих структур, обеспечивающих коммуникационные услуги, покажите, как взаимодействуют между собой маркетинг данных и маркетинг методов? Можете ли вы привести примеры лизинга данных и методов?
5. Как вы понимаете диалектическое единство данных и методов? Можете ли вы привести примеры аналогичного единства двух понятий из других научных дисциплин: естественных, социальных, технических?
6. Как вы понимаете динамический характер информации? Что происходит с ней по окончании информационного процесса?
7. Можем ли мы утверждать, что данные, полученные в результате информационного процесса, адекватны исходным? Почему? От каких свойств исходных данных и методов зависит адекватность результирующих данных?
8. Что такое *вектор данных*? Является ли список номеров телефонов населенного пункта вектором данных? Является ли вектором данных текстовый документ, закодированный двоичным кодом, если он не содержит элементов оформления?
9. Является ли цифровой код цветного фотоснимка вектором данных? Если нет, то чего ему не хватает?
10. Как вы понимаете следующие термины: *аппаратно-программный интерфейс, программный интерфейс, аппаратный интерфейс?* Как бы вы назвали специальность людей, разрабатывающих аппаратные интерфейсы? Как называется специальность людей, разрабатывающих программные интерфейсы?
11. На основе личных наблюдений сделайте вывод о том, какими средствами может пользоваться преподаватель для обеспечения интерфейса с аудиторией. Можете ли вы рассмотреть отдельно методические и технические средства, имеющиеся в его распоряжении? Может ли преподаватель рассматривать *вашу* тетрадь и авторучку как *свое* средство обеспечения интерфейса? Если да, то в какой мере?

Глава 5. Основы работы с операционной системой Windows XP

В предыдущей главе мы рассмотрели функции ряда операционных систем и требования к ним. Надо сказать, что многие из этих требований являются противоречивыми. Например, соотношение требований безотказности и совместимости с приложениями иных систем — это вопрос баланса. Соотношение требований безопасности и простоты обеспечения сетевых функций — это тоже вопрос баланса. На каждом конкретном рабочем месте эти вопросы решаются индивидуально.

В этом смысле сегодня особое место занимает операционная система *Windows XP*. Она обладает наибольшей универсальностью, имеет самое широкое распространение и, соответственно, получает особую поддержку со стороны производителей аппаратного и программного обеспечения. Для компьютера, работающего в этой системе, наиболее просто подобрать прикладные программы и драйверы устройств.

Почти все, что здесь сказано об операционной системе *Windows XP*, можно отнести и к другим операционным системам семейства *Windows*. В том, что касается приемов и методов работы, они в значительной степени совпадают.

5.1. Основные объекты и приемы управления Windows

Windows XP является графической операционной системой для компьютеров платформы *IBM PC*. Ее основные средства управления — графический манипулятор (мышь или иной аналогичный) и клавиатура. Система предназначена для управления автономным компьютером, но также содержит все необходимое для создания небольшой локальной компьютерной сети (*одноранговой сети*) и имеет средства для интеграции компьютера во всемирную сеть (*Интернет*).

Рабочий стол Windows XP

Стартовый экран *Windows XP* представляет собой системный объект, называемый *Рабочим столом*. Практически, экран *Windows XP* является Рабочим столом. Однако существуют видеоадаптеры, позволяющие создать Рабочий стол, размер которого больше, чем видимый размер экрана. Кроме того, *Windows XP* имеет штатные сред-

ства, позволяющие разместить Рабочий стол на нескольких экранах, если к компьютеру подключено несколько мониторов.

Рабочий стол — это *графическая среда*, на которой отображаются *объекты Windows* и *элементы управления Windows*. Все, с чем мы имеем дело, работая с компьютером в данной системе, можно отнести либо к *объектам*, либо к *элементам управления*. В исходном состоянии на Рабочем столе можно наблюдать несколько экранных значков и Панель задач (рис. 5.1). Значки — это графическое представление *объектов Windows*, а Панель задач — один из основных *элементов управления*.

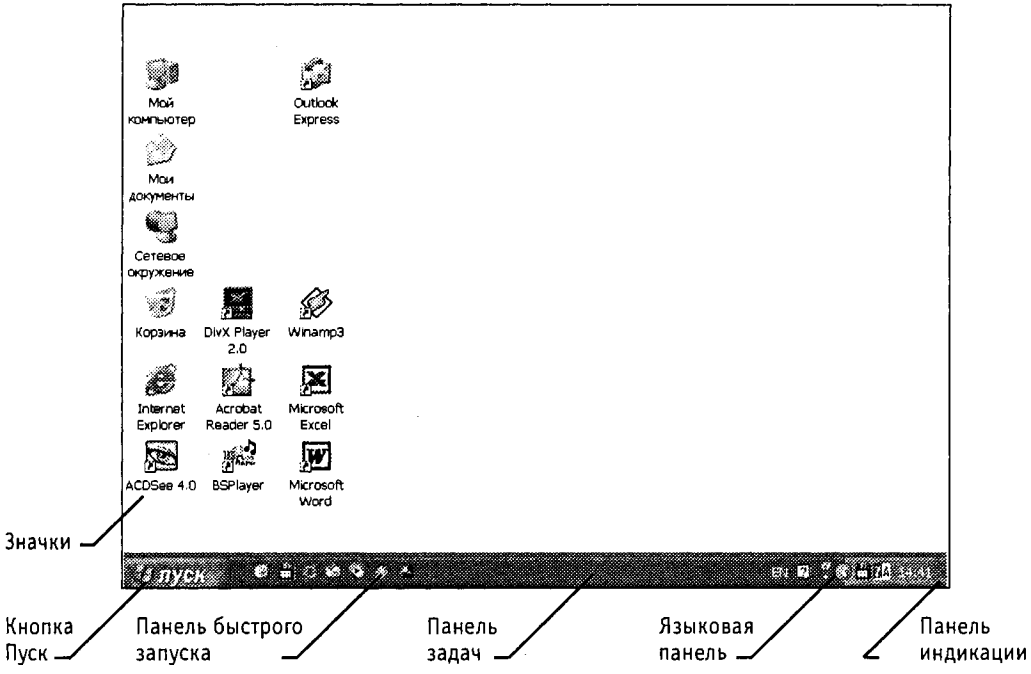


Рис. 5.1. Рабочий стол Windows XP

Управление Windows XP

В *Windows XP* большую часть команд можно выполнять с помощью мыши. С мышью связан активный элемент управления — *указатель мыши*. При перемещении мыши по плоской поверхности указатель перемещается по Рабочему столу, и его можно *позиционировать* на значках объектов или на пассивных элементах управления приложений.

Основными приемами управления с помощью мыши являются:

- *щелчок* — быстрое нажатие и отпускание левой кнопки мыши;
- *двойной щелчок* — два щелчка, выполненные с малым интервалом времени между ними;
- *щелчок правой кнопкой* — то же, что и *щелчок*, но с использованием правой кнопки;

- *перетаскивание* (*drag-and-drop*) — выполняется путем перемещения мыши при нажатой левой кнопке (обычно сопровождается перемещением экранного объекта, на котором установлен указатель);
- *протягивание мыши* (*click-and-drag*) — выполняется, как и *перетаскивание*, но при этом происходит не перемещение экранного объекта, а изменение его формы;
- *специальное перетаскивание* — выполняется, как и *перетаскивание*, но при нажатой правой кнопке мыши, а не левой;
- *зависание* — наведение указателя мыши на значок объекта или на элемент управления и задержка его на некоторое время (при этом обычно на экране появляется *всплывающая подсказка*, кратко характеризующая свойства объекта).

Значки и ярлыки объектов

Создание ярлыков объектов — это одна из функций приема специального перетаскивания, но нам надо пояснить, что же такое *ярлык*. Рассмотрим это понятие на примере Корзины.

Корзина — специальный объект *Windows*, выполняющий функции *контейнера*. Она служит для временного хранения удаляемых объектов. Если какой-то документ или программа стали не нужны, их можно удалить, но при этом они не удаляются безвозвратно, а откладываются в Корзину, из которой их впоследствии можно восстановить.

Откройте окно Мой Компьютер и попробуйте перетащить в него значок Корзины. Это не получится, поскольку Корзина — *реквизитный значок* Рабочего стола. Невозможность перетаскивания отображается специальным указателем мыши.

Теперь попробуйте перетащить значок Корзины в окно Мои документы. Обратите внимание на то, что возле указателя мыши появляется небольшая стрелочка, которая показывает, что при отпускании кнопки мыши будет создан *ярлык* — копия значка Корзина со стрелкой в левом нижнем углу. Ярлыком можно пользоваться точно так же, как обычно пользуются значками.

Значок является *графическим представлением объекта*. То, что мы делаем со значком, мы на самом деле делаем с объектом. Например, удаление значка приводит к удалению объекта; копирование значка приводит к копированию объекта и т. д. Ярлык же является только *указателем* на объект. Удаление ярлыка приводит к удалению указателя, но не объекта; копирование ярлыка приводит к копированию указателя, но не объекта.

Для пользователя приемы работы с ярлыками ничем не отличается от приемов работы со значками. Точно так же можно запускать программы двойным щелчком на их ярлыках, так же можно и открывать документы. Зато ярлыки позволяют экономить место на жестком диске.

Если объект (например, файл с текстовым документом) имеет большой размер, то его многократное копирование в различные окна папок привело бы фактически к появлению новых объектов (копий файла). При этом многократно увеличился бы

расход рабочего пространства на жестком диске, а у пользователя появились бы сложнейшие заботы по синхронизации содержимого этих копий (при редактировании одной копии ее изменения без специальных мер никак не отразятся на содержимом других копий).

С другой стороны, ярлык является лишь указателем, он занимает ничтожно мало места, и его размножение позволяет обеспечить удобный доступ к связанному с ним объекту из разных мест операционной системы. При этом расход рабочего пространства на жестком диске ничтожен, и нет проблем с синхронизацией данных. Из какой бы папки ни открывался документ щелчком на его ярлыке, редактированию всегда подвергается только один связанный с ним объект.

5.2. Файлы и папки Windows

Способ хранения файлов на дисках компьютера называется *файловой системой*. Иерархическая структура, в виде которой операционная система отображает файлы и папки диска, называется *файловой структурой*. Как все дисковые операционные системы, *Windows XP* предоставляет средства для управления этой структурой.

Просмотр папок Windows

Откройте окно Мой компьютер и найдите в нем значок жесткого диска C:. Щелкните на нем дважды, и на экране откроется новое окно, в котором представлены значки объектов, присутствующих на жестком диске. Обратите внимание на значки, представляющие папки, и значки, представляющие файлы. Двойной щелчок на значке любой папки открывает ее окно и позволяет ознакомиться с содержимым. Так можно погружаться вглубь структуры папок до последнего уровня вложения. В соответствующем окне будут представлены только значки файлов.

Окно папки

Окно папки — это *контейнер*, содержимое которого графически отображает содержимое папки. Любую папку *Windows* можно открыть в своем окне. Количество одновременно открытых окон может быть достаточно большим — это зависит от параметров конкретного компьютера. Окна — одни из самых важных объектов *Windows*. Абсолютно все операции, которые мы делаем, работая с компьютером, происходят либо на Рабочем столе, либо в каком-либо окне.

Окна папок — не единственный тип окон в *Windows*. По наличию однородных элементов управления и оформления можно выделить и другие типы окон: *диалоговые окна*, *окна справочной системы* и *рабочие окна приложений*, а внутри окон многих приложений могут существовать отдельные окна документов (если приложение позволяет работать с несколькими документами одновременно).

Если подходить к терминологии с академической строгостью, то за каждым открытым окном скрывается некое работающее приложение (принято говорить *процесс*) и все окна можно было бы назвать окнами приложений (*окнами процессов*), но в учебных целях их лучше все-таки рассматривать порознь.

Структура окна

На рис. 5.2 представлено окно папки \Windows. Такая папка обычно имеется на всех компьютерах, работающих в любой операционной системе семейства *Windows*. Окно папки содержит следующие обязательные элементы.

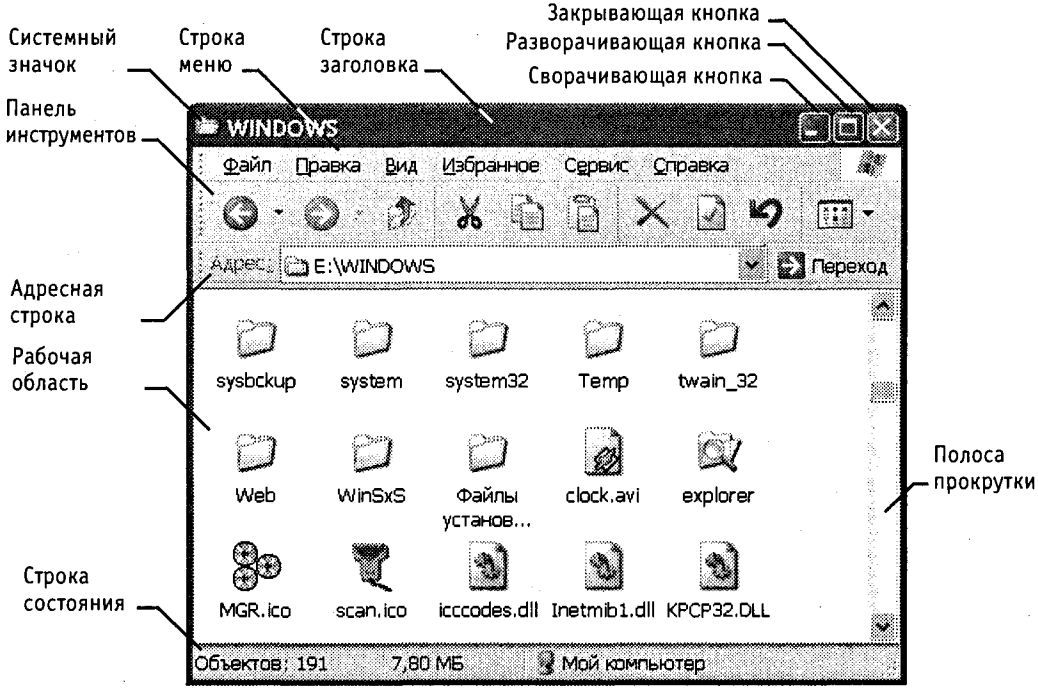


Рис. 5.2. Окно папки Windows

Строка заголовка — в ней написано название папки. За эту строку выполняется перетаскивание папки на Рабочем столе с помощью мыши.

Системный значок. Находится в левом верхнем углу любого окна папки. При щелчке на этом значке открывается меню, называемое *служебным*. Команды, представленные в данном меню, позволяют управлять размером и расположением окна на Рабочем столе — они могут быть полезны, если мышь не работает.

Кнопки управления размером. Эти кнопки дублируют основные команды служебного меню. В операционной системе *Windows XP* исключительно много дублирования. Большинство операций можно выполнить многими различными способами. Каждый пользуется теми приемами, которые ему удобны. Кнопок управления размером три: *закрывающая, сворачивающая, разворачивающая.*

Щелчок на закрывающей кнопке закрывает окно полностью (и прекращает процесс). Щелчок на сворачивающей кнопке приводит к тому, что окно сворачивается до размера кнопки, которая находится на Панели задач (при этом процесс, связанный

с окном, не прекращается). В любой момент окно можно восстановить щелчком на кнопке Панели задач.

Щелчок на разворачивающей кнопке разворачивает окно на полный экран. При этом работать с ним удобно, но доступ к прочим окнам затрудняется. В развернутом окне разворачивающая кнопка сменяется восстанавливающей, с помощью которой можно восстановить исходный размер окна.

Строка меню. Для окон папок строка меню имеет стандартный вид. При щелчке на каждом из пунктов этого меню открывается «ниспадающее» меню, пункты которого позволяют проводить операции с содержимым окна или с окном в целом.

Использование команд, доступных через строку меню, в большинстве случаев не самый эффективный прием работы в *Windows* (есть и более удобные элементы и средства управления), но зато строка меню гарантированно предоставляет *доступ ко всем командам*, которые можно выполнить в данном окне. Это удобно, если неизвестно, где находится нужный элемент управления. Поэтому при изучении работы с новым приложением в первое время принято пользоваться командами строки меню и лишь потом переходить к использованию других средств управления, постепенно повышая эффективность работы.

Панель инструментов. Содержит командные кнопки для выполнения наиболее часто встречающихся операций. В работе удобнее, чем строка меню, но ограничена по количеству команд. В окнах современных приложений панель инструментов часто бывает настраиваемой. Пользователь сам может разместить на ней те командные кнопки, которыми он пользуется чаще всего.

Адресная строка. В ней указан путь доступа к текущей папке, что удобно для ориентации в файловой структуре. Адресная строка позволяет выполнить быстрый переход к другим разделам файловой структуры с помощью раскрывающей кнопки на правом краю строки.

Рабочая область. В ней отображаются значки объектов, хранящихся в папке, причем способом отображения можно управлять (см. ниже). В окнах приложений в рабочей области размещаются окна документов и рабочие панели.

Полосы прокрутки. Если количество объектов слишком велико (или размер окна слишком мал), по правому и нижнему краям рабочей области могут отображаться полосы прокрутки, с помощью которых можно «прокручивать» содержимое папки в рабочей области.

Полоса прокрутки имеет движок и две концевые кнопки. Прокрутку выполняют тремя способами:

- щелчком на одной из концевых кнопок;
- перетаскиванием движка;
- щелчком на полосе прокрутке выше или ниже движка.

Строка состояния. Здесь выводится дополнительная, часто немаловажная информация. Так, например, если среди объектов, представленных в окне, есть скрытые или системные, они могут не отображаться при просмотре, но в строке состояния об их наличии имеется специальная запись.

5.3. Операции с файловой структурой

К основным операциям с файловой структурой относятся:

- навигация по файловой структуре;
- запуск программ и открытие документов;
- создание папок;
- копирование файлов и папок;
- перемещение файлов и папок;
- удаление файлов и папок;
- переименование файлов и папок;
- создание ярлыков.

Система окон Мой компьютер

Все операции с файлами и папками в *Windows XP* можно выполнять несколькими различными способами. Каждый выбирает себе те приемы, которые ему кажутся наиболее удобными. Обычно с приобретением опыта работы на компьютере совокупность используемых приемов меняется.

Простейшие приемы работы с файловой структурой предоставляет иерархическая система окон папок, берущая свое начало от известной нам папки \Мой компьютер. Диски, представленные в окне этой папки, можно открыть, а потом разыскать на них любые нужные папки и файлы. Копирование и перемещение файлов и папок из одной папки в другую можно выполнять путем перетаскивания их значков из окна одной папки в окно другой. Для удаления объектов можно использовать перетаскивание на значок Корзины, а можно пользоваться контекстным меню, которое открывается при щелчке правой кнопкой мыши на объекте. Для создания в папке ярлыка документа или программы можно использовать специальное перетаскивание или команду Создать ▸ Ярлык из контекстного меню.

При таком подходе к операциям с файловой структурой следует иметь в виду несколько замечаний.

1. В *Windows XP* на экране обычно присутствует только одно окно папки. Если в окне папки открыть вложенную папку, то ее окно замещает предыдущее. Это неудобно, если надо выполнять операции перетаскивания между окнами. Чтобы каждая папка открывалась в собственном окне, надо включить следующий переключатель: Пуск ▸ Настройка ▸ Панель управления ▸ Свойства папки ▸ Общие ▸ Открывать каждую папку в отдельном окне.
2. При перетаскивании значков объектов между папками, принадлежащими одному диску, автоматически выполняется *перемещение* объектов. Если нужно выполнить копирование, используют специальное перетаскивание.
3. При перетаскивании значков объектов между папками, принадлежащими разным дискам, автоматически выполняется *копирование* объектов. Если нужно выполнить перемещение, используют специальное перетаскивание.

Программа Проводник

Работа с файловой системой в окнах папок не вполне удобна, но для этой цели есть и более мощное средство — программа Проводник.

Проводник — служебная программа, относящаяся к категории *диспетчеров файлов*. Она предназначена для навигации по файловой структуре компьютера и ее обслуживания. Проводник очень глубоко интегрирован в операционную систему *Windows*. По сути, мы работаем с ним даже тогда, когда его не видим. Если по щелчку правой кнопкой мыши на каком-либо объекте мы получаем контекстное меню, это результат невидимой работы Проводника. Если при перетаскивании объектов из одного окна в другое происходит их копирование или перемещение, это тоже результат заочной деятельности Проводника. Однако с ним можно работать и «очно». Программа запускается командой Пуск ▸ Программы ▸ Стандартные ▸ Проводник.

Окно программы Проводник представлено на рис. 5.3. Как видно из рисунка, по элементам управления это окно очень похоже на окна папок. Основное отличие в том, что окно Проводника имеет не одну рабочую область, а две: левую панель, называемую *панелью папок*, и правую панель, называемую *панелью содержимого*.

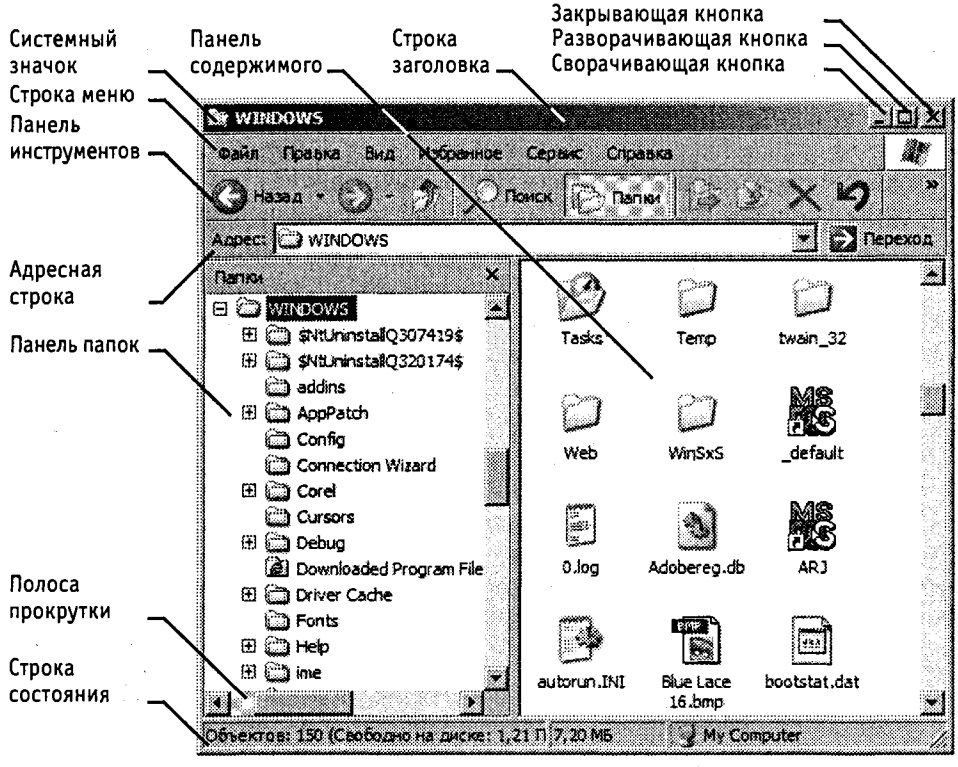


Рис. 5.3. Окно программы Проводник

Навигация по файловой структуре. Цель навигации состоит в обеспечении доступа к нужной папке и ее содержимому. Мы специально не говорим о том, что цель

навигации — это *поиск* нужных файлов и папок, поскольку для этой операции есть специальные средства.

Навигацию по файловой структуре выполняют на левой панели Проводника, на которой показана структура папок. Папки могут быть *развернуты* или *свернуты*, а также *раскрыты* или *закрыты*. Если папка имеет вложенные папки, то на левой панели рядом с папкой отображается *узел*, отмеченный знаком «+». Щелчок на узле разворачивает папку, при этом значок узла меняется на «–». Таким же образом папки и сворачиваются.

Для того чтобы раскрыть папку, надо щелкнуть на ее значке. Содержимое раскрытой папки отображается на правой панели. Одна из папок на левой панели раскрыта всегда. Закрыть папку щелчком на ее значке невозможно — она закроется автоматически при раскрытии любой другой папки.

Запуск программ и открытие документов. Эта операция выполняется двойным щелчком на значке программы или документа на правой панели Проводника. Если нужный объект на правой панели не показан, надо выполнить навигацию на левой панели и найти папку, в которой он находится.

Создание папок. Чтобы создать новую папку, сначала следует на левой панели Проводника раскрыть папку, внутри которой она будет создана. После этого надо перейти на правую панель, щелкнуть правой кнопкой мыши на свободном от значков месте и выбрать в контекстном меню пункт Создать ▸ Папку. На правой панели появится значок папки с названием Новая папка. Название выделено, и в таком состоянии его можно редактировать. После того как папка будет создана, она войдет в состав файловой структуры, отображаемой на левой панели.

Копирование и перемещение файлов и папок. Папку, из которой происходит копирование, называют *источником*. Папку, в которую происходит копирование, называют *приемником*. Копирование выполняют методом перетаскивания значка объекта с правой панели Проводника на левую.

Первая задача — найти и раскрыть папку-источник, чтобы на правой панели был виден копируемый объект. Вторая задача — найти на левой панели папку-приемник, но раскрывать ее не надо. Далее объект перетаскивают с правой панели на левую и помещают на значок папки-приемника. Эта операция требует аккуратности, поскольку попасть одним значком точно на другой не всегда просто. Для контроля точности попадания надо следить за названием папки-приемника. В тот момент, когда наведение выполнено правильно, подпись под значком меняет цвет, и кнопку мыши можно отпускать.

Если и папка-источник, и папка-приемник принадлежат одному диску, то при перетаскивании выполняется перемещение, а если разным — то копирование. В тех случаях, когда нужно обратное действие, выполняют специальное перетаскивание при нажатой правой кнопке мыши.

Удаление файлов и папок. Работа начинается с навигации. На левой панели открывают папку, содержащую удаляемый объект, а на правой панели выделяют нужный объект (или группу объектов).

Удаление можно выполнять несколькими способами. Классический способ — с помощью команды Файл ▸ Удалить из строки меню (если ни один объект не выделен, эта команда не активируется). Более удобный способ — использовать командную кнопку на панели инструментов. Еще более удобно воспользоваться контекстным меню. Щелкните правой кнопкой мыши на удаляемом объекте и выберите в контекстном меню пункт Удалить. Однако самый удобный способ удаления выделенного объекта состоит в использовании клавиши DELETE клавиатуры.

Использование манипуляторов, таких, как мышь, — это важное достоинство графических операционных систем. Однако профессионалами давно отмечено, что наивысшая производительность труда и минимальное утомление при работе достигаются при максимальном использовании клавиатуры. Для команд, представленных в строке меню, часто приводятся клавиатурные комбинации, которыми эти команды можно выполнить. Обращайте на них внимание, запоминайте их и старайтесь постепенно переходить к их использованию. Это один из приемов закрепления навыков профессиональной работы с компьютером.

Создание ярлыков объектов. Ярлыки объектов можно создавать двумя способами: методом специального перетаскивания (вручную) или с помощью специальной программы-мастера (автоматически). С приемом специального перетаскивания мы уже знакомы. Объект выбирается на правой панели Проводника и перетаскивается при нажатой правой кнопке мыши на значок нужной папки на левой панели. В момент отпускания кнопки на экране появляется меню, в котором надо выбрать пункт Создать ярлык.

Второй способ (с использованием мастера) менее нагляден, но во многих случаях более удобен. *Мастерами* в системе *Windows* называют специальные программы, работающие в режиме диалога с пользователем. Диалог строится по принципу «запрос — ответ». Если на все запросы от программы даны корректные ответы, программа автоматически выполнит черновую работу.

1. Для того чтобы запустить Мастер создания ярлыка, надо щелкнуть правой кнопкой мыши в окне той папки, в которой создается ярлык объекта.
2. В открывшемся контекстном меню следует выбрать пункт Создать ▸ Ярлык — произойдет запуск мастера.
3. В диалоговом окне мастера имеется командная строка, в поле которой следует ввести путь доступа к объекту, для которого создается ярлык, например \Windows\System32\Calc.exe — путь доступа к стандартной программе Калькулятор. Разумеется, пользователь не может помнить пути доступа ко всем нужным объектам, поэтому ввод адреса автоматизирован. Для этого служит командная кнопка Обзор.
4. При щелчке на кнопке Обзор открывается диалоговое окно Обзор папок. Это стандартное средство для установления пути доступа к объекту.

   Нужную папку и файл разыскивают примерно так же, как на левой панели программы Проводник. Выбирают диск, на котором расположен искомый файл (в нашем случае это диск C:), затем разворачивают все вышележащие папки. Список файлов отображается в этом окне ниже имени соответствующей папки.

Разыскав нужный объект, его выделяют и щелкают на кнопке OK. Путь доступа к объекту автоматически заносится в командную строку мастера создания ярлыка.

5. Переход к очередному диалоговому окну мастера выполняют щелчком на командной кнопке Далее.
6. В очередном окне мастера вводят название ярлыка, например: Калькулятор. Если это последнее окно мастера, то кнопка Далее сменяется кнопкой Готово. Щелчок на этой кнопке приводит к выполнению заданной операции.

Программа Калькулятор является системной, и ее значок операционной системе хорошо известен. Поэтому Мастер создания ярлыка не задает ни одного вопроса по выбору значка и использует для ярлыка стандартный значок Калькулятора. Если создается ярлык для объекта, неизвестного системе, то мастер продолжает свою работу и предлагает выбрать какой-либо значок из коллекции значков, имеющихся в составе системы.

Приемы повышения эффективности в работе с файловой структурой

Приемы, которые здесь описаны, являются общесистемными. Они относятся не только к Проводнику, но и ко всем окнам папок и большинству окон приложений.

Использование буфера обмена для работы с объектами. Система *Windows* создает и обслуживает на компьютере невидимую для пользователя область памяти, называемую *буфером обмена*. Этой областью можно и нужно уметь пользоваться. В любой момент времени в ней можно хранить только один объект.

Принцип работы с буфером обмена очень прост:

1. Открываем папку-источник. Выделяем щелчком нужный объект.
2. *Копируем* или *забираем* объект в буфер. В первом случае объект остается в папке-источнике и может быть размножен. Во втором случае он удаляется из папки-источника, но может некоторое время храниться в буфере. Последняя операция называется также *вырезанием* объекта.
3. Открываем папку-приемник и помещаем в нее объект из буфера обмена.

Три указанные операции (Копировать, Вырезать и Вставить) можно выполнять разными способами. Классический прием состоит в использовании пункта Правка в строке меню, но более удобно пользоваться командными кнопками панели инструментов:

 — Копировать;

 — Вырезать;

 — Вставить.

Самый же эффективный способ работы с буфером обмена состоит в использовании комбинаций клавиш клавиатуры:

CTRL+C — копировать в буфер;

CTRL+X — вырезать в буфер;

CTRL+V — вставить из буфера.

Эти приемы работают во всех приложениях *Windows*, и их стоит запомнить. Через буфер обмена можно переносить фрагменты текстов из одного документа в другой, можно переносить иллюстрации, звукозаписи, видеофрагменты, файлы, папки и вообще любые объекты. Буфер обмена — мощное средство для работы с приложениями и документами в *Windows*.

В буфере обмена всегда может находиться 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.

В общем случае буфер обмена невидим для пользователя, и обычно необходимость просмотра его содержимого не возникает. Однако, если она все-таки возникнет, можно воспользоваться специальной служебной программой Папка обмена, которая входит в состав операционной системы и запускается командой Пуск ▸ Программы ▸ Стандартные ▸ Служебные ▸ Буфер обмена. Если на каком-то конкретном компьютере этой программы нет, это означает, что при установке операционной системы ее *компонент* не был установлен. Его можно доустановить.

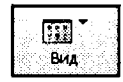
Групповое выделение объектов. Для многих операций (удаление, копирование, перемещение и т. п.) требуется выделить не один объект, а несколько. До сих пор мы использовали для выделения щелчок мыши, но он позволяет выделить только один объект. Для группового выделения при щелчке надо держать нажатой клавишу SHIFT или CTRL.

Если при щелчке держать нажатой клавишу CTRL, то выделение нового объекта не снимает выделение с объектов, выделенных ранее. Так можно выделить любую произвольную группу. Выделение при нажатой клавише CTRL действует как переключатель, то есть повторный щелчок на выделенном объекте снимает выделение.

Если выделяемые объекты расположены подряд, то можно воспользоваться клавишей SHIFT. В этом случае при нажатой клавише щелкают на первом выделяемом объекте группы и на последнем. Все промежуточные объекты выделяются автоматически. Для того чтобы использовать этот прием группового выделения, иногда бывает полезно предварительно упорядочить (отсортировать) объекты, представленные в окне.

Представление объектов. В системе *Windows* можно управлять тем, как представляются объекты в окнах папок или на правой панели программы Проводник. Существует четыре типа представления объектов:

- Плитка;
- Значки;
- Список;
- Таблица.



Выбор метода представления выполняют либо с помощью команд строки меню (пункт Вид), либо с помощью командной кнопки Вид на панели инструментов. Командная кнопка Вид действует как переключатель, автоматически изменяющий способ представления объектов в окне. Если же надо самостоятельно выбрать способ представления, то рядом с этой кнопкой есть раскрывающая кнопка, щелчок на которой раскрывает список возможных режимов.

Режим Плитка применяют в тех случаях, когда в папке находится небольшое количество уникальных объектов (например, программных файлов), каждый из которых важен. В этом режиме отображается не только имя и значок файла, но и некоторые другие его характеристики, зависящие от типа файла.

Режим Значки применяют, когда количество объектов в папке велико и в предыдущем режиме в окне помещается слишком мало значков.

Режим Список применяют в тех случаях, когда в окне присутствуют однотипные объекты, имеющие одинаковые значки. В этом случае содержание объекта характеризует не форма значка, а подпись под ним.

Режим Таблица применяют в тех случаях, когда важны дополнительные свойства объектов, такие как размер, дата создания и т. п. Этот режим интересен также тем, что предоставляет особые возможности по упорядочению объектов в окне.

Упорядочение объектов. Под упорядочением понимают прежде всего сортировку. В системе *Windows XP* существует четыре метода сортировки: Имя, Тип, Размер и Изменен. Метод упорядочения выбирают с помощью команды строки меню Вид ▸ Упорядочить значки.

Если используется метод сортировки Имя, объекты в окне располагаются в алфавитном порядке в соответствии с именами связанных с ними файлов. Когда при упорядочении во внимание принимается Тип, объекты располагаются тоже в алфавитном порядке, но в соответствии с расширениями имен связанных с ними файлов. Вариант Размер применяют перед проведением служебных операций. Например, перед очисткой жесткого диска с целью высвобождения рабочего пространства, удобно знать, какие объекты наиболее ресурсоемки.

Пункт Изменен используют при поиске файлов, изменявшихся в последние дни, или, наоборот, при поиске файлов, не изменявшихся очень долго. Есть вероятность, что документы, не востребованные в течение длительного периода, могут оказаться малонужными и их стоит отправить в архив.

Все методы сортировки работают в восходящем порядке. Файлы сортируются по именам от A до Z или от А до Я; по размерам — от 0 до 9; по датам — от ранних до более поздних. Но если объекты в окне отображаются в виде таблицы, то возможно проведение сортировки в нисходящем порядке. Особенность режима таблицы состоит в том, что каждый столбец имеет заголовок. Этот заголовок обладает свойствами командной кнопки. При первом щелчке на заголовке столбца происходит сортировка объектов по данному столбцу в восходящем порядке, при повторном щелчке — в нисходящем порядке.

5.4. Использование Главного меню

Структура Главного меню

Главное меню — один из основных системных элементов управления *Windows XP*. Оно отличается тем, что независимо от того, насколько Рабочий стол перегружен окнами запущенных процессов, доступ к Главному меню удобен всегда — оно открывается щелчком на кнопке Пуск. С помощью Главного меню можно запустить все программы, установленные под управлением операционной системы или зарегистрированные в ней, открыть последние документы, с которыми выполнялась работа, получить доступ ко всем средствам настройки операционной системы, а также доступ к поисковой и справочной системам *Windows XP*.

Главное меню — необходимый элемент управления для завершения работы с операционной системой. В нем имеется пункт Выключить компьютер, использование которого необходимо для корректного завершения работы с системой перед выключением питания.

В структуру Главного меню входят два раздела — *обязательный* и *произвольный*. Произвольный раздел расположен выше разделительной черты. Пункты этого раздела пользователь может создавать по собственному желанию. Иногда эти пункты образуются автоматически при установке некоторых приложений. Структура обязательного раздела Главного меню представлена в таблице 5.1.

5.5. Установка и удаление приложений Windows

В операционной системе *Windows XP* есть несколько способов установки приложений, но основным является метод, основанный на использовании значка Установка и удаление программ в папке Панель управления (Пуск ▸ Настройка ▸ Панель управления). Во всех случаях рекомендуется использовать именно это средство, поскольку прочие методы установки не гарантируют правильной регистрации приложений в реестре операционной системы.

Перед началом установки нового приложения следует закрыть все работающие программы и все открытые документы. В некоторых случаях необходимо закрывать и ряд фоновых процессов (их наличие может отображаться в виде значков панели индикации на правом краю Панели задач).

Особенности спецификации Windows

Приступая к установке приложений, необходимо знать особенности операционной системы, связанные с *совместным использованием ресурсов*, и помнить, что процедура установки непроверенных программных средств относится к категории потенциально опасных.

Принцип совместного использования ресурсов лежит в основе спецификации *Windows*, и в области программного обеспечения он приводит к тому, что разные приложения могут использовать общие программные ресурсы. Так, например, в большинстве приложений *Windows* можно встретить одинаковые элементы оформления и управления (окна, кнопки, раскрывающиеся списки, меню, флажки, пере-

Таблица 5.1. Структура Главного меню Windows XP

Пункт Главного меню	Назначение	Примечание
Программы	Открывает доступ к иерархической структуре, содержащей указатели для запуска приложений, установленных на компьютере. Для удобства пользования указатели объединяются в категории. Если категория имеет значок в виде треугольной стрелки, в ней имеются вложенные категории. Раскрытие вложенных категорий выполняется простым зависанием указателя мыши	Указатели, присутствующие в Главном меню, имеют статус ярлыков, а их категории – статус папок. Соответственно, указатели можно копировать и перемещать между категориями, перетаскивать на Рабочий стол и в окна папок. Это один из простейших способов создания ярлыка для недавно установленной программы.
Избранное	Открывает доступ к некоторым логическим папкам пользователя, в которых он может размещать наиболее часто используемые документы, ярлыки Web-документов и Web-узлов Интернета	Если с одним компьютером работают несколько пользователей, то каждый может иметь свою персональную группу избранных логических папок
Документы	Открывает доступ к ярлыкам последних пятнадцати документов, с которыми данный пользователь работал на компьютере	Физически эти ярлыки хранятся в скрытой папке \Recent
Настройка	Открывает доступ к основным средствам настройки Windows, в частности, к логической папке Панель управления. Служит также для доступа к папке Принтеры, через которую производится установка принтеров и настройка заданий на печать	При активной работе с компьютером приходится настолько часто использовать обращение к папке Панель управления, что целесообразно создать для нее ярлык на Рабочем столе, однако перетаскиванием из Главного меню это сделать не удается. Для создания ярлыка используйте значок Панель управления в окне Мой компьютер
Найти	Открывает доступ к средствам поиска, установленным на компьютере. Основным является средство Файлы и папки, с помощью которого производится поиск объектов в файловой структуре	При установке приложений, имеющих свои собственные средства поиска, может происходить автоматическое размещение дополнительных ярлыков в этой категории
Справка и поддержка	Пункт входа в справочную систему Windows XP	
Выполнить	Этот пункт открывает небольшое окно, имеющее командную строку для запуска приложений	Удобно использовать в тех случаях, когда необходимо в строке запуска приложения указать параметры запуска
Завершение сеанса ...	Если операционной системой зарегистрировано несколько пользователей одного компьютера, этот пункт позволяет завершить работу одного пользователя и передать компьютер другому	
Завершение работы	Корректное средство завершения работы с операционной системой. Открывает диалоговое окно Завершение работы в Windows, содержащее следующие пункты: • Ждущий режим; • Выключение; • Перезагрузка	Если закрыты все окна процессов, завершить работу с Windows можно комбинацией клавиш ALT+F4. Ждущий режим позволяет "заморозить", а затем восстановить состояние компьютера, хотя не все конфигурации оборудования это допускают и не все программы это хорошо переносят

ключатели и многое другое). Одинаковы и приемы управления ими, и методы их использования. С точки зрения приложений это означает, что их многие компоненты обрабатываются одним и тем же программным кодом. Поэтому в *Windows* принято выделять стереотипные программные фрагменты и группировать их в динамические библиотеки, к которым открыт доступ для разных программ (динамические библиотеки имеют расширение имени файла .DLL).

При установке новых приложений вместе с ними устанавливаются только те программные ресурсы, которые нужны для работы данного приложения, но отсутствуют на данном компьютере (то есть не зарегистрированы в его операционной системе). Поэтому для установки новых приложений очень важно, чтобы они проходили правильную регистрацию. Несмотря на то что в состав дистрибутивных комплектов большинства современных приложений входят специальные устанавливающие программы (Setup.exe), полагаться на то, что они правильно выполнят регистрацию, в общем случае не следует. Установку программ следует выполнять стандартными средствами. Этим обеспечивается надежная работа ранее установленных приложений и закладывается основа для корректной установки последующих приложений.

Стандартное средство установки приложений

Стандартное средство установки (и удаления) приложений *Windows* запускают командой Пуск ▸ Настройка ▸ Панель управления ▸ Установка и удаление программ. После двойного щелчка на указанном значке открывается диалоговое окно Свойства: Установка и удаление программ. Для установки произвольного программного обеспечения надо щелкнуть на значке Установка программ в левой части окна.

Установка приложения начинается с щелчка на кнопке CD или дискета. После этого запускается вспомогательная программа-мастер Установка программ с дискеты или компакт-диска. После щелчка на кнопке Далее мастер пытается автоматически запустить программу установки, найденную на съемном носителе.

Если ему это не удается, можно найти местоположение программы Setup.exe, которая входит в дистрибутивный комплект устанавливаемого приложения, с помощью кнопки Обзор. После этого надо щелкнуть на кнопке Готово.

После установки приложения нередко требуется перезагрузить компьютер. В системе *Windows XP* необходимость перезагрузки возникает реже, чем в предыдущих версиях операционных систем семейства *Windows*, но тоже может потребоваться. Это одна из причин, по которой до начала установки закрывают все открытые приложения и документы.

Необходимость перезагрузки связана с особенностью операционной системы. Некоторые операции выполняются удобно и безопасно только в момент завершения работы системы или на начальном этапе ее загрузки, когда большинство модулей системы еще не активированы.

Удаление приложений Windows

Удаление ранее установленных приложений *Windows* производится средствами того же диалогового окна Установка и удаление программ. Открыв его, следует щелк-

нуть на значке Изменение или удаление программ в левой части окна. Далее надо выбрать удаляемый объект. В зависимости от типа программы вы увидите две отдельные кнопки Изменить и Удалить или общую кнопку Заменить/Удалить. Щелчок на соответствующей кнопке запускает автоматическое средство удаления программы.

Удаление редко бывает полным. Скорее всего, какие-то компоненты останутся. Чаще всего остаются некоторые папки (как правило, пустые). Компоненты, не удаленные автоматически, следует удалить вручную. Рекомендуется удалять их в Корзину и наблюдать за компьютером в течение нескольких дней. Если после этого работоспособность прочих программ не нарушается, эти компоненты можно удалить и из Корзины.

5.6. Установка оборудования

В общем случае оборудование подключается к компьютеру дважды: аппаратно и программно. Под *аппаратным подключением* понимают физическое соединение с компьютером либо с помощью гнезд на материнской плате, либо с помощью внешних разъемов стандартных портов на задней стенке системного блока. Бывает и смешанное подключение, когда *интерфейсная плата* нового устройства вставляется в слот материнской платы и при этом создается новый (нестандартный) порт, разъем которого выходит на заднюю стенку. Таким способом подключают, как правило, устройства, требующие высокой скорости передачи данных, например сканеры или сетевые устройства.

Под *программным подключением* понимают установку программы-драйвера, являющейся посредником между операционной системой и устройством. При установке драйвера происходит выделение операционной системой части ресурсов новому устройству, а также регистрация устройства и его драйвера в реестре операционной системы.

Однако в общем правиле есть и исключения. Такие «стандартные» устройства, как жесткие диски, дисководы гибких дисков и клавиатура, не требуют драйверов, поскольку сведения о том, как с ними работать, уже имеются в базовой системе ввода-вывода (*BIOS*). Они должны распознаваться и работать еще до загрузки операционной системы. То же относится и к монитору, и к видеоадаптеру, но без драйверов они распознаются только как простейшие стандартные модели. Для того чтобы использовать все функциональные возможности конкретной модели, драйвер установить необходимо.

Несколько менее «стандартными» устройствами считаются мышь и дисковод *CD-ROM*. Они не всегда распознаются средствами *BIOS*, но после загрузки операционной системы *Windows XP* уже считаются стандартными устройствами и обслуживаются драйверами, имеющимися в ее составе; однако если речь идет о необычных моделях, особый драйвер для них может потребоваться.

Абсолютное большинство прочих устройств требуют наличия программного драйвера. При продаже аппаратного обеспечения общепринято прикладывать к устройству программные драйверы на компакт-диске. В отсутствие такой возможности

можно воспользоваться библиотекой драйверов, входящей в состав операционной системы. Если библиотека не поддерживает конкретную модель устройства, необходимый драйвер можно получить в Интернете на сервере фирмы, изготовившей оборудование, или на сервере компании *Microsoft*, где имеется коллекция драйверов устройств для операционных систем, выпускаемых этой компанией. Даже для старых и надежно работающих устройств рекомендуется периодически (два раза в год) посещать сервер изготовителя и получать обновленную версию драйвера. Своевременное обновление драйверов устройств повышает эффективность работы оборудования, улучшает совместимость с программным обеспечением и повышает общую надежность системы.

Средства программной установки оборудования

Базовое программное средство установки оборудования запускается двойным щелчком на значке Установка оборудования в окне папки Панель управления. С его помощью можно установить большую часть оборудования, хотя в общем правиле есть исключения.

Драйвер монитора можно установить в диалоговом окне свойств видеосистемы: Пуск ▸ Настройка ▸ Панель управления ▸ Экран ▸ Параметры ▸ Дополнительно ▸ Монитор ▸ Свойства ▸ Драйвер ▸ Обновить. Там же можно установить или заменить драйвер видеоадаптера: Пуск ▸ Настройка ▸ Панель управления ▸ Экран ▸ Параметры ▸ Дополнительно ▸ Адаптер ▸ Свойства ▸ Драйвер ▸ Обновить.

Специальные средства существуют для установки принтеров: Пуск ▸ Настройка ▸ Принтеры и факсы ▸ Установка принтера, а также для установки модемов Пуск ▸ Настройка ▸ Панель управления ▸ Телефон и модем.

Однако наиболее универсальным средством для большей части оборудования все-таки остается Мастер установки оборудования, который запускается двойным щелчком на значке Установка оборудования в окне папки Панель управления.

Порядок установки оборудования

Новое оборудование подключается при выключенном питании компьютера. Если устройство является самоустанавливающимся (соответствует спецификации *plug-and-play*), то после включения питания его наличие выявляется автоматически, и после сообщения Обнаружено неизвестное устройство операционная система приступает к подбору драйвера для него. В этот момент может потребоваться вставить дистрибутивный диск с операционной системой в дисковод *CD-ROM* или использовать компакт-диск с драйвером, полученным вместе с устройством. Иногда необходимы оба диска.

Если устройство не было опознано при запуске, надо воспользоваться Мастером установки оборудования. Мастер запускается командой Пуск ▸ Настройка ▸ Установка оборудования. На первом этапе он разыскивает устройства, соответствующие спецификации *plug-and-play*, и выдает список обнаруженных устройств. Если нужное устройство не входит в список, надо выбрать пункт Добавление нового устройства и щелкнуть на кнопке Далее. Мастер выполнит более тщательный поиск. Если нужное устройство вновь не удалось отыскать, остается возможность указать его

тип самостоятельно. После этого откроется диалоговое окно, в котором можно выбрать производителя и конкретную модель. При наличии нужной модели драйвер можно установить из базы данных *Windows* или с компакт-диска. Если абсолютного совпадения по модели достичь не удается, возможна только установка драйвера с диска, что выполняется после щелчка на кнопке Установить с диска.

По окончании процесса установки оборудования компьютер следует перезагрузить и выполнить проверку на наличие конфликтов. Для проверки наличия конфликтов используют значок Система в окне папки Панель управления или пункт Свойства контекстного меню значка Мой компьютер.

И в том и в другом случае открывается диалоговое окно Свойства: Система. На вкладке Оборудование необходимо щелкнуть на кнопке Диспетчер устройств. В окне Диспетчер устройств отображается список установленных устройств. Нераспознанные устройства в списке обозначены знаком «?», а конфликтующие — знаком «!». Простейший способ устранения конфликтов — удалить конфликтующие устройства с помощью кнопки Удалить и заново провести распознавание оборудования и установку драйверов обоих устройств. Во многих случаях это автоматически снимает проблемы. Более сложная технология устранения конфликтов предполагает назначение аппаратных ресурсов (номера прерывания, адреса порта, адреса канала прямого доступа к памяти) каждому из конфликтующих устройств вручную командой Свойства ▸ Ресурсы.

Практическое занятие

Упражнение 3.1. Отработка приемов управления с помощью мыши

15 мин

1. **Зависание.** Слева на Панели задач имеется кнопка Пуск. Это элемент управления *Windows*, называемый *командной кнопкой*. Наведите на нее указатель мыши и задержите на некоторое время — появится *всплывающая подсказка*: Начните работу с нажатия этой кнопки.

   Справа на Панели задач расположена *панель индикации*. На этой панели, в частности, расположен индикатор *системных часов*. Наведите на него указатель мыши и задержите на некоторое время — появится всплывающая подсказка с показаниями *системного календаря*.

2. **Щелчок.** Наведите указатель мыши на кнопку Пуск и щелкните левой кнопкой — над ней откроется *Главное меню Windows*. Меню — это один из элементов управления, представляющий собой список возможных команд. Команды, представленные в меню, выполняются щелчком на соответствующем пункте. Все команды, связанные с элементами управления, выполняются одним обычным щелчком.

   Однако у щелчка есть и другое назначение. Его применяют также для *выделения* объектов. Разыщите на Рабочем столе значок Мой компьютер и щелкните на нем. Значок и подпись под ним изменят цвет. Это произошло выделение объекта. Объекты выделяют, чтобы подготовить их к дальнейшим операциям.

Щелкните на другом объекте, например на значке Корзина. Выделение значка Мой компьютер снимется, а вместо него выделится значок Корзина. Если нужно снять выделение со всех объектов, для этого достаточно щелкнуть на свободном от объектов месте Рабочего стола.

3. **Двойной щелчок.** Двойной щелчок применяют для *использования* объектов. Например, двойной щелчок на значке, связанном с приложением, приводит к запуску этого приложения, а двойной щелчок на значке документа приводит к открытию данного документа в том приложении, в котором он был создан. При этом происходит одновременно и запуск этого приложения. Относительно документа оно считается *родительским.*

   В системе *Windows XP* с одним и тем же объектом можно выполнить много разных действий. Например, файл с музыкальной записью можно воспроизвести (причем в разных приложениях), его можно отредактировать, можно скопировать на другой носитель или удалить. Сколько бы действий ни было возможно с объектом, всегда существует одно *основное действие.* Оно и выполняется двойным щелчком.

   Выполните двойной щелчок на значке Мой компьютер, и на экране откроется одноименное окно Мой компьютер, в котором можно увидеть значки дисков, подключенных к компьютеру, значок Панели управления и другие значки.

   Если нужно закрыть окно, надо щелкнуть один раз на *закрывающей кнопке*, которая находится в правом верхнем углу окна. Закрывающая кнопка — это элемент управления, и для работы с ним достаточно одного щелчка.

4. **Щелчок правой кнопкой.** Щелкните правой кнопкой на значке Мой компьютер, и рядом с ним откроется элемент управления, который называется *контекстным меню.* У каждого объекта *Windows* свое контекстное меню. Состав его пунктов зависит от свойств объекта, на котором произошел щелчок. Для примера сравните содержание контекстного меню объектов Мой компьютер и Корзина, обращая внимание на их различия.

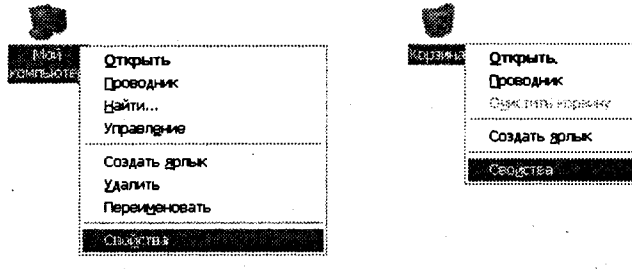


Рис. 5.4. Контекстные меню разных объектов имеют разный состав

Доступ к контекстному меню — основное назначение щелчка правой кнопкой. В работе с объектами *Windows* (особенно с незнакомыми) щелчок правой кнопкой используется очень часто.

Контекстное меню чрезвычайно важно для работы с объектами операционной системы. Выше мы говорили, что двойной щелчок позволяет выполнить только то действие над объектом, которое считается *основным*. В противоположность этому в контекстном меню приведены *все действия*, которые можно выполнить над данным объектом. Более того, во всех контекстных меню любых объектов имеется пункт Свойства. Он позволяет просматривать и изменять свойства объектов, то есть выполнять настройки программ, устройств и самой операционной системы.

5. **Перетаскивание.** Перетаскивание — очень мощный прием для работы с объектами операционной системы. Наведите указатель мыши на значок Мой компьютер. Нажмите левую кнопку и, не отпуская ее, переместите указатель — значок Мой компьютер переместится по поверхности Рабочего стола вместе с ним.

   Откройте окно Мой компьютер. Окно можно перетаскивать с одного места на другое, если «подцепить» его указателем мыши за строку заголовка. Так прием перетаскивания используют для оформления рабочей среды.

6. **Протягивание.** Откройте окно Мой компьютер. Наведите указатель мыши на одну из рамок окна и дождитесь, когда он изменит форму, превратившись в двунаправленную стрелку. После этого нажмите левую кнопку и переместите мышь. Окно изменит размер. Если навести указатель мыши на правый нижний угол окна и выполнить протягивание, то произойдет изменение размера сразу по двум координатам (по вертикали и горизонтали).

   Изменение формы объектов *Windows* — полезное, но не единственное использование протягивания. Нередко этот прием используют для *группового выделения* объектов. Наведите указатель мыши на поверхность Рабочего стола, нажмите кнопку мыши и протяните мышь вправо-вниз — за указателем потянется прямоугольный контур выделения. Все объекты, которые окажутся внутри этого контура, будут выделены одновременно.

7. **Специальное перетаскивание.** Наведите указатель мыши на значок Мой компьютер, нажмите правую кнопку мыши и, не отпуская ее, переместите мышь. Этот прием отличается от обычного перетаскивания только используемой кнопкой, но дает иной результат. При отпускании кнопки не происходит перемещение объекта, а вместо этого открывается так называемое *меню специального перетаскивания*. Содержимое этого меню зависит от перемещаемого объекта. Для большинства объектов в нем четыре пункта (Копировать, Переместить, Создать ярлыки и Отменить). Для таких *уникальных* объектов, как Мой компьютер или Корзина, в этом меню только два пункта: Создать ярлыки и Отменить.

Мы убедились, что, несмотря на то что стандартная мышь имеет только две кнопки, с их помощью можно реализовать весьма разнообразные приемы управления. Мы узнали наиболее характерные особенности этих приемов и их общепринятое назначение. В основе идеологии Windows лежит принцип, согласно которому базовые приемы управления операционной системой должны использоваться и при управлении ее приложениями. Знание общесистемных приемов пригодится в работе с любыми приложениями данной системы.

Упражнение 3.2. Изучение приемов работы с объектами

1. Откройте папку \Мои документы (Пуск ▸ Документы ▸ Мои документы).
2. Щелчком на раскрывающей кнопке разверните окно на полный экран.
3. В строке меню дайте команду Файл ▸ Создать ▸ Папку. Убедитесь в том, что в рабочей области окна появился значок папки с присоединенной надписью Новая папка.
4. Щелкните правой кнопкой мыши на свободной от значков рабочей области окна текущей папки. В открывшемся контекстном меню выберите команду Создать ▸ Папку. Убедитесь в том, что в пределах окна появился значок папки с надписью Новая папка (2).
5. Щелкните правой кнопкой мыши на значке Новая папка. В открывшемся контекстном меню выберите пункт Переименовать. Дайте папке содержательное имя, например Экспериментальная. Аналогично переименуйте папку Новая папка (2). Убедитесь в том, что операционная система не допускает существования в одной папке (\Мои документы) двух объектов с одинаковыми именами. Дайте второй папке имя Мои эксперименты.
6. Восстановите окно папки \Мои документы до нормального размера щелчком на восстанавливающей кнопке.
7. Откройте окно Мой компьютер. В нем откройте окно с содержимым жесткого диска (C:). Пользуясь полосами прокрутки, разыщите в нем папку \Windows и откройте ее двойным щелчком. Ознакомьтесь с текстом предупреждающего сообщения о том, что изменение содержания этой системной папки может быть потенциально опасным. Включите отображение содержимого папки щелчком на ссылке Отображать содержимое этой папки. В открывшемся содержимом разыщите значок папки \Temp и откройте ее (эта папка считается папкой временного хранения данных, и экспериментировать с ее содержимым можно без опасений). Перетаскиванием переместите папку \Экспериментальная из папки \Мои документы в папку C:\Windows\Temp. Специальным перетаскиванием переместите папку \Мои эксперименты в папку C:\Windows\Temp и по окончании перетаскивания выберите пункт Переместить в открывшемся контекстном меню.
8. Откройте окно C:\Windows\Temp. Щелчком выделите значок папки \Экспериментальная. При нажатой клавише CTRL щелчком выделите значок папки \Мои эксперименты. Убедитесь в том, что в рабочей области одновременно выделено два объекта (групповое выделение).
9. Заберите выделенные объекты в буфер обмена комбинацией клавиш CTRL+X. Убедитесь в том, что их значки исчезли в рабочей области папки.
10. Откройте окно папки \Мои документы. Вставьте в него объекты, находящиеся в буфере обмена (CTRL+V).
11. Выделите значки папок \Экспериментальная и \Мои эксперименты в папке \Мои документы. Щелкните правой кнопкой мыши и в открывшемся контекстном

меню выберите пункт Удалить. В открывшемся диалоговом окне подтвердите необходимость удаления объектов. Закройте окно папки \Мои документы.

12. Двойным щелчком на значке откройте окно Корзина. Убедитесь, что в нем находятся значки удаленных папок \Экспериментальная и \Мои эксперименты. Выделите оба значка. Щелкните правой кнопкой мыши и в открывшемся контекстном меню выберите пункт Восстановить. Закройте Корзину.
13. Откройте окно папки \Мои документы. Убедитесь в том, что в нем восстановились значки папок \Экспериментальная и \Мои эксперименты. Выделите оба значка. Удалите их с помощью клавиши DELETE при нажатой клавише SHIFT. В открывшемся диалоговом окне подтвердите необходимость удаления объектов. Закройте окно папки \Мои документы.
14. Откройте окно Корзины. Убедитесь в том, что объекты, удаленные при нажатой клавише SHIFT, не поступили в Корзину. Закройте Корзину.

Мы научились создавать новые папки с помощью строки меню и контекстного меню, научились давать папкам осмысленные имена, познакомились с тремя приемами копирования и перемещения объектов между окнами папок (перетаскиванием, специальным перетаскиванием и с использованием буфера обмена). Мы освоили приемы группового выделения объектов, удаления объектов в Корзину и окончательного удаления, минуя Корзину.

Упражнение 3.3. Работа с файловой структурой в программе Проводник


30 мин

1. Включите персональный компьютер, дождитесь окончания загрузки операционной системы.
2. Запустите программу Проводник с помощью Главного меню (Пуск ▸ Программы ▸ Проводник). Обратите внимание на то, какая папка открыта на левой панели Проводника в момент запуска. Это должна быть папка \Мои документы.
3. На правой панели Проводника создайте новую папку \Экспериментальная.
4. На левой панели разверните папку \Мои документы одним щелчком на значке узла «+». Обратите внимание на то, что *раскрытие* и *разворачивание* папок на левой панели — это разные операции. Убедитесь в том, что на левой панели в папке \Мои документы образовалась вложенная папка \Экспериментальная.
5. Откройте папку \Экспериментальная на левой панели Проводника. На правой панели не должно отображаться никакое содержимое, поскольку эта папка пуста.
6. Создайте на правой панели Проводника новую папку \Мои эксперименты внутри папки \Экспериментальная. На левой панели убедитесь в том, что рядом со значком папки \Экспериментальная образовался узел «+», свидетельствующий о том, что папка имеет вложенные папки. Разверните узел и рассмотрите образовавшуюся структуру на левой панели Проводника.
7. На левой панели Проводника разыщите папку \Windows и разверните ее.

8. На левой панели Проводника внутри папки \Windows разыщите папку для временного хранения объектов — \Temp, но не раскрывайте ее.
9. Методом перетаскивания переместите папку \Экспериментальная с правой панели Проводника на левую — в папку C:\Windows\Temp. Эту операцию надо выполнять аккуратно. Чтобы «попадание» было точным, следите за цветом надписи папки-приемника. При точном наведении надпись меняет цвет — в этот момент можно отпускать кнопку мыши при перетаскивании. Еще труднее правильно «попасть в приемник» при перетаскивании групп выделенных объектов. Метод контроля тот же — по выделению надписи.
10. На левой панели Проводника откройте папку C:\Windows\Temp. На правой панели убедитесь в наличии в ней папки \Экспериментальная.
11. Разыщите на левой панели Корзину и перетащите папку \Экспериментальная на ее значок. Раскройте Корзину и проверьте наличие в ней только что удаленной папки. Закройте окно программы Проводник.

Мы научились выполнять навигацию с помощью левой панели программы Проводник и изучили приемы копирования и перемещения объектов методом перетаскивания между панелями. Те, кто считает, что с левой панелью Проводника работать не очень удобно, могут исполнять все операции, пользуясь только правой панелью. При этом используют следующие свойства Проводника:

- возможность копирования и перемещения объектов через буфер обмена;
- программу Проводник можно запустить несколько раз — соответственно, на Рабочем столе можно иметь несколько правых панелей, между которыми удобно выполняются все операции обмена.

Исследовательская работа

Задание 3.1. Исследование методов запуска программы Проводник

В операционной системе *Windows XP* большинство операций можно выполнить многими разными способами. На примере программы Проводник мы исследуем различные приемы запуска программ.

1. Щелкните правой кнопкой мыши на кнопке Пуск и в открывшемся контекстном меню используйте пункт Проводник. Обратите внимание на то, какая папка открыта на левой панели в момент запуска.
2. Щелкните правой кнопкой мыши на значке Мой Компьютер и в открывшемся контекстном меню используйте пункт Проводник. Обратите внимание на то, какая папка открыта на левой панели в момент запуска.
3. Проверьте контекстные меню всех значков, открытых на Рабочем столе. Установите, для каких объектов контекстное меню имеет средства запуска Проводника, и выясните, какая папка открывается на левой панели в момент запуска.
4. Выполните запуск Проводника через пункт Программы Главного меню.
5. Выполните запуск Проводника через пункт Выполнить Главного меню.

6. Выполните запуск Проводника через ярлык папки \Мои документы (Пуск ▸ Документы ▸ Мои документы ▸ *щелчок правой кнопкой мыши* ▸ Проводник).
7. Выполните запуск Проводника с Рабочего стола (предварительно на Рабочем столе следует создать ярлык Проводника).
8. Выполните запуск Проводника с Панели быстрого запуска (предварительно на этой панели следует создать ярлык Проводника).
9. Заполните отчетную таблицу по образцу:

Метод запуска Проводника	**Используемый элемент управления**	**Папка открытия**
Через контекстное меню кнопки Пуск	Кнопка Пуск	\Главное меню

В этой и следующей главе рассматриваются понятия, методы и приемы, относящиеся к созданию текстовых документов с помощью персонального компьютера. Условно (из чисто методических соображений) мы выделим две группы создаваемых документов — *простые* и *комплексные*. Первые представляют собой форматированный текст, а вторые содержат кроме текста объекты иной природы (чертежи, рисунки, формулы, таблицы, объекты мультимедиа и прочие).

10.1. Общие сведения о текстовом процессоре Microsoft Word

Общее название программных средств, предназначенных для создания, редактирования и форматирования простых и комплексных текстовых документов, — *текстовые процессоры*. В настоящее время в России наибольшее распространение имеет текстовый процессор *Microsoft Word*. Это связано, прежде всего, с тем, что его создатели относительно давно предусмотрели *локализацию* программы в России путем включения в нее средств поддержки работы с документами, исполненными на русском языке.

Основные версии текстового процессора Microsoft Word

Первоначальные версии текстового процессора *Microsoft Word* относятся к восьмидесятым годам и, соответственно, к операционной системе *MS-DOS*. Последней версией процессора для неграфической операционной среды была версия *Microsoft Word 5.0*. Она позволяла создавать, редактировать и распечатывать форматированные текстовые документы.

Поскольку операционная система *MS-DOS* не является графической, данная версия программы не могла соблюдать принятый ныне принцип соответствия экранного изображения печатному (принцип *WYSIWYG*) и операции форматирования документа выполнялись в известной степени «вслепую». Однако возможность просмотра документа в «натуральном» виде все-таки была. Она реализовывалась

специальным режимом *предварительного просмотра* (*preview*), который сохранился и в современных версиях программы, хотя и не имеет уже решающего значения.

Основным преимуществом текстового процессора *Word 5.0*, отличавшим эту программу от конкурентных продуктов, была возможность встраивания в текст графических объектов, правда, без взаимодействия текста и графики (*обтекания графических изображений текстом*). Сегодня текстовым процессором *Word 5.0* еще иногда пользуются при работе на устаревшем оборудовании (*IBM PC AT/286*).

Принцип *WYSIWYG* впервые был реализован в следующей версии программы, которая называлась *Microsoft Word for Windows* (*Word 6.0*). Благодаря этому принципу значительно упростились и стали наглядными приемы форматирования документов. Будучи приложением *Windows* 3.1, программа получила возможность использовать системный буфер обмена, а пользователи получили мощное и удобное средство для создания комплексных документов.

Следующая версия программы называлась *Microsoft Word 95* (*Word 7.0*). Она была ориентирована на графическую операционную систему *Windows 95*. Основным достижением этой версии стало то, что после нее текстовый процессор уже не рассматривается только как отдельное приложение. В состав мощного офисного пакета *Microsoft Office* входит несколько приложений (с каждой новой версией пакета этот состав расширяется), и на процессор *Microsoft Word* возлагаются дополнительные функции интеграции прочих приложений. Он занимает центральное положение в системе и позволяет организовать эффективный обмен данными между составляющими приложениями, что позволило в значительной степени автоматизировать разработку офисных документов разной содержательности и сложности.

Еще одним важным нововведением седьмой версии стало управление взаимодействием текста со встроенными объектами, что значительно расширило набор возможностей при форматировании документов. А особенный успех этой версии программы в России (она очень широко используется и сегодня) завоевали встроенные средства поддержки русского языка (автоматическая проверка орфографии и грамматики).

Восьмая версия программы *Microsoft Word 97* (*Word 8.0*), вошедшая в состав пакета *Microsoft Office 97*, внесла относительно мало практически полезных изменений для повседневной офисной работы. Так, например, ее жесткая ориентация на использование шрифтов *UNICODE* затруднила обмен данными с большинством приложений, выпущенных «третьими» фирмами, и создала пользователям проблемы при печати материалов на большинстве печатающих устройств. Дополнительные средства оформления текстовых документов, представленные в этой версии, имели практическое значение только при разработке электронных (экранных) документов. Возможность сохранения документов в «электронных» форматах *HTML* и *PDF*, рассчитанная на публикацию документов в Интернете, осталась проработанной не до конца и не вошла в практику *Web*-дизайнеров.

Начиная с этой версии текстовый процессор *Microsoft Word* можно рассматривать как *средство автоматизации авторской деятельности* (*authoring system*). При использовании этой программы следует четко определять целевой объект — документ *электронный* или *печатный*. Для разных типов документов используют раз-

ные средства, приемы и методы. Применение неадекватных средств значительно усложняет последующие этапы работы с документом. В итоге в качестве средства разработки электронных документов *Word 97* не заменил *Web*-редакторы, а в качестве средства создания печатных документов внес ряд неудобств по сравнению с предыдущей версией *Word 95*.

Очередной стала версия текстового процессора *Microsoft Word 2000* (*Word 9.0*), входящая в состав пакета *Microsoft Office 2000*. В ней устранены основные недостатки предыдущей версии, заметно улучшена система управления и введены мощные средства поддержки сетевых режимов работы. Предполагается, что основным стилем производительной работы с текстовым процессором *Word* 2000 должна стать совместная деятельность рабочих групп над общими проектами в рамках корпоративных сетей.

Последней (ко времени подготовки данного пособия) является версия текстового процессора *Microsoft Word XP* (*Word 10.0*). Они входит в состав пакета *Microsoft Office XP*. В ней заметно расширены средства работы со стилями и шаблонами, введены механизмы, позволяющие автоматически обеспечить единство оформления документа.

Рабочее окно процессора Microsoft Word 2000

Рабочее окно процессора *Microsoft Word XP* представлено на рис. 10.1. Его основные элементы управления: строка меню, панель инструментов, рабочее поле и строка состояния, включающая индикаторы. Начиная с процессора *Microsoft Word 95*, панель инструментов является настраиваемой.

Режимы отображения документов

Начиная с шестой версии, текстовый процессор *Microsoft Word* поддерживает несколько режимов представления документов.

В *обычном режиме* представляется только содержательная часть документа без реквизитных элементов оформления, относящихся не к тексту, а к печатным страницам (колонтитулы, колонцифры, подстраничные сноски и т. п.). Этот режим удобен на ранних этапах разработки документа (ввод текста, редактирование, рецензирование), а также во всех случаях, когда содержательная часть документа имеет более высокое значение, чем внешнее представление. В этом режиме операции с объемными документами проходят быстрее, что важно при работе на малопроизводительных компьютерах.

В *режиме Web-документа* экранное представление не совпадает с печатным. Это отступление от принципа *WYSIWYG*, но оно характерно для электронных публикаций в *World Wide Web*, поскольку заранее не известно, каким средством просмотра и на каком оборудовании будет отображаться документ. Понятие печатной страницы для электронных документов не имеет смысла, поэтому назначенные параметры страницы не учитываются, а форматирование документа на экране является *относительным*. В этом режиме разрабатывают электронные публикации.

В *режиме разметки* экранное представление документа полностью соответствует печатному, вплоть до назначенных параметров печатной страницы. Этот режим

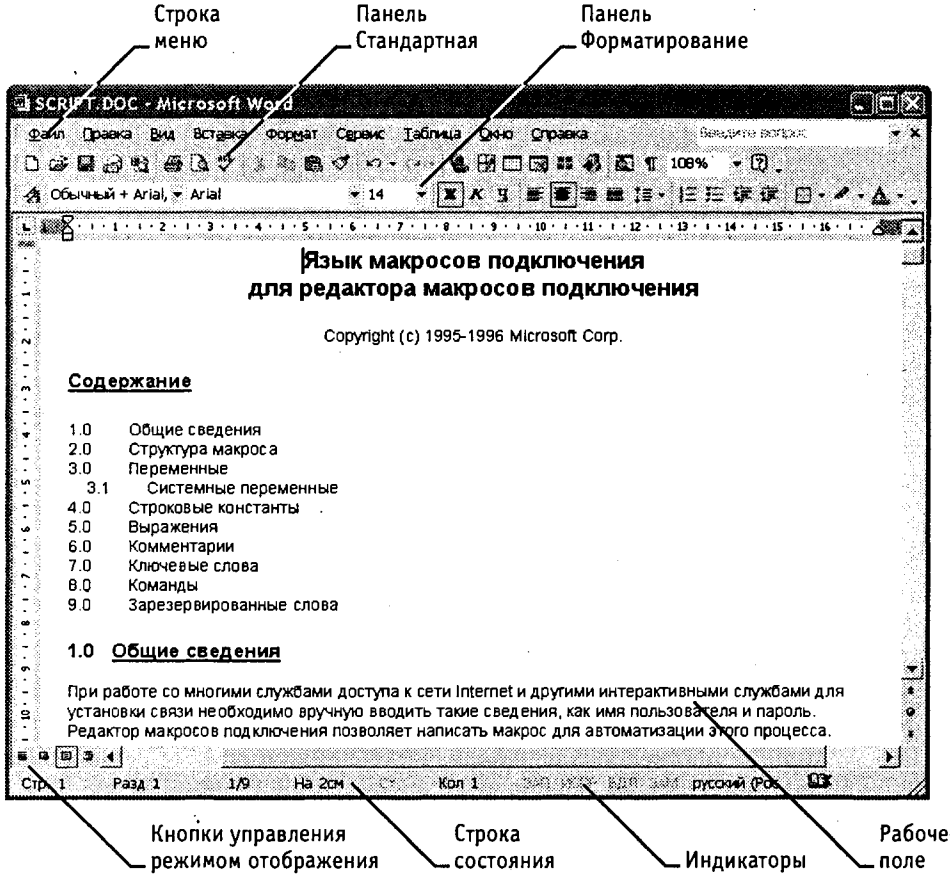


Рис. 10.1. Рабочее окно программы Word XP

удобен для большинства работ, связанных с форматированием текста, предназначенного для печати.

В *режиме структуры* можно отобразить только заголовки документа. Режим полезен в тех случаях, когда разработку документа начинают с создания плана содержания. Если предполагаемый размер документа превышает 5–7 печатных страниц, следует начинать работу именно с создания первичного плана. Режим структуры отличается тем, что при его включении автоматически открывается вспомогательная панель инструментов Структура, элементы управления которой позволяют править структуру документа.

Выбор одного из четырех указанных режимов представления документа выполняют с помощью командных кнопок, расположенных в левом нижнем углу окна приложения, или командами меню Вид.

Через меню Вид доступно также специальное представление (пятый режим) Схема документа, при котором окно приложения имеет две рабочие панели. На левой панели

представляется структура документа, а на правой — сам документ. Этот режим, сочетающий достоинства режима разметки и режима структуры, полезен при навигации по объемному документу — его удобно использовать не при создании, а при просмотре документов сложной структуры.

Через меню Файл доступны еще два режима представления документа, используемые для предварительного просмотра. Для электронных документов используют команду Файл ▸ Предварительный просмотр Web-страницы, а для печатных документов — Файл ▸ Предварительный просмотр. В первом случае созданный документ отображается как *Web*-страница в окне браузера, зарегистрированного операционной системой в качестве принятого по умолчанию (желательно, чтобы это был браузер *Microsoft Internet Explorer* 6.0). Во втором случае документ представляется в специальном окне.

Приемы работы с командами строки меню

Как и в большинстве других приложений, корректно соблюдающих идеологию *Windows*, строка меню текстового процессора *Microsoft Word XP* как элемент управления отличается тем, что обеспечивает доступ ко всем функциональным возможностям программы. Не всегда этот доступ самый удобный, во многих случаях другие элементы управления использовать проще, но строка меню удовлетворяет *принципу функциональной полноты*.

Меню, открывающиеся из строки меню, обладают свойством *функциональной автонастройки*. Расширенные возможности приложения не могли не отразиться в изобилии элементов управления, открываемых через строку меню. В нем не всегда удобно ориентироваться. Поэтому пункты строки меню открываются в два приема. На первом этапе открывают *сокращенное меню*, и, если необходимого элемента управления в нем нет, открывают *расширенное меню* наведением указателя мыши на *пункт раскрытия*. Использованные пункты расширенной части меню далее открываются в составе сокращенного меню (рис. 10.2).

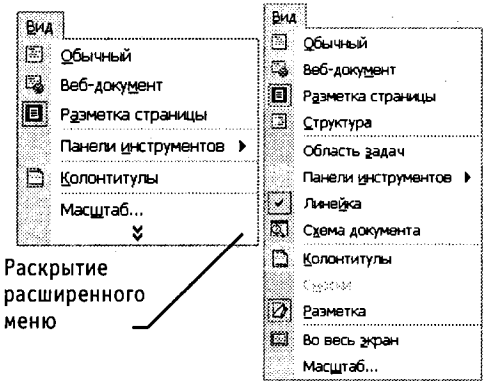


Рис. 10.2. Команды меню Вид, сокращенный и расширенный вариант

Панели инструментов Microsoft Word XP

Начиная с седьмой версии, программа *Microsoft Word* поддерживает возможность самостоятельной настройки панелей инструментов. Настройку выполняет пользователь путем подключения функциональных панелей, необходимых ему по роду деятельности (Вид ▸ Панели инструментов). Расширение общей панели инструментов сопровождается некоторым уменьшением площади рабочего окна документа. Перемещение функциональных панелей производят методом перетаскивания за рубчик, расположенный на левом краю панели.

В последних версиях текстового процессора панели инструментов не только допускают настройку, но и обладают контекстной чувствительностью. Так, при выделении в поле документа какого-либо объекта, автоматически открывается панель инструментов, предназначенная для его редактирования. Назначение панелей инструментов приведено в таблице 10.1.

Таблица 10.1. Панели инструментов программы Word XP

Панель инструментов	Состав, назначение	Примечание
Стандартная	Элементы управления файловыми операциями, редактированием, экранным отображением	Устанавливается по умолчанию
Форматирование	Элементы управления форматированием документа	Устанавливается по умолчанию
Visual Basic	Доступ к средствам создания и редактирования макросов и Web-сценариев, а также к настройке средств обеспечения безопасности при запуске макросов	Макросы служат для автоматизации типовых операций. Web-сценарии обеспечивают динамичный характер просмотра Web-страниц
Word-Art	Элементы управления для создания художественных заголовков	
Автотекст	Средство быстрого доступа к настройке функции автотекста	Одновременно предоставляет быстрый доступ к средствам настройки функций автозамены и автоформата
База данных	Элементы управления, характерные для работы с базами данных (сортировка, поиск, управление структурой таблиц и прочее)	В качестве базы данных могут выступать как таблицы Access, так и собственные таблицы Word
Веб-компоненты	Комплект готовых компонентов для создания элементов управления Web-страницы или электронной формы	Применяются для создания обратной связи с потребителем документа (опросные листы, анкеты, бланки заказов и заявок и прочее)
Веб-узел	Элементы управления для навигации в Web-структурах данных	В качестве Web-структур могут выступать World Wide Web, корпоративные сети intranet, системы Web-документов локального компьютера
Настройка изображения	Элементы управления для основных функций настройки растровых изображений	Позволяют настраивать яркость, контрастность, размер, рамку, режимы обтекания текстом и прочие параметры выделенного растрового объекта
Рамки	Элементы управления для создания фреймов (не путать с рамками, создаваемыми с помощью панели инструментов Таблицы и границы)	Текстовый процессор Word XP поддерживает два типа фреймов. Фреймы в электронных документах представляют собой особые прямоугольные области, предназначенные для вывода нескольких Web-документов в рамках одной Web-страницы. Фреймы в печатных документах представляют особые области печатной страницы для вывода специальной информации, например колонтитулов

Таблица 10.1. Панели инструментов программы Word 2000 (окончание)

Панель инструментов	Состав, назначение	Примечание
Рецензирование	Элементы управления для проведения редактирования и комментирования документов без искажения исходного текста	Измененные данные сохраняются в том же документе на правах новых версий. Автор исходного текста имеет возможность просмотреть замечания и предлагаемые изменения, после чего принять их или отвергнуть
Рисование	Элементы управления и инструменты для выполнения простейших чертежно-графических работ	Графические объекты, создаваемые инструментами данной панели, имеют характер векторных объектов
Слияние	Инструменты для работы с документами слияния, содержащими постоянную и переменную части	Используется при использовании программы Word XP, например, для массовой подготовки писем аналогичного содержания
Статистика	Позволяет получить информацию об объеме документа	Сведения о числе знаков, слов, строк, абзацев, страниц
Структура	Инструменты для работы с логической структурой документа	Позволяет управлять заголовками и порядком следования логических частей текста. Активно используется при работе с документом в режиме структуры
Таблицы и границы	Элементы управления для создания таблиц и оформления текстовых блоков рамками	Дополнительно предоставляет средства для сортировки данных и проведения итоговых расчетов в таблицах (функция Автосумма)
Формы	Элементы управления для разработки стандартных форм	Программа Word XP позволяет создавать три типа форм: Web-формы, являющиеся объектами Web-страниц; формы Word, распространяемые и заполняемые как электронный документ; печатные формы
Элементы управления	Набор готовых компонентов ActiveX для создания элементов управления Web-страниц и Web-форм	Средства данной панели инструментов позволяют не только использовать около 150 готовых компонентов, но и проводить установку и регистрацию дополнительных компонентов ActiveX

Кроме того, в программе *Word XP* роль специальной контекстно-зависимой информационно-инструментальной панели играет область задач. Эта область открывается на правах панели инструментов и обычно располагается у правого края окна программы. Область задач может использоваться для выполнения разных функций, в зависимости от выбранного режима. Выбор режима осуществляется из меню, открывающегося при щелчке на треугольной кнопке в верхней строке Область задач. Кроме того, существуют специальные команды для открытия области задач в нужном режиме (например, Правка ▸ Буфер обмена Office или Формат ▸ Показать форматирование). Режимы работы области задач описаны в таблице 10.2.

Таблица 10.2. Режимы области задач

Режим	Команда	Содержание Области задач	Назначение
Создание документа	Файл ▸ Создать	Список недавно открывавшихся документов, команды создания новых документов	Открытие существующих и создание новых документов
Буфер обмена	Правка ▸ Буфер обмена Office	Содержание буфера обмена Office (до 24 объектов)	Выбор объектов для вставки в документ
Поиск	Файл ▸ Найти	Команды для поиска и информация о результатах поиска	Поиск текста в форматированных файлах
Вставка картинки	Вставка ▸ Рисунок ▸ Картинки	Команды для поиска клипартов и информация о результатах поиска	Выбор клипартов и других графических изображений для вставки в документ
Стили и форматирование	Формат ▸ Стили и форматирование	Сведения о стилях и форматировании текста и средства для их изменения	Выбор и создание стилей на основе существующего оформления текста
Показать форматирование	Формат ▸ Показать форматирование	Сведения о характеристиках форматирования в месте расположения курсора	Информация о форматировании, сравнение форматов разных фрагментов
Слияние	Сервис ▸ Письма и рассылки ▸ Мастер слияния	Этапы создания документа слияния	Создание документов слияния (например, писем), содержащих постоянную и переменную части
Перевод	Сервис ▸ Язык ▸ Перевод	Исходный текст и результат перевода	Перевод отдельных слов и коротких фраз

Основные принципы практической работы с текстовым процессором Microsoft Word

Основные принципы практической работы зависят от используемой версии программы. Базовый принцип здесь состоит в том, что чем больше возможностей имеет программа, тем строже надо подходить к выбору тех функций, которыми можно пользоваться в каждом конкретном случае. Удобен подход, когда набор допустимых средств оформления и форматирования документа определяет его заказчик.

Заказчик есть у каждого документа. Даже если документ готовится для личного употребления, условным заказчиком является сам автор. Заказчиком можно считать и исполнителя, которому передается документ для последующих операций, например, для рецензирования или вывода на печать. К категории «заказчиков» относятся и предполагаемые клиенты, для которых данный документ разрабатывается. При этом возникает ряд вопросов, которые надо решить до начала работы с документом.

К какому типу относится документ? Современные текстовые процессоры позволяют создавать документы трех типов. Во-первых, это *печатные документы*, кото-

рые создаются и распечатываются на одном рабочем месте или в одной рабочей группе. Дальнейшее движение документа происходит только в бумажной форме. Состав допустимых средств оформления в данном случае определяется только техническими возможностями печатающего устройства.

Второй тип — *электронные документы в формате текстового процессора*, например *Microsoft Word*. Такие документы передаются заказчику в виде файлов. Электронный документ, как правило, не является окончательным. В большинстве случаев заказчик может его дорабатывать, редактировать, форматировать, распечатывать или использовать его компоненты для подготовки своих документов (книг, журналов, сборников статей и т. п.). Набор разрешенных средств в данном случае, как правило, минимален и определяется заказчиком.

Третий тип — *Web-документы*. Предполагается, что в этом качестве они останутся навсегда, и их преобразование в печатные документы не планируется. В *Web*-документах большую роль играет управление цветом. Для этой категории документов наиболее широк выбор средств форматирования и оформления.

Кто является заказчиком документа? Самый типичный случай — когда заказчиком документа является работодатель, то есть администрация предприятия или учреждения. Надо выяснить правила оформления документов, принятые в данной организации, и строго их придерживаться. Если существуют готовые шаблоны, их надо использовать, а если их нет, то разработать свои и согласовать с руководством.

Самый простой случай — когда заказчика нет, и автор делает документ для себя. Он может использовать любые средства, которые ему подскажет фантазия и которые поддерживаются его устройствами вывода (экран для *Web*-документов или принтер для печатных документов).

Самый трудный случай — когда заказчик внешний, особенно если он не вполне определен. В этом случае исполнители часто путают понятия *представление* документа и *предоставление* документа. Для *представления* документа они стремятся использовать все средства форматирования, которые наилучшим образом подчеркивают достоинства документа. При *предоставлении* документа ситуация обратная. Здесь не автор, а заказчик определяет форму и средства форматирования и оформления. Использование этих средств в данном случае имеет *разрешительный характер*. Если же требования заказчика еще не известны, следует предполагать, что нежелательны большинство средств форматирования документов, передаваемых для дальнейшей обработки. В частности, необходимо:

- ограничить используемые наборы шрифтов только теми, которые входят в состав операционной системы (не более двух наборов: один — для основного текста, другой — для заголовков и вспомогательного текста);
- минимизировать использование средств форматирования абзацев: отказаться от выравнивания по ширине и от переноса слов, ограничить число используемых шрифтовых начертаний (не более двух: основного и дополнительного);
- отключить все автоматические средства форматирования: расстановку колонтитулов, нумерацию страниц, маркировку и нумерацию списков и прочие;

- не использовать встроенные средства текстового процессора для создания встроенных объектов (художественные заголовки, векторные рисунки, рамки и прочие) — все объекты должны создаваться специальными программами, храниться в отдельных файлах, вставляться в текст документа методом связывания и прилагаться к файлу документа;
- исключить использование приемов взаимодействия встроенных объектов с текстом;
- сохранять готовые документы в простейших форматах, несущих минимум информации о форматировании (для документов *Microsoft Word* таковыми являются форматы Только текст, Текст в формате RTF или Word 6.0/95);
- в каждом случае отступления от этих правил, например при необходимости использовать формулы, таблицы и специальные символы, согласовывать свои действия с заказчиком.

Эти требования к документам, предоставляемым для дальнейшей технологической обработки, связаны с тем, что большинство средств оформления и форматирования текстового процессора являются «вещью в себе». Достоинства этих средств проявляются только при выводе окончательного документа средствами того же самого процессора, будь то вывод на печать, просмотр на экране или публикация в *Web*-структуре. При обработке данных, содержащихся в документе, другими программными средствами преимущества форматирования и оформления могут оборачиваться тяжкими проблемами.

Первичная настройка текстового процессора Microsoft Word

Приступая к первому знакомству с текстовым процессором *Microsoft Word*, следует выполнить ряд первичных настроек. Некоторые средства автоматизации, имеющиеся в программе, могут отвлекать начинающего пользователя от главной задачи — освоения основных приемов. В ряде случаев из-за работы автоматических средств результаты операций получаются неожиданными — это препятствует установлению обратной связи и эффективному усвоению практических приемов.

Комплекс настроек, рекомендуемых перед началом освоения текстового процессора, приведен в упражнении 10.1.

10.2. Приемы работы с текстами в процессоре Microsoft Word

К базовым приемам работы с текстами в текстовом процессоре *Microsoft Word* относятся следующие:

- создание документа;
- ввод текста;
- редактирование текста;
- рецензирование текста;
- форматирование текста;
- сохранение документа;
- печать документа.

Создание документа

В текстовом процессоре *Word XP* принято использовать два метода создания нового документа: на основе готового шаблона или на основе существующего документа. Второй метод проще, но первый методически более корректен.

Создание документа на основе имеющегося документа. Этот метод потенциально опасен, и потому его использование категорически не рекомендуется! Тем не менее, им очень широко пользуются, и мы его рассматриваем, чтобы предупредить о возможных опасностях и обратить внимание на правильный порядок действий.

При создании документа на основе существующего документа:

- открывают готовый документ (Файл ▸ Открыть);
- сохраняют его под новым именем (Файл ▸ Сохранить как);
- выделяют в нем все содержимое (Правка ▸ Выделить все);
- удаляют его нажатием клавиши DELETE;
- в результате получают пустой документ, имеющий собственное имя и сохраняющий все настройки, ранее принятые для исходного документа.

Этот метод характерен для начинающих пользователей, не умеющих создавать шаблоны и пользоваться ими. Получив задание у руководителя, они запрашивают образец и приступают к его правке. Метод интуитивно прост, но чреват весьма неприятными ошибками. Если забыть сохранить новый файл под другим именем, можно легко уничтожить ценный документ, даже не успев создать новый. Кроме того, при небрежной правке содержание документа, взятого за основу, может переходить в новый документ. Для рабочих мест, на которых создаются десятки документов в сутки, этот метод весьма опасен.

Создание документа на основе шаблона. Шаблоны — это те же образцы документов, но защищенные от досадных неприятностей. Создание документа на основе готового шаблона выполняется следующим образом.

1. Команда Файл ▸ Создать открывает Область задач в режиме создания документа. Щелкните на этой панели на ссылке Общие шаблоны — откроется диалоговое окно Шаблоны. Надо включить переключатель Создать документ и выбрать подходящий шаблон. Если никаких предпочтений нет, следует выбрать шаблон Новый документ на вкладке Общие. Созданный документ приобретает имя Документ1, принятое по умолчанию. Его целесообразно сразу же сохранить под «правильным» именем, выбрав для него соответствующую папку и дав команду Файл ▸ Сохранить как.
2. Диалоговое окно Сохранение документа в текстовом процессоре *Microsoft Word XP*, представленное на рис. 10.3, практически не отличается от аналогичного окна ранее рассмотренных нами стандартных приложений. Оно обычно предполагает сохранение файла в папку \Мои документы, но обеспечивает быстрый доступ и к некоторым иным папкам.
3. В левой части окна Сохранение имеется пять кнопок, позволяющих быстро выбрать место для сохранения файла.

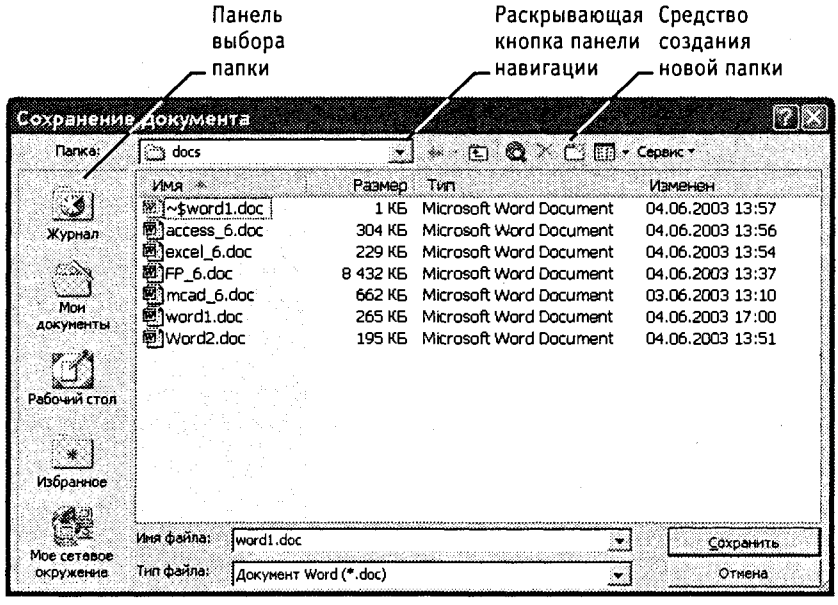


Рис. 10.3. Диалоговое окно Сохранение документа

Журнал — логическая папка. Если нужно сохранить документ в одну из папок, которой пользовались в последнее время, это очень удобное средство доступа.

Мои документы — традиционная папка для хранения авторских документов в операционных системах семейства *Windows*.

Рабочий стол — не слишком удобное место для хранения документов, поскольку его принято содержать «в чистоте». Есть два случая, когда Рабочий стол используют для хранения документов:

- если документ временный и после просмотра будет удален в Корзину;
- если документом предполагается пользоваться особенно часто (например, это список номеров телефонов коллег по работе).

Избранное — особая логическая папка пользователя, предназначенная для хранения ярлыков *Web*-страниц. Ее нецелесообразно использовать для сохранения текстовых документов, но для открытия документов она может использоваться активно.

Мое сетевое окружение — этот значок обеспечивает быстрый доступ к сохранению документа не на своем компьютере, а в локальной сети, например на файловом сервере. При этом требуются дополнительные операции по навигации, связанные с выбором конкретной папки на конкретном сетевом компьютере.

При необходимости сохранить документ в произвольную папку, не представленную в данном списке, следует выполнить навигацию по файловой структуре с использованием раскрывающей кнопки на правом краю поля Папка.

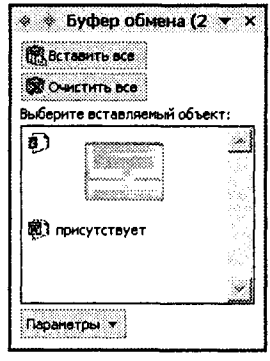
Специальные средства ввода текста

Технология ввода текста и переключения языковых раскладок клавиатуры, применение регистровых клавиш и буфера обмена *Windows* были представлены выше при описании стандартного приложения Блокнот. В данном разделе мы остановимся на особенностях текстового процессора *Microsoft Word XP*, позволяющих автоматизировать ввод текста.

Средства отмены и возврата действий. Все операции ввода, редактирования и форматирования текста протоколируются текстовым процессором, и потому необходимое количество последних действий можно отменить. Последнее действие отменяют комбинацией клавиш CTRL+Z. Эта команда имеет кумулятивный эффект: серия команд отменяет серию последних действий. Другие аналогичные средства — команда Правка ▸ Отменить действие и кнопка Отменить действие на панели инструментов Стандартная. Длинные последовательности действий можно отменять также с помощью списка действий (кнопка, раскрывающая список, присоединена к кнопке Отменить действие).

После отмены ряда действий существует возможность вернуться к состоянию, предшествовавшему отмене. Для этого служит команда Правка ▸ Вернуть действие или кнопка Вернуть действие на панели инструментов Стандартная. (К ней также присоединена кнопка, раскрывающая список действий, допускающих возврат.)

Расширенный буфер обмена. При компиляции документа путем использования фрагментов текста, взятых из разных первоисточников, удобно пользоваться расширенным буфером обмена. Впервые расширенный буфер обмена появился в версии *Microsoft Word 2000*, в *Word XP* его объем был удвоен. Расширенный буфер обмена может хранить до 24 объектов (имеются также ограничения на общий объем используемой памяти).



Если между двумя последовательными операциями копирования текста в буфер обмена не было операции вставки, программа автоматически открывает Область задач в режиме Буфер обмена. Содержание конкретного элемента буфера также указывается в Области задач. При переполнении расширенного буфера самый старый элемент теряется, а очередной поступает в освобожденную ячейку.

Область задач позволяет вставить любой из имеющихся элементов, а также скомпоновать их в единый объект и вставить все сразу.

Автотекст. Автотекст — это режим автоматического ввода фрагментов текста. Он представлен двумя функциями: *автозавершением* и собственно *автотекстом*. Их принцип действия состоит в следующем.

Текстовый процессор хранит *словарь автотекста*, состоящий из слов и фраз, встречающихся в документах достаточно часто. При вводе первых четырех символов словарного элемента на экране появляется всплывающая подсказка с полным текстом слова или фразы. Если это то, что имел в виду пользователь, он завершает

ввод всего фрагмента нажатием клавиши ENTER — так работает функция *автозавершения*. Однако пользователь может самостоятельно выбрать необходимый элемент текста из списка с иерархической структурой — это функция *автотекста*. Список элементов автотекста открывается с помощью панели инструментов Автотекст (Вид ▸ Панели инструментов ▸ Автотекст).

Настройку словаря автотекста выполняют в диалоговом окне Автозамена (Вставка ▸ Автотекст ▸ Автотекст). Простейший способ наполнения словаря новым содержанием — выделить текст на экране, щелкнуть на кнопке Автотекст на панели инструментов Автотекст и в открывшемся диалоговом окне использовать кнопку Добавить.

Использование средства автозамены при вводе. Последние версии текстового процессора *Microsoft Word* позволяют эффективно сократить объем вводимого текста за счет использования средства Автозамена. Оно позволяет заменить ввод длинных последовательностей символов произвольным (желательно коротким) сочетанием других символов. Например, если в тексте очень часто встречается словосочетание «диалоговое окно», его можно заменить коротким сочетанием «.до». Соответственно вместо «диалоговых окон» использовать «.дн», а вместо «диалогового окна» — «.да». Точка перед символами стоит специально, чтобы отличать их от двухбуквенных предлогов или союзов, таких как «да».

Настройку средства Автозамена выполняют в диалоговом окне Сервис ▸ Параметры автозамены. Для этого надо установить флажок Заменять при вводе, ввести заменяемую комбинацию в поле Заменить, а замещающую комбинацию в поле На, после чего пополнить список автозамены щелчком на кнопке Добавить.

Как будет показано ниже, средство автоматической замены символов при вводе используется также для ввода специальных символов. Например, выполнив соответствующие настройки, можно вводить греческие буквы π и ρ обычным русским текстом: «пи» или «ро».

Ввод специальных и произвольных символов. При вводе текста часто существует необходимость ввода специальных символов, не имеющих соответствующей клавиши в раскладке клавиатуры, а также произвольных символов, раскладка для которых неизвестна. Основным средством для ввода специальных и произвольных символов, а также для закрепления их за избранными клавишами является диалоговое окно Символ (Вставка ▸ Символ). Данное диалоговое окно имеет две вкладки: Символы и Специальные знаки.

На вкладке Специальные знаки присутствует список специальных символов, таких как «длинное» («полиграфическое») тире, «копирайт», «торговая марка» и других. Для вставки такого символа достаточно щелкнуть на кнопке Вставить. Вместе с тем, для большинства специальных символов существуют клавиатурные комбинации — они приведены в списке, и их стоит запомнить. На первых порах, пока навык их ввода не закреплен, это окно используют для получения справки. В том же окне имеются кнопки Автозамена и Сочетание клавиш, позволяющие либо выполнять ввод специальных символов обычными символами и автоматически производить замену, либо закрепить специальный символ за избранной комбинацией клавиш.

На вкладке Символы представлены элементы управления для ввода произвольных символов любых символьных наборов (рис. 10.4). Центральное положение в окне занимает таблица символов текущего набора. Выбор шрифта выполняют в раскрывающемся списке Шрифт. Если шрифт относится к категории универсальных шрифтов *UNICODE*, то для него имеется и возможность выбора символьного набора в раскрывающемся списке Набор.

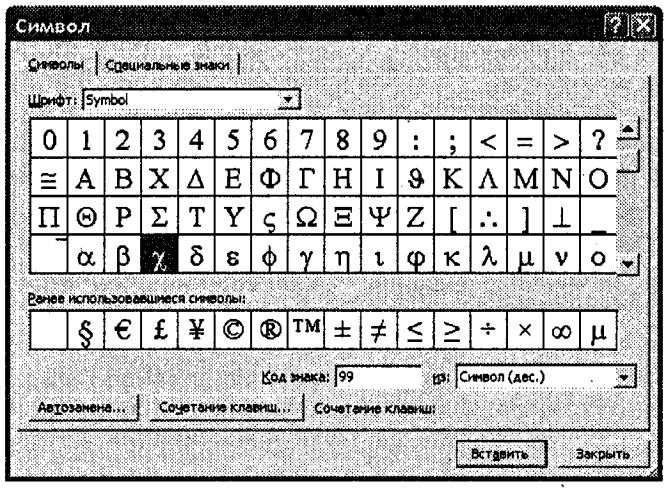


Рис. 10.4. Средство ввода специальных символов

Если символ надо вставить только один раз, достаточно щелкнуть на командной кнопке Вставить. Если предполагается многократное использование данного символа, за ним можно закрепить постоянную комбинацию клавиш (кнопка Сочетание клавиш) или создать элемент для списка Автозамена с помощью одноименной кнопки.

Специальные средства редактирования текста

Базовые приемы редактирования текста мы рассмотрели в разделе, посвященном стандартному приложению Блокнот. В данном разделе мы рассмотрим специальные средства редактирования, характерные для текстового процессора *Microsoft Word*, на примере последней версии *Microsoft Word XP*.

Режимы вставки и замены символов. Текстовый процессор предоставляет возможность выбора между двумя режимами редактирования текста: *режимом вставки* и *режимом замены*. В режиме вставки вводимый текст «раздвигает» существующий текст, а в режиме замены новые символы замещают символы предшествующего текста, находившиеся в точке ввода. Режим вставки применяют при разработке основных содержательных блоков текстовых документов, а режим замены — при редактировании стандартных форм и стандартных элементов (колонтитулов, реквизитных элементов в письмах, служебных записках, бланках).

Текущий режим правки текста индицируется в строке состояния. В режиме замены индикатор ЗАМ в строке состояния окна программы включен, в противном случае

он выключен. Двойной щелчок на этом индикаторе позволяет переключать режимы. Настройка режима правки выполняется на вкладке Правка диалогового окна Параметры (Сервис ▸ Параметры ▸ Правка).

Если установлены флажки Режим замены и Использовать клавишу INS для вставки, правка осуществляется в режиме замены символов. Если оба эти флажка сброшены, то режим можно выбирать с помощью клавиши INSERT. Если флажок Режим замены сброшен, а флажок Использовать клавишу INS для вставки установлен, то правка осуществляется в режиме вставки. Возможность изменять режим путем двойного щелчка на индикаторе в строке состояния сохраняется в любом случае.

Использование Тезауруса. Тезаурус представляет собой словарь смысловых синонимов. При подготовке технической документации особую роль играют смысловые синонимы к используемым глаголам. Для выделенного слова тезаурус удобно вызывать через пункт Синонимы контекстного меню. Однако этот прием срабатывает далеко не для всех слов (преимущественно для глаголов в неопределенной форме). Общий прием вызова тезауруса состоит в использовании команды строки меню Сервис ▸ Язык ▸ Тезаурус.

Окно Тезаурус имеет две панели. Его интересная особенность состоит в том, что в то время, как на левой панели отображаются синонимы выделенного слова, на правой панели могут отображаться синонимы к выбранному синониму, то есть поиск синонима является как бы двухуровневым. Заменяющий синоним можно выбирать как на левой панели, так и на правой. Замена производится щелчком на командной кнопке Заменить. Кроме синонимов в некоторых случаях тезаурус позволяет находить *антонимы* слов и *связанные* (как правило, однокоренные) слова.

Средства автоматизации проверки правописания. Средства автоматизации проверки правописания включают средства проверки орфографии и грамматики. Текстовый процессор позволяет реализовать два режима проверки правописания — *автоматический* и *командный.*

Для работы в автоматическом режиме надо установить флажки Автоматически проверять орфографию и Автоматически проверять грамматику на вкладке Правописание диалогового окна Параметры (Сервис ▸ Параметры ▸ Правописание). В автоматическом режиме слова, содержащие орфографические ошибки, подчеркиваются красным цветом, а выражения, содержащие грамматические ошибки, — зеленым. Для того чтобы узнать характер ошибки, надо щелкнуть правой кнопкой мыши на помеченном фрагменте. В зависимости от характера ошибки контекстное меню содержит пункт Орфография или Грамматика. С их помощью открывается диалоговое окно, в котором имеются элементы управления для получения более точной справки о том, какое правило нарушено, и предложены варианты исправления предполагаемой ошибки.

Встроенное автоматическое средство проверки правописания является, по существу, экспертной системой и допускает настройку. Так, например, если рекомендации экспертной системы неточны или неприемлемы, от них можно отказаться командой Пропустить (обычно такое бывает при проверке грамматики). Если же слово отмечено как орфографическая ошибка только потому, что оно отсутствует

в словаре системы автоматической проверки (например, слово браузер), то его можно добавить в словарь.

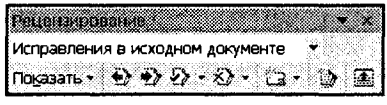
Встроенный словарь системы проверки правописания не подлежит правке. Все дополнения и изменения вносятся в специальный подключаемый *пользовательский словарь*. Каждый пользователь может создать несколько специализированных пользовательских словарей, ориентированных на различные области знаний (автомобильное дело, машиностроение, вычислительная техника и т. п.). Подключение нужного словаря для работы с конкретным документом выполняется выбором словарного файла в диалоговом окне Вспомогательные словари. Чтобы открыть его, надо щелкнуть на кнопке Словари на вкладке Сервис ▸ Параметры ▸ Правописание. Постепенно наполняясь конкретным содержанием, вспомогательные словари пользователя становятся мощным средством повышения производительности его труда.

В командном режиме проверка правописания выполняется независимо от установки элементов управления на вкладке Сервис ▸ Параметры ▸ Правописание. Запуск средства проверки выполняют командой Сервис ▸ Правописание. Проверка начинается от местоположения курсора и продолжается до появления первой ошибки. После исправления ошибки проверка продолжается дальше, а по достижении конца документа проверка может быть продолжена с его начала. Естественное завершение проверки происходит, когда документ просмотрен целиком.

В тех случаях, когда пользователь отказывается от предлагаемых исправлений и дает команду Пропустить, в документе накапливается *список пропускаемых слов*, то есть слов и выражений, не подлежащих проверке. Для того чтобы очистить этот список и начать проверку заново, используют командную кнопку Сервис ▸ Параметры ▸ Правописание ▸ Повторная проверка.

Средства рецензирования текста

Под рецензированием можно понимать два процесса: *редактирование текста с регистрацией изменений* и *комментирование* текста. В отличие от обычного редактирования при рецензировании текст документа изменяется не окончательно — новый вариант и старый «сосуществуют» в рамках одного документа на правах различных *версий*.



Основным средством рецензирования является панель Рецензирование (Вид ▸ Панели управления ▸ Рецензирование). На ней представлены несколько групп элементов управления, предназначенных для:

- создания, просмотра и удаления примечаний;
- регистрации, просмотра, принятия и отмены изменений;
- выбора цвета выделения примечаний;
- сохранения версий документа.

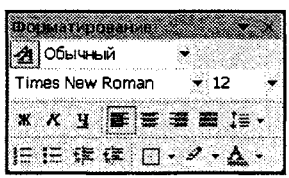
Для создания примечания служит кнопка Создать примечание. При ее использовании местоположение курсора выделяется заданным цветом, а на правом поле

страницы открывается область ввода текста примечания. Созданное примечание отображается только при просмотре документа, но не при его печати.

Для регистрации изменений в тексте служит кнопка Исправления. Все редактирование текста в режиме регистрации исправлений считается неавторским и выделяется особым методом (метод выделения можно задать на вкладке Исправления диалогового окна Сервис ▸ Параметры). Прочие элементы управления данной панели позволяют выполнять переходы между исправлениями, принимать их или отвергать.

Если документ проходит многоступенчатое редактирование, часто возникает необходимость хранить его промежуточные версии. Текстовый процессор *Microsoft Word XP* позволяет хранить несколько версий документа в одном файле. Это удобное средство отличается тем, что при сохранении нескольких версий (в отличие от нескольких копий) эффективно используется рабочее место на диске. Дело в том, что при сохранении очередной версии не происходит повторного сохранения всего документа — сохраняются только отличия текущей версии от предшествующей. Для сохранения текущей версии и для загрузки одной из промежуточных версий применяют команду Файл ▸ Версии.

Форматирование текста



Форматирование текста осуществляется средствами меню Формат или панели Форматирование. Основные приемы форматирования включают:

- выбор и изменение гарнитуры шрифта;
- управление размером шрифта;
- управление начертанием и цветом шрифта;
- управление методом выравнивания;
- создание маркированных и нумерованных списков (в том числе многоуровневых);
- управление параметрами абзаца.

Возможно, вы обратили внимание на то, что внешний вид панели Форматирование, представленной на рисунке справа, отличается от того, что вы видите в своей программе. Это связано с тем, что все инструментальные панели программы Word можно сделать плавающими. Перетаскиванием за рубчик, имеющийся на левом краю панели, их можно переместить в любое место экрана, в том числе и вне пределов основного рабочего окна. Правда, преимущества такого приема ощутят не все пользователи, а только те, у кого мониторы имеют достаточно большой размер.

Настройка шрифта. При выборе гарнитуры шрифта следует иметь в виду следующие обстоятельства:

- Выбор гарнитуры шрифта действует на выделенный текстовый фрагмент. Если ни один фрагмент не выделен, он действует на весь вводимый текст до очередной смены гарнитуры.
- Начиная с версии *Microsoft Word* 97, текстовые процессоры *Word* ориентированы на работу с многоязычными шрифтовыми наборами (*UNICODE*). При использовании других шрифтовых наборов возможны проблемы, которые воз-

никают при переключении раскладки клавиатуры с основной (английской) на дополнительную (русскую). В таком случае возможен неконтролируемый автоматический возврат к использованию одного из стандартных шрифтов *UNICODE*, зарегистрированных в операционной системе.

Напомним, что как операционная система Windows XP, так и сам текстовый процессор Microsoft Word поставляются с наборами шрифтов UNICODE, то есть использование шрифтов, входящих в стандартную поставку, является гарантией от непредвиденных осложнений.

Настройку шрифта выполняют в диалоговом окне Шрифт (Формат ▸ Шрифт). В версии *Microsoft Word XP* данное диалоговое окно имеет три вкладки: Шрифт, Интервал и Анимация.

На вкладке Шрифт выбирают:

- гарнитуру шрифта;
- его размер (измеряется в полиграфических пунктах);
- вариант начертания;
- цвет символов;
- наличие подчеркивания;
- характер видоизменения.

При выборе гарнитуры шрифта следует иметь в виду, что существует две категории шрифтов: с засечками и без засечек (*рубленые*). Характерными представителями первой категории являются шрифты семейства *Times*, а второй категории — шрифты семейства *Arial*. Шрифты, имеющие засечки, легче читаются в больших текстовых блоках — их рекомендуется применять для оформления основного текста.

Шрифты, не имеющие засечек, рекомендуется использовать для заголовков в технических текстах, а также для оформления дополнительных материалов (врезок, примечаний и прочего).

Кроме того, считается, что шрифты с засечками лучше воспринимаются в документах, напечатанных на бумаге. Для электронных документов, которые предполагается читать с экрана, многие предпочитают применять рубленые шрифты.

Большинство гарнитур шрифтов являются *пропорциональными*. Это означает, что и ширина отдельных символов, и расстояние между соседними символами не являются постоянными величинами и динамически меняются так, чтобы сопряжение символов было наиболее благоприятным для чтения.

Особую группу представляют так называемые *моноширинные* шрифты. В них каждый символ вместе с окаймляющими его интервалами имеет строго определенную ширину. Такие шрифты применяют в тех случаях, когда надо имитировать шрифт пишущей машинки, а также при вводе текстов, представляющих листинги программ. Характерными представителями таких шрифтов являются шрифты семейства *Courier*.

При выборе размера шрифта руководствуются назначением документа, а также вертикальным размером печатного листа. Для документов, имеющих формат типо-

вой книжной страницы, обычно применяют шрифт размером 10 пунктов. Для документов, готовящихся для печати на стандартных листах формата А4 (210×297 мм), выбирают размер 12 пунктов. При подготовке документов, предназначенных для передачи средствами факсимильной связи, применяют увеличенный размер — 14 пунктов и больше (факсимильные документы часто воспроизводятся с искажениями, и увеличенный размер шрифта улучшает удобство их чтения).

При подготовке электронных документов, распространяемых в формате *Microsoft Word*, размер шрифта выбирают, исходя из разрешения экрана. В настоящее время наиболее распространены компьютеры, видеоподсистема которых настроена на экранное разрешение 800×600 точек или 1024×768 точек. Для этих параметров целесообразно готовить электронные документы с размером шрифта 12 пунктов. На этот размер по умолчанию настроены последние версии процессора *Microsoft Word XP*. (Версия *Microsoft Word* 97 была настроена по умолчанию на размер экранного шрифта 10 пунктов, но практика показала, что он неудобен.)

Использование прочих средств управления шрифтом (выбор начертания, подчеркивания и других видоизменений) определяется стилевым решением документа, которое задает заказчик или работодатель. Приступая к первому заданию, следует выяснить, какие стилевые решения уже существуют в данной организации, каковы ограничения на использование средств оформления и форматирования. По возможности, надо получить от заказчика готовые шаблоны документов или хотя бы печатные образцы.

Из прочих, не рассмотренных здесь средств управления шрифтами надо отметить управление интервалом между символами и возможность использования эффектов анимации. Интервал задается путем выбора одного из трех значений (Обычный, Разреженный, Уплотненный) на вкладке Формат ▸ Шрифт ▸ Интервал.

Эффекты анимации используют очень редко и только при подготовке электронных документов, распространяемых в формате текстового процессора. В печатных документах эти эффекты невоспроизводимы по очевидным причинам, а в *Web*-документах их нет смысла применять, так как они пока не поддерживаются *Web*-браузерами.

Настройка метода выравнивания. Все последние версии текстового процессора *Microsoft Word* поддерживают четыре типа выравнивания:

- по левому краю;
- по центру;
- по правому краю;
- по ширине.

Выбор метода выполняют соответствующими кнопками панели инструментов Форматирование или из раскрывающегося списка Формат ▸ Абзац ▸ Отступы и интервалы ▸ Выравнивание. Избранный метод действует на текущий и последующие вводимые абзацы. Выбор метода выравнивания определяется назначением документа. Так, например, для *Web*-страниц нет смысла выполнять выравнивание по ширине, поскольку все равно неизвестна ширина окна браузера, в котором документ будет просматриваться, однако выравнивание по центру использовать можно.

Для документов, передаваемых на последующую обработку, все методы выравнивания, кроме тривиального выравнивания по левому краю, являются излишними. Для печатных документов, выполненных на русском или немецком языках, рекомендуется в основном тексте использовать выравнивание по ширине с одновременным включением функции переноса, а для документов на английском языке основной метод выравнивания — по левому полю.

Настройка параметров абзаца. Кроме режима выравнивания настраиваются следующие параметры абзаца:

- величина отступа слева (от левого поля);
- величина отступа справа (от правого поля);
- величина отступа первой строки абзаца («красная строка»);
- величина интервала (отбивки между абзацами) перед абзацем и после него.

Для печатных документов величину отступа для основного текста, как правило, не задают (необходимое положение текста определяется шириной полей), но ее задают для дополнительных материалов и заголовков, если они не выравниваются по центру. В то же время, для *Web*-страниц величина отступа для абзацев имеет большое значение. Это один из весьма немногих параметров форматирования, допускаемых для *Web*-документов, поэтому его используют очень широко.

Роль отбивок между абзацами, как и роль отступа первой строки абзаца, состоит в том, чтобы визуально выделить абзацы. При этом следует помнить, что эти средства несовместимы. То есть, применяя отступ первой строки абзаца, не следует применять отбивки между абзацами, и наоборот. Комбинация этих стилей допускается только для маркированных и нумерованных списков (основной текст оформляется с отступом первой строки, а списки — без него, но с отбивкой между абзацами).

Обычная практика назначения формата состоит в том, что для документов простой структуры (художественных) используют отступ первой строки (это особенно важно для текстов на русском и немецком языках), а для документов сложной структуры (технических) и документов на английском языке используют отбивки между абзацами. Промежуточное положение занимают документы, относящиеся к естественнонаучным и гуманитарным дисциплинам, — при их подготовке кроме точки зрения автора руководствуются сложившейся практикой и устоявшимися традициями.

В *Web*-документах применяют только отбивки между абзацами. Отступ первой строки в них обычно не используют в связи с повышенными трудностями его создания.

Средства создания маркированных и нумерованных списков. Специальное оформление маркированных и нумерованных списков редко применяют в художественных документах и персональной переписке, но в служебных документах и особенно в *Web*-документах оно используется очень широко. В *Web*-документах оформление маркированных списков особо усиливают за счет применения специальных графических маркеров, стиль которых должен тематически сочетаться с содержанием и оформлением документов.

Для создания нумерованных и маркированных списков нужно сначала выполнить *настройку*, затем *вход* в список и, наконец, *выход* из него. Настройку выполняют в диалоговом окне Список, открываемом командой Формат ▸ Список. Данное окно имеет четыре вкладки: Маркированный список, Нумерованный список, Многоуровневый список и Список стилей. В качестве элементов управления здесь представлены образцы оформления списков. Для выбора нужного достаточно щелкнуть на избранном образце.

Вход в список может осуществляться автоматически или по команде. Чтобы автоматически создать маркированный список, достаточно начать запись строки с ввода символа «*». По завершении строки и нажатии клавиши ENTER символ «*» автоматически преобразуется в маркер, а на следующей строке маркер будет установлен автоматически. Для автоматического создания нумерованного списка достаточно начать строку с цифры, после которой стоят точка и пробел, например «1. », «2. » и т. д. Этот метод позволяет начать нумерацию с любого пункта (не обязательно с единицы).

Для создания списка по команде служат кнопки Нумерация и Маркеры, представленные на панели Форматирование. Как маркированный, так и нумерованный список легко превратить в многоуровневый. Для перехода на новые (или возврата на предшествующие уровни) служат кнопки Увеличить отступ и Уменьшить отступ на панели Форматирование.

Для списков с очень глубоким вложением уровней (более трех) можно настроить стиль оформления каждого из уровней. Для этого служит командная кнопка Изменить на вкладке Многоуровневый диалогового окна Список (Формат ▸ Список).

Вкладка Список стилей также предназначена для оформления многоуровневых списков. Она позволяет выбрать или определить для каждого уровня списка особый стиль. О работе со стилями мы поговорим чуть позже.

Характерной особенностью процессора *Microsoft Word XP*, связанной с его ориентацией на создание *Web*-документов, является возможность использования графических маркеров. Для выбора такого маркера на вкладке Маркированный диалогового окна Список (Формат ▸ Список) выберите один из образцов списка и щелкните на кнопке Изменить. Откроется диалоговое окно Изменение маркированного списка, в котором надо щелкнуть на кнопке Рисунок. Эта кнопка открывает диалоговое окно Рисованный маркер, в котором можно выбрать подходящее изображение.

Для завершения маркированного или нумерованного списка и выхода из режима его создания достаточно по завершении ввода последней строки дважды нажать клавишу ENTER.

10.3. Приемы и средства автоматизации разработки документов

С рядом приемов автоматизации ввода и редактирования текста мы познакомились выше. К ним относятся средства Автотекст, Автозамена, средства проверки правописания, средства расстановки переносов, средства поиска и замены фрагментов текста.

В этом разделе мы познакомимся с наиболее общими средствами автоматизации разработки и оформления документов, к числу которых относятся *стили оформления абзацев*, *шаблоны документов* и *темы оформления*.

Работа со стилями

Абзац — элементарный элемент оформления любого документа. Каждый заголовок документа тоже рассматривается как отдельный абзац. Выше мы видели, что в меню Формат ▸ Абзац имеется немало различных элементов управления, и выполнять их настройку для каждого абзаца отдельно — неэффективная и утомительная задача. Она автоматизируется путем использования понятия *стиль*.

Стиль оформления — это именованная совокупность настроек параметров шрифта, абзаца, языка и некоторых элементов оформления абзацев (линий и рамок). Благодаря использованию стилей обеспечивается простота форматирования абзацев и заголовков текста, а также единство их оформления в рамках всего документа.

Особенностью текстовых процессоров *Microsoft Word* является то, что они поддерживают четыре типа стилей: *стили абзаца*, *стили знаков* (символов), *стили списков* и *стили таблиц*. С помощью стилей абзаца выполняют форматирование абзацев, а с помощью знаковых стилей можно изменять оформление выделенных фрагментов текста внутри абзаца. Стиль списка предполагает наличие в начале абзаца номера или маркера. Стиль таблицы обеспечивает согласование границ, заливки, выравнивания и шрифтов в таблицах.

Наличие разных типов стилей позволяет реализовать довольно сложные приемы форматирования. Например, внутри абзаца, оформленного одним шрифтом, могут содержаться фрагменты текста, оформленные другим шрифтом. В данной книге, например, специальный шрифт использован для записи названий элементов управления.

Работа со стилями состоит в создании, настройке и использовании стилей. Некоторое количество стандартных стилей присутствует в настройке программы по умолчанию, сразу после ее установки. Их используют путем выбора нужного стиля из раскрывающегося списка на панели управления Форматирование.

Все работы по созданию новых стилей и изменению существующих выполняют с помощью Области задач в режиме Стили и форматирование. Если Область задач закрыта или находится в ином режиме, надо дать команду Формат ▸ Стили и форматирование.

Настройка стиля. Настройку стиля (рис. 10.5) выполняют в диалоговом окне Стиль (Формат ▸ Стиль). Настраиваемый стиль выбирают в списке Стили (при этом на панелях Абзац и Знаки отображаются образцы применения данного стиля). Для изменения стиля служит командная кнопка Изменить, открывающая диалоговое окно Изменение стиля. Каждый из компонентов стиля настраивается в отдельном диалоговом окне. Выбор компонента выполняют в меню, открываемом с помощью командной кнопки Формат.

При проведении настройки стиля важно правильно выбрать исходный стиль. Он должен быть как можно ближе к желаемому, чтобы минимизировать количество необходимых настроек.

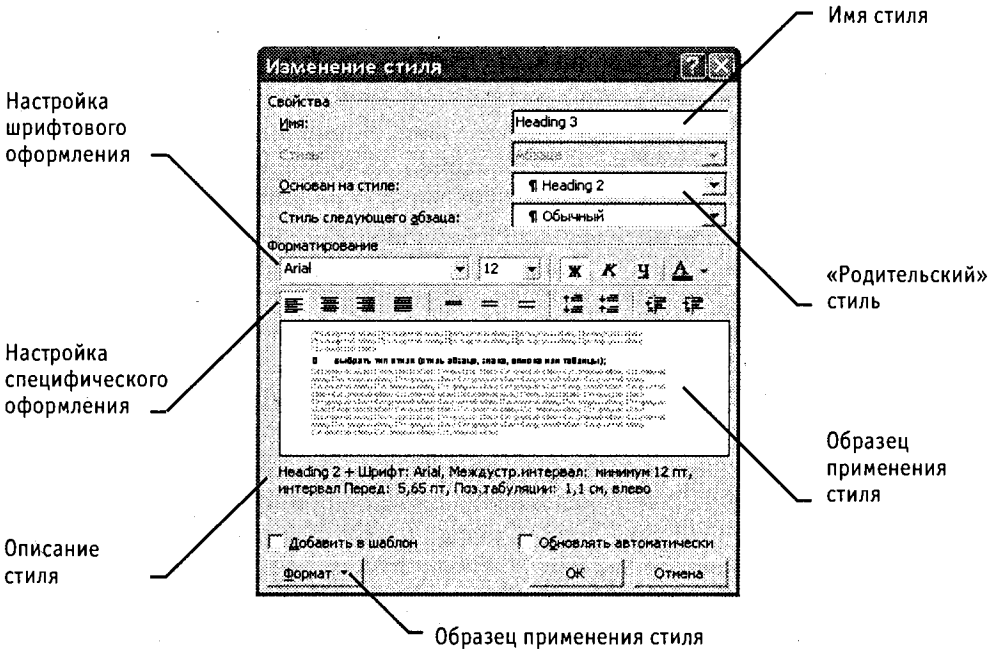


Рис. 10.5. Настройка стиля

Создание стиля. Для создания нового стиля надо щелкнуть на кнопке Создать стиль в Области задач. Откроется диалоговое окно Создание стиля.

В данном окне следует:

- ввести название нового стиля в поле Имя;
- выбрать тип стиля (стиль абзаца, знака, списка или таблицы);
- выбрать стиль, на котором основан новый стиль;
- указать стиль следующего абзаца;
- настроить основные элементы стиля, используя средства данного диалогового окна;
- настроить дополнительные элементы стиля с помощью кнопки Формат.

Важной чертой программы является принцип *наследования стилей*. Он состоит в том, что любой стиль может быть основан на каком-то из существующих стилей. Это позволяет, во-первых, сократить настройку стиля до минимума, сосредоточившись только на отличиях от базового, а во-вторых, обеспечить принцип единства оформления всего документа в целом. Так, например, при изменении базового стиля автоматически произойдут и изменения наследуемых элементов в стилях, созданных на его основе.

Стиль следующего абзаца указывают для обеспечения автоматического применения стиля к следующему абзацу, после того как предыдущий абзац закрывается клавишей ENTER.

Разработка новых стилей и их настройка являются достаточно сложными технологическими операциями. Они требуют тщательного планирования, внимательности и аккуратности, особенно в связи с тем, что согласно принципу наследования свойств стилей желаемые изменения в одном стиле могут приводить к нежелательным изменениям во многих других стилях.

В связи с трудоемкостью изучения и освоения приемов практической работы со стилями начинающие пользователи часто ими пренебрегают. Действительно, при разработке небольших документов (одна-две страницы) можно обойтись без настройки и использования стилей, выполнив все необходимое форматирование вручную средствами меню Формат. Однако при разработке объемных документов вручную очень трудно обеспечить единство оформления, особенно если разные разделы документа разрабатывались разными авторами.

☑ Прийти к использованию стилей надо как можно раньше. Правильное и рациональное использование этого средства является залогом высокой эффективности работы с процессором Microsoft Word и высокого качества разрабатываемых документов. На изучение средств управления стилями может потребоваться несколько часов, но полученные при этом навыки останутся на всю жизнь и пригодятся многократно.

Шаблоны

Совокупность удачных стилевых настроек сохраняется вместе с готовым документом, но желательно иметь средство, позволяющее сохранить их и вне документа. Тогда их можно использовать для подготовки новых документов. Такое средство есть — это шаблоны, причем некоторое количество универсальных шаблонов поставляется вместе с текстовым процессором и устанавливается на компьютере вместе с ним.

По своей сути, шаблоны — это тоже документы, а точнее говоря, заготовки будущих документов. От обычных документов шаблоны отличаются тем, что в них приняты специальные меры, исключающие возможность их повреждения. Открывая шаблон, мы начинаем новый документ и вносим изменения в содержание шаблона. При сохранении же мы записываем новый документ, а шаблон, использованный в качестве его основы, остается в неизменном виде и пригоден для дальнейшего использования.

Использование шаблона для создания документа. По команде Файл ▸ Создать открывается Область задач в режиме Создание документа. Здесь можно выбрать шаблон, на базе которого документ будет разрабатываться. В этом случае документ сразу получает несколько готовых стилей оформления, которые содержатся в шаблоне. Основные шаблоны перечислены в области задач в разделе Создание. Если их недостаточно, надо щелкнуть на ссылке Общие шаблоны и выбрать подходящий шаблон на одной из вкладок открывшегося диалогового окна Шаблоны.

Изменение шаблона готового документа. Эта достаточно редкая операция выполняется с помощью диалогового окна Шаблоны и настройки (Сервис ▸ Шаблоны и настройки). Для смены текущего шаблона следует использовать кнопку Присоединить и в открывшемся диалоговом окне Присоединение шаблона выбрать нужный шаблон в папке C:\Program Files\Microsoft Office\Шаблоны.

Создание нового шаблона на базе шаблона. Открыв диалоговое окно Шаблон щелчком на ссылке Общие шаблоны в Области задач (режим Создание документа), включите переключатель Шаблон. Теперь надо выбрать стандартный шаблон, на базе которого создается новый (рис. 10.6). После настройки стилей и редактирования содержания выполняется сохранение шаблона командой Сохранить как с включением пункта Шаблон документа в поле Тип файла.

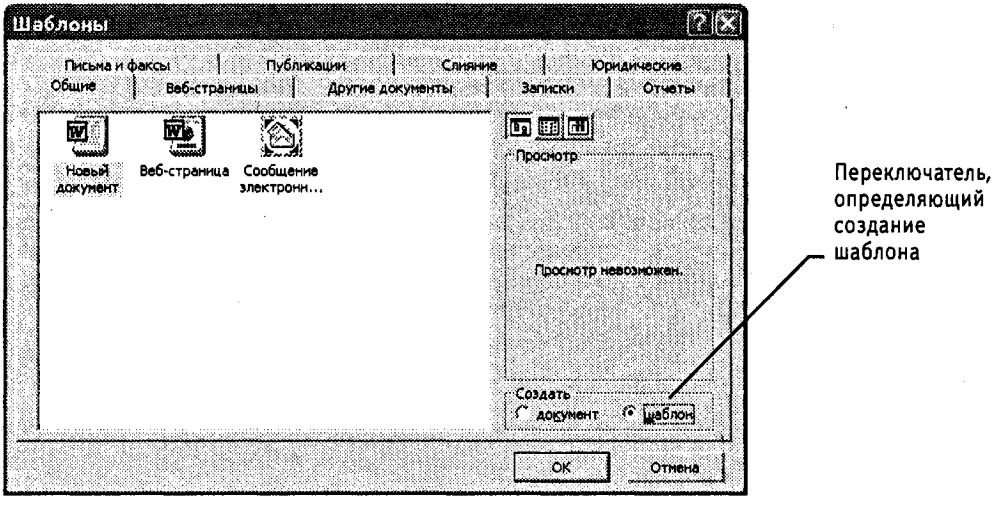


Рис. 10.6. Диалоговое окно Создание документа

Создание нового шаблона на базе документа. Если готовый документ может быть использован в качестве заготовки для создания других документов, его целесообразно сохранить как шаблон. Командой Файл ▸ Открыть открывают готовый документ, в нем правят содержание и настраивают стили, а потом сохраняют документ как шаблон командой Сохранить как с включением пункта Шаблон документа в поле Тип файла.

Темы

Последние версии текстового процессора *Microsoft Word* (начиная с *Word 2000*) имеют специальное средство автоматического оформления, предназначенное в первую очередь для электронных документов (для *Web*-документов и документов, распространяемых в формате процессора). Это средство называется *темы*. Тема представляет собой совокупность следующих элементов оформления:

- фоновый узор;
- стили оформления основного текста и заголовков;
- стиль оформления маркированных списков;
- стиль графических элементов оформления (линий).

Доступ к выбору тем выполняется командой Формат ▸ Темы.

Практическое занятие

Упражнение 10.1. Первичные настройки текстового процессора Microsoft Word XP

1. Запустите текстовый процессор командой Пуск ▸ Программы ▸ Microsoft Word.
2. Откройте заранее подготовленный файл (произвольный).
3. Откройте меню настройки панелей управления (Вид ▸ Панели управления) и убедитесь в том, что включено отображение только двух панелей: Стандартная и Форматирование.
4. В качестве режима отображения документа выберите Режим разметки. Для этого используйте соответствующую кнопку в левом нижнем углу окна документа или команду Вид ▸ Разметка страницы.
5. Если шрифт на экране выглядит слишком мелким, настройте масштаб отображения командой Вид ▸ Масштаб. Можно также использовать раскрывающийся список Масштаб на панели инструментов Стандартная. Если желаемого масштаба нет в списке (например, 125%), введите нужное значение непосредственно в поле списка и нажмите клавишу ENTER. Для эффективного использования площади окна документа при достаточном разрешении экрана можно использовать пункты По ширине страницы или По ширине текста.
6. В качестве единицы измерения для настройки параметров документа выберите миллиметры (Сервис ▸ Параметры ▸ Общие ▸ Единицы измерения).
7. Настройте список быстрого открытия документов. После запуска программы в меню Файл можно найти список из нескольких документов, открывавшихся в текстовом процессоре в последнее время. Это удобно для быстрого открытия нужного документа. Количество документов, отображаемых в этом списке, задайте счетчиком Сервис ▸ Параметры ▸ Общие ▸ Помнить список из ... файлов.
8. Отключите замену выделенного фрагмента при правке текста, сбросив флажок Сервис ▸ Параметры ▸ Правка ▸ Заменять выделенный фрагмент. Это несколько снижает производительность труда при редактировании текста, но страхует начинающих от нежелательных ошибок. С набором опыта практической работы этот флажок можно установить вновь.
9. Включите контекстно-чувствительное переключение раскладки клавиатуры (Сервис ▸ Параметры ▸ Правка ▸ Автоматическая смена клавиатуры). Эта функция удобна при редактировании текста. При помещении курсора в английский текст автоматически включается англоязычная раскладка, а при помещении его в текст на русском языке — русскоязычная.
10. Запретите «быстрое» сохранение файлов, сбросив флажок Сервис ▸ Параметры ▸ Сохранение ▸ Разрешить быстрое сохранение. При «быстром» сохранении сохраняется не сам файл, а только его изменения по сравнению с предыдущей сохраненной версией. Это действительно сокращает время операции сохранения, но замедляет другие операции с документами. При этом также заметно возрастают размеры итогового файла.

11. Настройте функцию *автосохранения* с помощью счетчика Сервис ▸ Параметры ▸ Сохранение ▸ Автосохранение каждые ... минут. Имейте в виду следующие обстоятельства:
    - при автосохранении данные записываются в специальный файл, который в аварийных ситуациях может быть использован для восстановления несохраненных данных, но только однократно(!);
    - функция автосохранения не отменяет необходимости периодически во время работы и после ее завершения сохранять файл прямыми командами Сохранить или Сохранить как.
12. Временно отключите средства проверки правописания. На вкладке Сервис ▸ Параметры ▸ Правописание сбросьте флажки Автоматически проверять орфографию и Автоматически проверять грамматику. На ранних этапах работы с документом надо сосредоточиться на его содержании, а средства проверки правописания могут отвлекать от этого. Завершая работу над документом, необходимо вновь подключить и использовать эти средства.
13. Временно отключите функцию *автозамены при вводе* сбросом флажка Сервис ▸ Параметры автозамены ▸ Автозамена ▸ Заменять при вводе.
14. Включите автоматическую замену «прямых» кавычек парными: Сервис ▸ Параметры автозамены ▸ Автоформат при вводе ▸ Заменять при вводе «прямые» кавычки парными. В русскоязычных текстах прямые кавычки не применяются. Для подготовки англоязычных текстов и листингов программ отключите эту функцию.
15. Временно отключите ряд средств автоматического форматирования, в частности автоматическую маркировку и нумерацию списков. На вкладке Сервис ▸ Параметры автозамены ▸ Автоформат при вводе сбросьте флажки Применять при вводе стили маркированных списков и Применять при вводе стили нумерованных списков. После приобретения первичных навыков работы с текстами вновь подключите эти средства.
16. Отключите Помощника. Помощник — удобное интерактивное средство для получения конкретной справки, но справочная система программы в целом обладает более высокой методической ценностью. В текстовом процессоре *Microsoft Word XP* Помощник «перехватывает» все запросы к справочной системе, поэтому для полноценной работы со справочной системой его надо принудительно отключить.
    - Вызовите Помощника: Справка ▸ Справка: Microsoft Word.
    - Щелкните на изображении Помощника правой кнопкой мыши и выберите в контекстном меню пункт Параметры — откроется диалоговое окно Помощник.
    - На вкладке Параметры сбросьте флажок Использовать Помощника.
    - Закройте диалоговое окно Помощник щелчком на кнопке OK.

    Проверьте, как работает вход в справочную систему: Справка ▸ Справка: Microsoft Word. Вместо Помощника должно открываться окно справочной системы.
17. Отключите автоматическую расстановку переносов. В абсолютном большинстве случаев на ранних этапах работы с документами она не нужна. Для *Web*-документов, для документов, распространяемых в формате текстового процес-

сора, и для документов, передаваемых на последующую обработку, расстановка переносов не только бесполезна, но и вредна. Для документов, которые окончательно форматируются и распечатываются в одной рабочей группе, расстановка переносов может быть полезной, но и в этом случае ее применяют только на заключительных этапах форматирования и при этом очень тщательно проверяют соответствие переносов, расставленных автоматически, нормам и правилам русского языка.

Расстановку переносов отключают сбросом флажка Сервис ▸ Язык ▸ Расстановка переносов ▸ Автоматическая расстановка переносов.

18. Включите запрос на подтверждение изменения шаблона «Обычный»: Сервис ▸ Параметры ▸ Сохранение ▸ Запрос на сохранение шаблона Normal.dot. Шаблон «Обычный» является первоосновой для всех остальных шаблонов (они создаются на его базе и наследуют его свойства). При обычной работе с программой необходимость его изменения не возникает (если надо что-то изменить в этом шаблоне, достаточно создать его копию под другим именем и работать с ней). Включением данного флажка предупреждаются случайные внесения изменений в шаблон со стороны пользователя, а также попытки макровирусов сохранить свой код в данном шаблоне (для дальнейшего размножения в документах, создаваемых на его основе).

Мы научились выполнять первичные настройки текстового процессора и узнали, что доступ к ним осуществляется следующими командами:

- Сервис ▸ Параметры;
- Сервис ▸ Параметры автозамены;
- Сервис ▸ Язык;
- Вид ▸ Панели инструментов;
- Вид ▸ Масштаб.

Упражнение 10.2. Первичные настройки параметров печатного документа

Форматирование документов, предназначенных для печати на принтере, выполняется в «привязке» к параметрам печатной страницы. Поэтому создание документов этой категории необходимо начинать с настройки параметров страницы. К этим параметрам относятся прежде всего размер листа бумаги и величина полей.

Характерная ошибка начинающих заключается в том, что они начинают подготовку документов с ввода текста. Интуитивно понятно, что текст — это важнейший компонент документа, но для ввода текста служат программы иного класса — текстовые редакторы. Имея дело с текстовым процессором, начинать надо не с ввода текста документа, а с настройки параметров печатной страницы, поскольку от нее зависят все используемые приемы форматирования. Тем, кому утомительно начинать создание каждого документа с настройки параметров страницы, можно порекомендовать чаще пользоваться заранее заготовленными шаблонами.

1. Запустите текстовый процессор командой Пуск ▸ Программы ▸ Microsoft Word.
2. Дайте команду для создания нового документа: Файл ▸ Создать.

3. Щелкните на ссылке Новый документ в Области задач, которая открылась в режиме Создание документа.
4. Откройте диалоговое окно Параметры страницы (Файл ▸ Параметры страницы).
5. На вкладке Размер бумаги выберите в раскрывающемся списке Размер бумаги пункт A4 210×297 mm (этот формат принят в России в качестве стандартного). При использовании нестандартного формата выбирают пункт Другой и с помощью кнопок счетчиков Ширина и Высота задают его параметры.
6. На вкладке Поля задайте ориентацию бумаги (Книжная или Альбомная). При «альбомной» ориентации бумага располагается длинной стороной по горизонтали.
7. На этой же вкладке задайте размеры полей:

   Верхнее — 15 мм Нижнее — 20 мм

   Левое — 25 мм Правое — 15 мм
8. На вкладке Источник бумаги задайте для нижнего поля интервал от края до колонтитула 12 мм (в нижнем колонтитуле будет размещаться номер печатной страницы).
9. Если предполагается двусторонняя печать (четные страницы печатаются на оборотной стороне нечетных страниц), выберите на вкладке Поля пункт Зеркальные поля в списке Несколько страниц. Восстановите обычную настройку.
10. Проверьте, как действует настройка печати двух страниц на одном листе. Выберите в списке Несколько страниц пункт 2 страницы на листе. На панели Образец рассмотрите результат настройки. Установите «альбомную» ориентацию страниц. Оцените результат настройки. Восстановите «книжную» ориентацию и печать одной страницы на листе.
11. Создайте нижний колонтитул для размещения номера печатной страницы. Дайте команду Вид ▸ Колонтитулы — откроется панель инструментов Колонтитулы. Пользуясь кнопкой Верхний/нижний колонтитулы, создайте область нижнего колонтитула. Вставьте в нее номер страницы щелчком на кнопке Номер страницы на панели инструментов Колонтитулы. Отцентрируйте номер страницы щелчком на кнопке По центру на панели инструментов Форматирование. Закройте панель Колонтитулы. Убедитесь в том, что в документе появились нижние колонтитулы с номерами страниц.

Прямой команды для удаления колонтитулов нет. Чтобы удалить колонтитулы по всему документу, надо очистить область колонтитула на одной из страниц. Колонтитул, лишенный содержимого, удаляется автоматически. Для удаления содержимого колонтитула откройте панель Колонтитулы (Вид ▸ Колонтитулы), переключитесь на верхний или нижний колонтитул (по ситуации) кнопкой Верхний/нижний колонтитулы, выделите элемент содержимого и нажмите клавишу DELETE.

12. Закройте панель инструментов Колонтитулы. Сохраните документ командой Сохранить как, дав ему имя Эксперимент и использовав для сохранения папку \Мои документы.

Мы научились создавать и настраивать печатные документы. В порядке эксперимента мы создали «пустой» документ, имеющий настроенные параметры страницы, стили, соответствующие шаблону «Обычный», и нижний колонтитул для размещения номеров печатных страниц. Мы готовы к наполнению данного документа текстовым содержанием с последующим редактированием и форматированием.

Упражнение 10.3. Ввод специальных символов

30 мин

В этом упражнении мы рассмотрим пять приемов ввода символов греческого алфавита. Особо отметим, что это еще далеко не все возможные приемы для текстового процессора *Microsoft Word*. Упражнение будем выполнять вводом фразы: Длина окружности равна 2πR. Для подготовки к упражнению запустите текстовый процессор и создайте пустой документ, взяв за основу шаблон Обычный.

1. *Замена шрифта*. Введите текст: Длина окружности равна 2pR. Выделите букву «p». На панели Форматирование раскройте список шрифтов и выберите символьный набор Symbol. Символ «p» заменится символом «π».

   Если панель Форматирование скрыта, то доступ к списку шрифтов можно получить командой Формат ▸ Шрифт. Это наиболее стандартный прием. Им можно пользоваться во всех программах, имеющих средства для изменения шрифта, но для его применения нужно заранее знать, какой символ латинского шрифта соответствует нужному символу греческого шрифта, а это не всегда возможно.

2. *Классический подход*. Введите текст: Длина окружности равна 2xR. Выделите символ «x». Откройте программу Таблица символов (Пуск ▸ Программы ▸ Стандартные ▸ Служебные ▸ Таблица символов). В окне этой программы выберите шрифт Symbol. В поле таблицы разыщите символ π, выделите его, щелкните на кнопке Выбрать и на кнопке Копировать. Вернитесь в окно *Microsoft Word* и комбинацией клавиш CTRL+V вставьте из буфера обмена скопированный символ на место выделенного.

   Этот прием действует в большинстве программ. Его применяют, если заранее не известно, какому символу латинского шрифта соответствует необходимый символ.

3. *Использование стиля*. Если документ содержит много символов греческого алфавита, имеет смысл создать для них специальный *знаковый стиль*. На базе существующего знакового стиля, например стиля Основной шрифт абзаца создайте новый знаковый стиль, например Греческий. Для этого откройте Область задач в режиме Стили и форматирование (Формат ▸ Стили и форматирование) и щелкните на кнопке Создать стиль. В диалоговом окне Создание стиля в поле Имя введите имя нового стиля, в раскрывающемся списке Стиль выберите пункт Знака и в списке Основан на стиле выберите базовый стиль. Если предполагается и в дальнейшем создание аналогичных документов, созданный стиль можно сохранить в шаблоне, установив флажок Добавить в шаблон. После этого выберите символьный набор Symbol в раскрывающемся списке на панели Форматирование. В дальнейшем при необходимости ввода греческих букв достаточно на панели форматирование выбрать стиль Греческий.

Этот прием специфичен для программы *Microsoft Word*. Далеко не все текстовые редакторы и процессоры позволяют создавать знаковые стили — большинство используют только стили абзаца, применение которых изменяет шрифт во всем абзаце целиком.

4. *Применение «горячих клавиш»*. Это самый эффективный прием. Нет более быстрого способа ввода нестандартных символов, чем ввод с помощью заранее назначенных клавиатурных комбинаций. Так, например, мы можем закрепить символ π за комбинацией клавиш CTRL+ALT+P и использовать ее всюду, где в этом возникает необходимость.

   Дайте команду Вставка ▸ Символ — откроется диалоговое окно Символ. В списке Шрифт выберите шрифт Symbol. В таблице символов разыщите и выберите символ π. Щелкните на кнопке Сочетание клавиш — откроется диалоговое окно Настройка клавиатуры. Убедитесь в том, что текстовый курсор находится в поле Новое сочетание клавиш (в таких случаях говорят, что *фокус ввода* принадлежит элементу управления Новое сочетание клавиш). Если это не так, переместите фокус ввода в нужное поле последовательными нажатиями клавиши TAB.

   Когда фокус ввода находится в нужном поле, нажмите желаемую комбинацию клавиш, например CTRL+ALT+P. Обратите внимание на запись, появившуюся в поле, и щелкните на кнопке Назначить. Закройте открытые диалоговые окна и проверьте работу данной комбинации.

   Обратите внимание на то, что для одного и того же символа можно назначать несколько комбинаций клавиш. Если нужно изменить назначение, следует в диалоговом окне Настройка клавиатуры выделить назначенную комбинацию и щелкнуть на кнопке Удалить. Если нужно, чтобы назначенная комбинация действовала во всех вновь создаваемых документах, ее можно сохранить в текущем шаблоне, выбрав его в раскрывающемся списке Сохранить изменения.

5. *Использование средства автозамены*. У метода «горячих клавиш» есть существенный недостаток: надо запоминать, какому символу какая комбинация соответствует. Если предполагается ввод множества нестандартных символов, удобно использовать средство автоматической замены символов при вводе.

   Дайте команду Вставка ▸ Символ — откроется диалоговое окно Символ. В списке Шрифт выберите шрифт Symbol. В таблице символов разыщите и выберите символ π. Щелкните на кнопке Автозамена — откроется диалоговое окно Автозамена. В поле Заменить введите заменяемую комбинацию «.пи.». (Зачем символы «пи» оконтурены точками с двух сторон, выясните самостоятельно, экспериментируя с вводом выражения 2πR). Аналогичным образом можно организовать ввод и других символов: «.фи.», «.тау.», «.кси.» и т. д. Как видите, ничего не надо специально запоминать.

В текстовом процессоре Microsoft Word, как и во многих других приложениях Windows, одну и ту же операцию можно выполнить множеством разных способов. У каждого способа есть достоинства и недостатки. Пользователи опытным путем подбирают наиболее удобные для себя приемы. Выбор приема зависит от объема и характера исполняемой работы, а также от периодичности ее исполнения.

В предыдущей главе мы рассмотрели приемы создания *простых* текстовых документов средствами текстового процессора *Microsoft Word*. К условной категории *простых* эти документы были отнесены только потому, что не содержали объектов, встроенных в текст. Соответственно, нами не были рассмотрены вопросы взаимодействия текста и встроенных объектов.

В этой главе мы рассмотрим приемы создания *комплексных* текстовых документов, содержащих специальные элементы оформления и встроенные объекты нетекстовой природы (формулы, таблицы, диаграммы, художественные заголовки, растровые и векторные иллюстрации, а также объекты мультимедиа).

11.1. Приемы управления объектами Microsoft Word

Особенности объектов Word

Текстовый процессор *Word XP* обладает развитой функциональностью по работе с объектами нетекстовой природы. Среди встроенных объектов могут быть стандартные объекты, созданные другими программами (рисунки, анимационные и звуковые клипы и многое другое), а также объекты, созданные средствами самого текстового процессора. В частности, программа позволяет создавать и встраивать геометрические фигуры, художественные заголовки, диаграммы, формульные выражения, заготовленные векторные иллюстрации (клипарты), то есть в ней имеются средства, отдаленно напоминающие средства специализированных графических редакторов. Правда, среди этих средств нет ничего для создания и обработки растровых иллюстраций — их можно только импортировать из других программ, но зато есть средства для управления их визуализацией, например для изменения яркости, контрастности и масштаба изображения.

Несмотря на столь разностороннюю природу объектов, с которыми может работать текстовый процессор *Word XP*, у них есть общие свойства, например такие, как размер, положение на странице, характер взаимодействия с текстом. Сначала мы остановимся на изучении самых общих свойств встроенных объектов, не обсуждая

их природу, — это поможет освоить базовые приемы работы с объектами. А с конкретными свойствами конкретных объектов мы познакомимся чуть позже. Но перед тем как приступать к изучению приемов работы с объектами *Word XP*, необходимо сделать важное замечание о целесообразности их применения. На этот счет существуют весьма противоречивые мнения.

1. Все объекты *Microsoft Word XP* безусловно можно использовать, если документ готовится для печати, то есть предполагается, что он будет передаваться заказчику или распространяться в виде бумажной копии, выполненной на принтере. Оформление документов с помощью встроенных объектов позволяет сделать их *представительными*.
2. Если документ предполагается передать в виде файла для последующей обработки (а именно так передают рукописи в редакции), то все собственные средства программы по созданию и размещению встроенных объектов не только бесполезны, но и вредны. Это связано с тем, что объекты *Microsoft Word XP* не стандартны и не поддерживаются профессиональными программами. Компания *Microsoft* имеет лидирующее положение в отрасли и может не считаться с общепринятыми стандартами и правилами, а внедрять свои. Поэтому объекты, созданные в программах этой компании, могут полноценно использоваться только в других программах той же компании.
3. Из последнего замечания вытекает еще одно направление для использования объектов, созданных в *Microsoft Word*. Их можно успешно экспортировать через буфер обмена *Windows* в другие программные продукты, входящие в пакет *Microsoft Office XP*, например такие, как система управления электронными таблицами *Excel*, система управления базами данных *Access* и другие.

Взаимодействие объектов Word с текстом и страницей

Управление размером и положением объекта. Взгляните на рис. 11.1. Здесь представлен графический объект, встроенный в текст документа. Этот объект обладает рядом свойств. Самое очевидное свойство — его размер. Когда объект выделен,

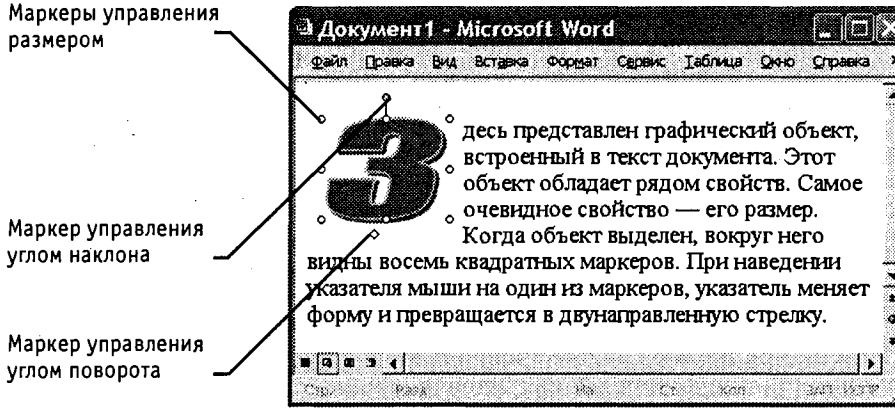


Рис. 11.1. Пример объекта, встроенного в текст

вокруг него видны восемь квадратных маркеров. При наведении указателя мыши на один из маркеров указатель меняет форму и превращается в двунаправленную стрелку. В этот момент размер объекта можно менять методом протягивания мыши. Угловые маркеры позволяют пропорционально изменять размер объекта как по горизонтали, так и по вертикали. Четыре маркера, расположенные на сторонах воображаемого прямоугольника, позволяют управлять размером по одному направлению (по вертикали или горизонтали).

При наведении указателя мыши на сам объект указатель меняет форму и превращается в четырехнаправленную стрелку. В таком состоянии объект можно перетаскивать с помощью мыши по рабочему полю документа. Он займет новое положение в тот момент, когда левая кнопка мыши будет отпущена после перетаскивания.

Расширенное управление свойствами объектов. Вручную мы можем только управлять размером, поворотом и положением объекта на странице. Для управления всеми остальными свойствами объектов нужны дополнительные средства — их можно найти в двух местах:

- на панели инструментов, соответствующей типу объекта (она открывается автоматически, когда объект выделен);
- в диалоговом окне Формат объекта (рис. 11.2), которое открывают из контекстного меню объекта (после щелчка правой кнопкой мыши на объекте).

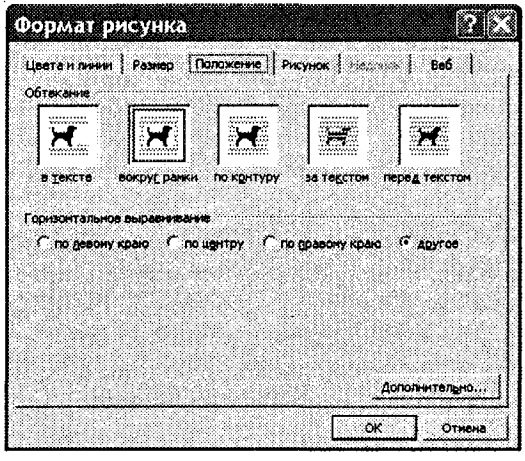


Рис. 11.2. Основное средство управления общими параметрами встроенного объекта

С помощью панели инструментов управляют индивидуальными свойствами объектов (у разных типов объектов они различны), а с помощью диалогового окна Формат объекта управляют наиболее общими свойствами, имеющимися у объектов любых типов.

Взаимодействие объекта с окружающим текстом. Вставив объект в текст, следует задать характер его взаимодействия с текстом. Средства для этого представлены на вкладке Положение диалогового окна Формат объекта. Возможны следующие варианты.

1. Вариант В тексте используют для графических объектов малого размера, сопоставимого с размерами символов текста. В этом случае объект вставляется в текстовую строку на правах графического символа и далее перемещается по странице только вместе с текстом.
2. Вариант Вокруг рамки использован в примере на рис. 11.2. В этом случае текст располагается вокруг воображаемой прямоугольной рамки, охватывающей весь контур объекта.
3. Вариант По контуру отличается от предыдущего тем, что воображаемая прямоугольная рамка не проводится и текст плавно обтекает контур объекта (если он криволинейный).
4. Вариант Перед текстом — это прием вставки объекта без обтекания. Текст и объект лежат на разных слоях, причем объект лежит выше и загораживает часть текста. Этим приемом пользуются, когда оформление важнее содержания.
5. Вариант За текстом — это еще один прием вставки объекта без обтекания. Текст и объект тоже лежат на разных слоях, но в данном случае объект лежит на нижнем слое и загораживается текстом. Этот вариант используют для размещения текста на тематическом художественном фоне.

Дополнительные варианты взаимодействия текста со встроенным объектом можно найти в диалоговом окне Дополнительная разметка, которое открывают с помощью кнопки Дополнительно.

6. Вариант Сквозное — это прием обтекания, аналогичный обтеканию По контуру, но в данном случае текст обтекает объект не только снаружи, но и изнутри.
7. Там же, в диалоговом окне Дополнительная разметка можно выбрать вариант обтекания Сверху и снизу. Этот прием используют наиболее часто — его считают основным для объектов, ширина которых составляет более половины ширины страницы.

Прочие параметры взаимодействия объекта с окружающим текстом. Более тонкую настройку взаимодействия объектов с текстом выполняют с помощью элементов управления, имеющихся в диалоговом окне Дополнительная разметка. В частности, здесь можно с помощью переключателей конкретно указать, с каких сторон объекта происходит обтекание, а с каких — нет. Здесь же можно указать величину интервала в миллиметрах между текстом и объектом.

Управление горизонтальным положением объекта относительно элементов печатной страницы. Завершив настройку взаимодействия объекта с текстом, приступают к размещению объекта на странице. Как уже говорилось выше, это можно сделать вручную методом перетаскивания объекта с помощью мыши, но более точную настройку выполняют с помощью рассмотренного диалогового окна Формат объекта ▸ Положение.

Варианты горизонтального размещения объекта:

- по левому краю;
- по правому краю;
- по центру;
- другое.

Варианты По левому краю и По правому краю обычно используют при обтекании По контуру или Вокруг рамки. Вариант По центру часто сочетают с обтеканием Сверху и снизу, а последний вариант соответствует ручному размещению объекта перетаскиванием с помощью мыши.

Управление вертикальным положением объекта относительно элементов печатной страницы. К объекту, встроенному в текст, можно подходить с двух позиций: как к элементу оформления страницы или как к элементу оформления содержания, то есть текста. Разница заключается в том, что происходит с объектом во время редактирования текста: он перемещается вместе с ним (с абзацами, к которым он примыкает) или он неподвижен, а текст перемещается, обтекая объект по заданным правилам.

В первом случае объект надо закрепить относительно абзаца, а во втором случае — относительно страницы. Необходимую настройку выполняют элементами управления вкладки Положение рисунка в диалоговом окне Дополнительная разметка. Вертикальное положение объекта относительно элементов страницы задают установкой переключателя Выравнивание и выбором метода выравнивания и элемента, относительно которого происходит выравнивание. Вертикальное положение относительно текста задают установкой переключателя Положение и выбором объекта, относительно которого положение задается, например абзаца.

Чтобы объект был связан с элементом страницы и не перемещался вместе с текстом, устанавливают флажок Установить привязку. Чтобы объект мог перемещаться вместе с текстом, устанавливают флажок Перемещать вместе с текстом.

Управление свойствами объектов Microsoft Word

Управление размерами объекта. Мы знаем, что размерами встроенных объектов можно управлять перетаскиванием графических маркеров с помощью мыши. Это прием ручного управления. Однако существуют и приемы автоматического управления. Их реализуют с помощью элементов управления вкладки Размер рассмотренного выше диалогового окна Формат объекта. Счетчиками Высота, Ширина и Поворот задают вертикальные и горизонтальные размеры объекта, а также его угол поворота по часовой стрелке.

Размерами объектов можно управлять не только в абсолютном исчислении, но и в относительном (в процентах относительно исходного). Для этого служат счетчики группы Масштаб. Чтобы размеры объекта синхронно изменялись по вертикали и горизонтали, надо установить флажок Сохранить пропорции.

Управление свойствами линии. Большинство объектов, создаваемых средствами самой программы *Word XP*, имеют векторную природу, то есть в их основе лежат простейшие геометрические фигуры — линии. Эти линии, в свою очередь, имеют собственные свойства: толщину, цвет и тип. Управление этими свойствами выполняют с помощью средств вкладки Формат объекта ▸ Цвета и линии.

Управление свойствами замкнутых линий. Замкнутые линии, в отличие от обычных, обладают дополнительным свойством — *заливкой*. Свойство заливки задают

на вкладке Формат объекта ▸ Цвета и линии. Заливка может быть *простой* и *комбинированной*. Вид заливки выбирают в раскрывающейся палитре Цвет.

Простая заливка — одноцветная. Цвет заливки может быть одним из сорока стандартных, имеющихся в палитре, или одним из дополнительных (выбирается в палитре с помощью кнопки Другие цвета). Простые цвета отличаются тем, что их можно назначить полупрозрачными, — тогда через закрашенные контуры может просвечивать текст или объект нижележащего слоя (рис. 11.3).

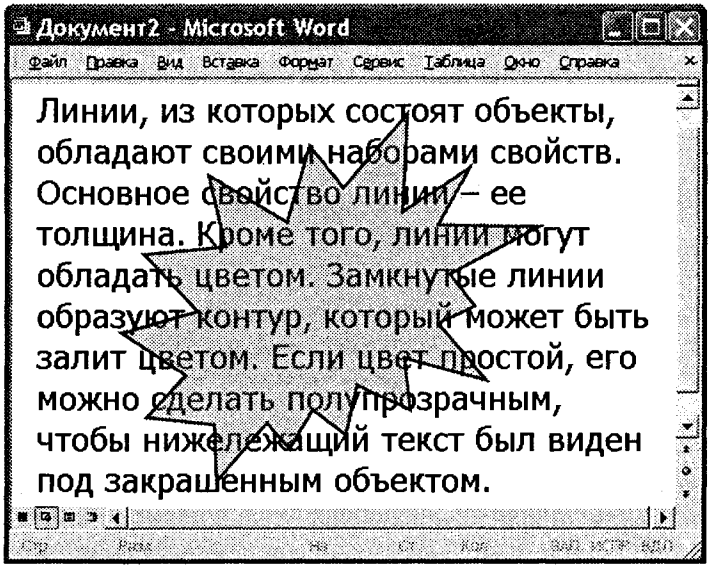


Рис. 11.3. Объекты, залитые сплошным цветом, можно сделать полупрозрачными

Комбинированная заливка имеет более сложный характер. В программе *Word XP* реализовано четыре метода комбинированной заливки:

- градиентная заливка;
- текстурная заливка;
- заливка узором;
- заливка рисунком (изображением-картой).

Для выбора метода комбинированной заливки в палитре цветов имеется кнопка Способы заливки. Она открывает диалоговое окно Способы заливки, имеющее четыре вкладки: Градиентная, Текстура, Узор и Рисунок.

Градиентная заливка — это многоцветная заливка, при которой осуществляется плавный переход между заданными цветами. Количество исходных цветов, сами цвета и направление градиента произвольно выбираются на вкладке Градиентная.

Текстурная заливка — это заливка, воспроизводящая нерегулярную текстуру. Обычно используется для имитации поверхности материала. Выбор текстуры выполняют на вкладке Текстура (рис. 11.4). Если представленных там текстур недостаточно, с помощью кнопки Другая текстура можно загрузить графический файл с изображением дополнительной текстуры.

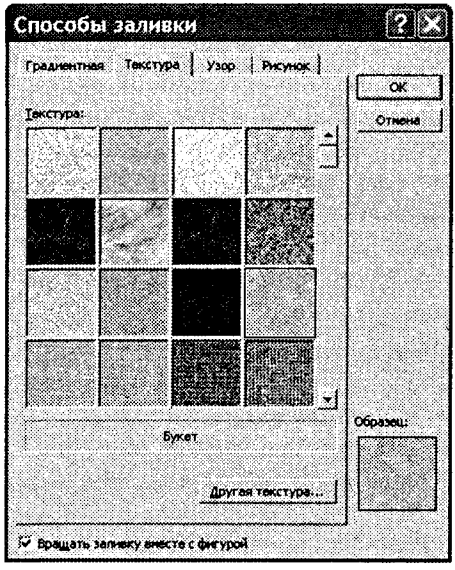


Рис. 11.4. Выбор текстуры для заливки замкнутых контуров

Заливка узором, как и заливка текстурой, — это заливка заранее подготовленным изображением, но имеющим регулярный характер. Выбор узора выполняют на вкладке Узор. Там же можно настроить цвет переднего плана рисунка узора и цвет его фона.

Заливка изображением-картой — это аналог текстурной заливки, при котором замкнутый контур заполняется специально подготовленным графическим изображением. Выбор изображения выполняют выбором файла, в котором оно хранится. Для этого служит вкладка Рисунок.

Взаимодействие объектов друг с другом

Мы рассмотрели, как происходит взаимодействие объектов с текстом и с элементами печатной страницы, но если на одной странице имеется несколько встроенных объектов, то они могут взаимодействовать и друг с другом. Характером этого взаимодействия тоже нужно управлять.

Первое, что нужно решить, — это разрешено ли объектам перекрывать друг друга. Для тех объектов, которым перекрытие разрешено, следует установить флажок Формат объекта ▸ Положение ▸ Дополнительно ▸ Положение объекта ▸ Разрешить перекрытие. Напомним, что доступ к диалоговому окну Формат объекта открыва-

ется командой (для разных объектов она может называться по-разному) контекстного меню объекта.

Управление взаимным положением объектов выполняют с помощью операций:

- группирования;
- задания порядка следования;
- выравнивания;
- распределения.

Группирование объектов. Если на странице представлено несколько объектов и при этом важно строго зафиксировать их взаимное расположение, то их объединяют в один комплексный (групповой) объект с помощью операции группирования. После этой операции свойства группового объекта можно настраивать точно так же, как мы настраивали свойства простейших объектов, — ему может быть задан характер обтекания текстом, метод привязки к абзацу или к элементам печатной страницы и т. п.

Для группирования нескольких объектов их следует выделить (выделение нескольких объектов выполняют при нажатой клавише SHIFT), щелкнуть на любом из объектов группы правой кнопкой мыши и выбрать в контекстном меню команду Группировка ▸ Группировать. Сгруппированные объекты можно перемещать как единое целое. Чтобы разгруппировать объекты и получить доступ к индивидуальным свойствам каждого из них, надо выделить группу и дать команду Группировка ▸ Разгруппировать.

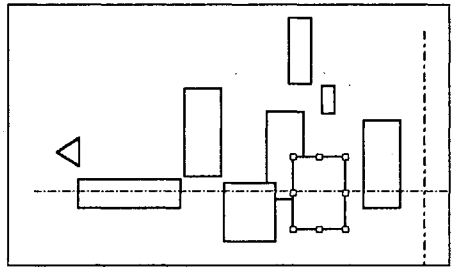
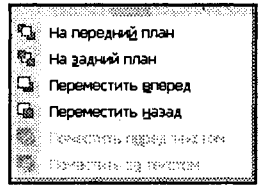


Рис. 11.5. Разгруппированный комплексный объект

Управление порядком следования объектов. Если на странице документа размещается несколько объектов, то предполагается, что у каждого объекта есть свой *слой*. По умолчанию порядок следования слоев связан с порядком создания объектов, то есть те объекты, которые были созданы раньше, лежат на слоях ниже, чем объекты, созданные позже. Если между объектами нет перекрытия, то мы не замечаем, что существует некий порядок следования объектов, однако, когда объекты перекрывают друг друга, этот порядок становится заметен.

Управляют порядком следования объектов с помощью команды Порядок контекстного меню. Она открывает вложенное меню, средствами которого можно поднять объект на передний план, опустить на задний план, сместить на один слой вверх или вниз и задать положение объекта относительно текста.



Выравнивание объектов. Если объекты, составляющие композицию, не перекрывают друг друга, важно иметь средство их относительного выравнивания между собой. Выравнивание объектов выполняют до группирования, ведь после него

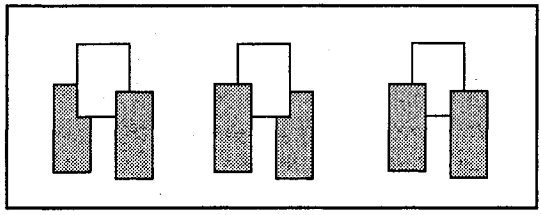


Рис. 11.6. Управление порядком следования

объекты уже нельзя сдвинуть друг относительно друга. В этом случае операция группирования закрепляет взаимное расположение объектов. После нее объекты уже не могут сдвинуться друг относительно друга, и положением всей группы на странице можно управлять как единым целым. Чтобы выполнить выравнивание, необходимо предварительно открыть дополнительную панель инструментов Рисование (Вид ▸ Панели инструментов ▸ Рисование).

Для выравнивания нескольких объектов между собой их следует выделить при нажатой клавише SHIFT, а затем дать команду Действия ▸ Выровнять/распределить (с помощью кнопки Действия панели инструментов Рисование). Существует шесть методов выравнивания. Им соответствуют три команды горизонтального выравнивания (По левому краю, По правому краю, По центру) и три команды выравнивания вертикального (По верхнему краю, По нижнему краю, По середине). Следует обратить внимание на особенность действия команд выравнивания. Так, например, если два объекта выравниваются по *нижнему* полю, значит, они выравниваются по *нижнему* полю *нижнего* объекта. Выравнивание по *правому* полю — это выравнивание по *правому* полю самого *правого* объекта из числа выделенных и так далее. Если необходимо выполнить выравнивание относительно полей страницы, следует предварительно установить флажок меню Действия ▸ Выровнять/распределить ▸ Относительно страницы.

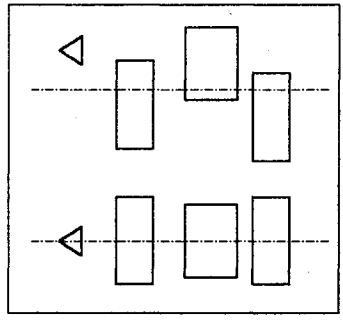


Рис. 11.7. Выравнивание «по середине»

Распределение объектов. Эта операция родственна выравниванию. Ее суть в том, что между объектами устанавливаются равные интервалы по горизонтали или (и) вертикали. Соответственно, в меню команды Действия ▸ Выровнять/распределить имеются команды: Распределить по горизонтали и Распределить по вертикали.

Равномерное распределение объектов обычно выполняют после выравнивания, но, разумеется, до группирования. Нередко объекты выравнивают по вертикали и одновременно равномерно распределяют по горизонтали или, соответственно, наоборот. Дополнительное отличие команд распределения от команд выравнивания заключается еще и в том, что для взаимного выравнивания достаточно иметь два выделенных объекта, а для команд распределения должно быть выделено не менее трех объектов.

11.2. Ввод формул

Необходимость в наличии средства для ввода математических выражений в текстовый документ характерна для научно-технической документации. Одним из таких средств является специальное приложение *Mathcad*, представленное в главе 18. Но функции системы *Mathcad* намного шире, и есть немало оснований для того, чтобы иметь простое средство ввода формул в самом текстовом процессоре.

В программе *Microsoft Word* таким средством является редактор формул *Microsoft Equation* 3.0. Он позволяет создавать формульные объекты и вставлять их в текстовый документ. При необходимости вставленный объект можно редактировать непосредственно в поле документа.

Запуск и настройка редактора формул

Для запуска редактора формул служит команда Вставка ▸ Объект. В открывшемся диалоговом окне Вставка объекта следует выбрать пункт Microsoft Equation 3.0 в списке Тип объекта на вкладке Создание. Откроется панель управления Формула, представленная на рис. 11.8. При этом строка меню текстового процессора замещается строкой меню редактора формул.

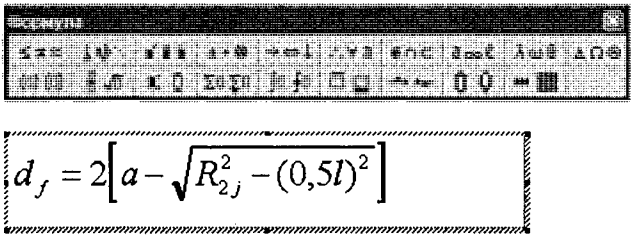


Рис. 11.8. Панель управления редактора формул

Прежде чем пользоваться редактором формул, следует выполнить его настройку. Настройка состоит в назначении шрифтов для различных элементов, входящих в формулы. Она выполняется в диалоговом окне Стили, открываемом командой Стиль ▸ Определить (рис. 11.9). Эта настройка является обязательной — без нее редактор формул работать не будет, но выполнить ее достаточно только один раз.

Прочие (необязательные) настройки редактора формул выполняют в диалоговом окне Интервал (Формат ▸ Интервал). Многочисленные средства настройки, присутствующие в нем, предназначены для задания размеров различных элементов формул.

Панель инструментов редактора формул содержит два ряда кнопок. Кнопки нижнего ряда создают своеобразные шаблоны, содержащие поля для ввода символов. Так, например, для ввода обыкновенной дроби следует выбрать соответствующий шаблон, имеющий два поля: числитель и знаменатель. Заполнение этих полей может производиться как с клавиатуры, так и с помощью элементов управления верхней строки. Переходы между полями выполняются с помощью клавиш управления курсором.

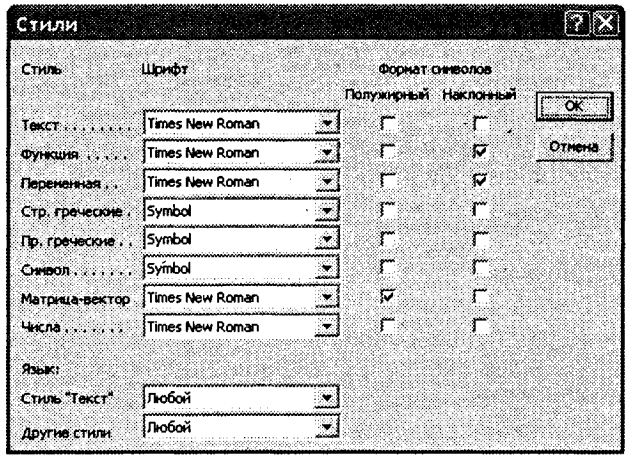


Рис. 11.9. Пример обязательных настроек редактора формул

Ввод и редактирование формул завершается нажатием клавиши ESC или закрытием панели редактора формул. Можно также щелкнуть левой кнопкой мыши где-либо в поле документа вне области ввода формулы. Введенная формула автоматически вставляется в текст в качестве объекта. Далее ее можно переместить в любое иное место документа через буфер обмена (CTRL+X — вырезать; CTRL+V — вставить). Для редактирования формулы непосредственно в документе достаточно выполнить на ней двойной щелчок. При этом автоматически открывается окно редактора формул.

Особенности редактора формул

1. Редактор формул *Microsoft Equation* 3.0 представляет собой отдельный компонент, поэтому при установке текстового процессора требуется специально указать необходимость его подключения.
2. При работе с редактором формул следует стремиться к максимальной полноте вводимых выражений. Так, например, выражение (формула) может содержать компоненты, ввод которых возможен и без использования редактора формул, но для удобства работы и простоты дальнейшего редактирования следует вводить всю формулу целиком только в редакторе формул, не используя иные средства.

$$S=\boxed{\frac{at^2}{2}}$$ — неправильно; $\boxed{S=\frac{at^2}{2}}$ — правильно.

3. При вводе формул и выражений не рекомендуется использовать символы русского алфавита. В тех случаях, когда они необходимы, например, в качестве описательных индексов переменных, им следует назначать стиль Текст.

$$E_{\text{кинет.}}=\frac{mV^2}{2}$$

4. В редакторе формул не работает клавиша ПРОБЕЛ, поскольку необходимые интервалы между символами создаются автоматически. Однако если необходимость ввода пробелов все-таки возникнет, то их можно вводить с помощью кнопки Пробелы и многоточия панели инструментов Формула. Всего предусмотрено пять разновидностей пробелов различной ширины.

11.3. Работа с таблицами

Данные, представленные в табличной форме, отличаются наглядностью. Таблицы всегда были неотъемлемым атрибутом печатной научно-технической документации, а в последние годы стали и эффективным средством оформления *Web*-страниц Интернета. Это связано с тем, что в силу естественных причин возможности форматирования *Web*-страниц весьма ограниченны. Поэтому многие *Web*-дизайнеры используют таблицы (в том числе и скрытые), чтобы принудительно управлять отображением данных на экране клиента и не доверять этот ответственный процесс средству просмотра *Web* (браузеру). Так, например, таблицы — это простейшее средство для имитации на *Web*-странице газетного или журнального текста, имеющего две и более колонок.

Ячейки таблиц могут содержать не только текст, но и графические и прочие объекты. Благодаря этому можно размещать несколько иллюстраций по ширине *Web*-страницы (обычные средства форматирования *Web*-страниц не позволяют это сделать).

При создании страниц можно управлять методом представления ячеек и рамок, как внешних, так и внутренних. При создании печатных документов таблицы оформляют так, чтобы они соответствовали стилю и содержанию документа. При создании *Web*-страниц существует прием, когда рамки вообще не отображают, а между ячейками делают зазор. В результате этого объекты, находящиеся в ячейках, образуют ровные регулярные структуры на экране, в то время как никаких следов таблиц на экране не видно (рис. 11.10).

Текстовый процессор *Microsoft Word* обладает удивительно гибкими и мощными средствами создания таблиц как для печатных, так и для электронных документов.

Три основных средства создания таблиц — это:

- кнопка Добавить таблицу на панели инструментов Стандартная;
- диалоговое окно Вставка таблицы (Таблица ▸ Вставить ▸ Таблица);
- средство рисования таблиц Таблицы и границы (Таблица ▸ Нарисовать таблицу).

Создание таблиц

Кнопку Добавить таблицу используют для создания простейших таблиц небольшого размера. Созданные таким методом таблицы можно в дальнейшем развивать, по мере необходимости увеличивая в них количество строк и столбцов командами меню Таблица ▸ Вставить.

Команду Таблица ▸ Вставить ▸ Таблица используют для создания более сложных таблиц. Она открывает диалоговое окно Вставка таблицы, в котором задают число

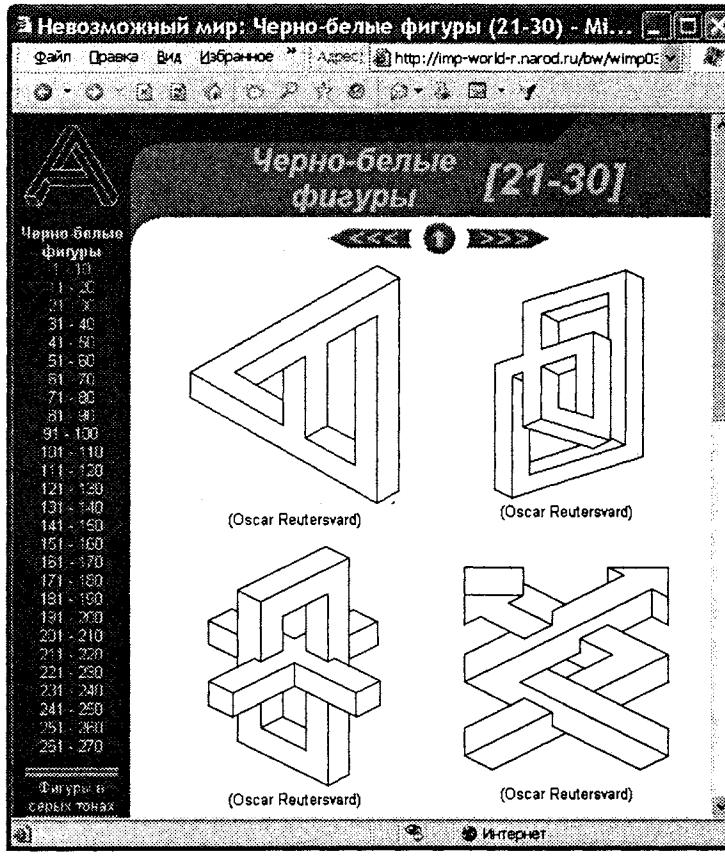


Рис. 11.10. Подобное оформление Web-страниц достигается путем использования таблиц

строк и столбцов, а также ширину столбцов. Если вместо конкретного размера задать параметр Авто, включается режим Автоподбор, благодаря которому столбцы могут эластично форматироваться в соответствии с имеющимся содержанием. Режим автоподбора задают соответствующим переключателем:

- постоянная ширина — общая ширина таблицы равна ширине поля набора документа, а ширина каждого столбца постоянна и зависит от количества столбцов (режим удобен при создании печатных документов);
- по содержимому — ширина каждого столбца пропорциональна объему данных, содержащихся в нем (режим удобен при создании электронных документов, распространяемых в формате текстового процессора);
- по ширине окна — специальный режим для таблиц, размещаемых на *Web*-страницах (окончательное форматирование таблицы происходит не в момент ее создания, а во время просмотра).

Таблицы сложной структуры удобно создавать методом «рисования». Необходимые для этого элементы управления сосредоточены на панели инструментов Таблицы и границы (открывается командой Таблица ▸ Нарисовать таблицу). Порядок действий, необходимых для создания таблиц этим методом, рассмотрен в упражнении 11.1.

Редактирование таблиц

Говоря о редактировании таблиц, мы имеем в виду не редактирование их содержимого, а только редактирование их структуры. Редактирование содержимого осуществляется обычными средствами, рассмотренными в предыдущей главе. Фактически редактирование структуры таблиц сводится к следующим операциям:

- добавление заданного количества строк;
- добавление заданного количества столбцов;
- удаление выделенных ячеек, строк и столбцов;
- слияние выделенных ячеек;
- разбиение выделенных ячеек.

Комбинируя вышеуказанные операции, можно на базе таблиц с простой структурой готовить таблицы, имеющие сложную структуру. Средства для выполнения этих операций находятся в меню Таблица (возможно, потребуется раскрыть *расширенное меню*) или доступны через контекстные меню выделенных объектов.

Форматирование таблиц

При работе с таблицами следует различать *форматирование таблиц* и *форматирование содержимого*. В первом случае происходит управление размерами структурных элементов таблицы (ячеек, строк, столбцов и т. п.), а во втором — управление размещением содержимого ячеек.

Форматирование таблиц можно выполнять в командном или интерактивном режиме. В командном режиме для этой цели используют диалоговое окно Свойства таблицы (Таблица ▸ Свойства таблицы). Его можно открыть и из контекстного меню таблицы, если щелкнуть в ее пределах правой кнопкой мыши. Элементы управления вкладок диалогового окна Свойства таблицы позволяют:

- задать метод выравнивания таблицы относительно страницы документа (Таблица ▸ Свойства таблицы ▸ Таблица ▸ Выравнивание);
- задать метод взаимодействия таблицы с окружающим текстом (Таблица ▸ Свойства таблицы ▸ Таблица ▸ Обтекание);
- определить или переопределить вариант оформления внешних и внутренних рамок таблицы, а также настроить характер оформления ячеек (Таблица ▸ Свойства таблицы ▸ Таблица ▸ Границы и заливка);
- задать размеры внутренних полей в ячейках и интервалы между ячейками (Таблица ▸ Свойства таблицы ▸ Таблица ▸ Параметры);
- назначить параметры текущей строки или выделенных строк (Таблица ▸ Свойства таблицы ▸ Строка);

- назначить параметры текущего столбца или выделенных столбцов (Таблица ▸ Свойства таблицы ▸ Столбец);
- назначить параметры текущей ячейки или выделенных ячеек (Таблица ▸ Свойства таблицы ▸ Ячейка).

В интерактивном режиме таблицу форматируют с помощью маркеров, появляющихся при наведении указателя мыши на таблицу или ее элементы. Маркер в левом верхнем углу таблицы позволяет перемещать таблицу по рабочему полю документа. Маркер в правом нижнем углу позволяет управлять общими размерами таблицы. Маркеры изменения размера, появляющиеся при наведении указателя мыши на рамки таблицы, позволяют интерактивно изменять размеры столбцов и строк методом перетаскивания.

Ввод и форматирование содержимого таблиц

Выделение нужной ячейки для ввода текста выполняют с помощью мыши. Отдельную ячейку выделяют тройным щелчком левой кнопки. Перемещение между ячейками выполняют клавишей TAB (к следующей ячейке) или комбинацией SHIFT+TAB (к предыдущей ячейке). Для навигации по ячейкам таблицы можно также использовать клавиши управления курсором. В тексте курсорные клавиши выполняют перемещение курсора внутри ячейки, но по достижении границы текста они позволяют переходить к соседним ячейкам.

Все команды форматирования текста относятся к выделенному элементу. Выделенным элементом может быть любая ячейка, строка (группа строк), столбец (группа столбцов) или вся таблица в целом. Группы ячеек выделяют методом протягивания мыши. Большинство команд, связанных с форматированием элементов таблицы и содержащихся в них объектов, можно выполнить с помощью панели инструментов Форматирование.

Автоматическое форматирование таблиц

Автоматическое форматирование таблиц выполняют с помощью встроенного средства Автоформат (рис. 11.11), которое запускается командой Таблица ▸ Автоформат таблицы (при наличии выделенной таблицы). Набор предлагаемых форматов представлен в списке Стили таблиц, а результат, получающийся при их использовании, — в поле Образец. Работа по форматированию таблицы полностью автоматизирована и сводится к тому, чтобы выбрать такой формат и так установить сопутствующие элементы управления, чтобы представленный образец наиболее соответствовал запланированному результату.

11.4. Работа с диаграммами

Диаграммы являются удобным средством визуального представления данных и наряду с таблицами очень широко используются в научно-технической документации. Для создания диаграмм текстовый процессор *Microsoft Word* имеет подключаемое средство *Microsoft Graph*. Как и описанный выше редактор формул *Microsoft Equation 3.0*, эта программа является внешним компонентом, и ее установка должна специально заказываться при установке текстового процессора.

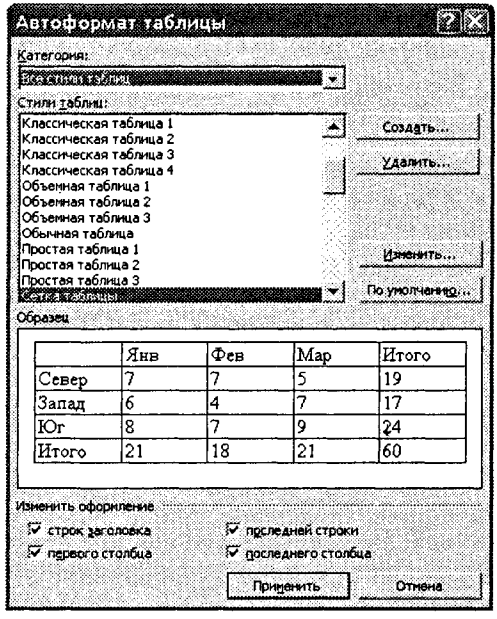


Рис. 11.11. Средство автоматического форматирования таблиц

Текстовый процессор *Microsoft Word XP* предоставляет два метода для вставки диаграмм в документ. Более общий метод основан на том, что сначала в документ вставляется некая произвольная диаграмма, с которой связана некая произвольная *базовая таблица* данных. Далее производится настройка диаграммы, которая состоит в настройке внешнего вида и в редактировании содержания. Поскольку содержание основано на базовой таблице, то оно редактируется путем заполнения этой таблицы нужными данными.

Второй, частный метод основан на том, что диаграмма создается на базе конкретной таблицы, имеющейся в документе. В этом случае настройка диаграммы состоит только в настройке внешнего вида. Этот метод очевидно более удобен, но злоупотреблять им не следует, поскольку данные в таблице и диаграмме дублируют друг друга, а не во всяком документе это оправдано. Приемы создания диаграмм на базе таблиц документа мы рассмотрим в упражнении 11.2.

Создание базовой диаграммы

Создание диаграммы начинается с создания базовой диаграммы командой Вставка▸Объект. В открывшемся диалоговом окне Вставка объекта следует выбрать пункт Microsoft Graph Chart, после чего в документ вставляется диаграмма, с которой связана некая *базовая таблица* (рис. 11.12). Рассматривайте эту таблицу как шаблон. Ее ячейки следует заполнить собственными данными, причем заполнение можно автоматизировать путем импорта данных из какой-либо иной таблицы, например из таблицы *Microsoft Excel*.

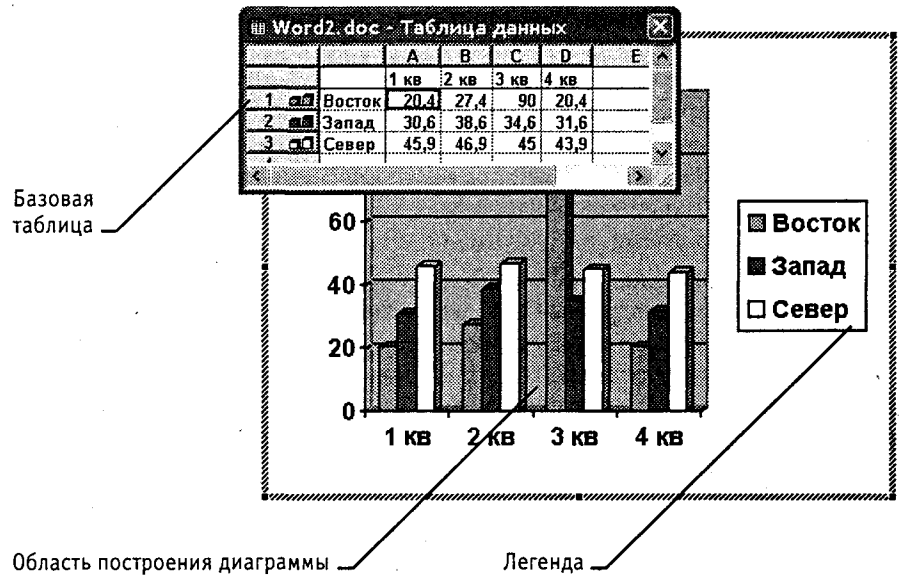


Рис. 11.12. Сначала в документ вставляется произвольная диаграмма и связанная с ней таблица. Далее диаграмма и таблица редактируются по месту

Настройка внешнего вида диаграммы

Существует множество различных типов диаграмм и графиков, отличающихся способом визуального представления связанных с ними данных. Выбор типа диаграммы производят в диалоговом окне Тип диаграммы (Диаграмма ▸ Тип диаграммы), которое имеет пару вкладок (для стандартных и нестандартных типов диаграмм).

Тип диаграммы выбирают в поле Тип, просматривая при этом внешний вид образцов в поле Вид. Выбрав форму диаграммы, приступают к ее настройке. Настройка диаграммы состоит в выборе *элементов оформления диаграммы* и *элементов представления данных* и выполняется в диалоговом окне Параметры диаграммы (Диаграмма ▸ Параметры диаграммы).

Элементы представления данных — это точки на графиках, столбцы гистограмм, секторы круговых диаграмм — в общем, все то, что служит для непосредственного отображения данных. *Элементы оформления* — это название диаграммы, названия ее осей, «легенда» (специальное поле, в котором приведены условные обозначения для групп элементов данных), подписи к элементам данных и линии координатной сетки. Настройку выполняют подключением или отключением тех или иных элементов.

Элементы диаграммы бывают *связанными* или *присоединенными*. Так, например, название диаграммы, названия ее осей и легенду можно редактировать отдельно — это *присоединенные элементы оформления*. Подписи к элементам данных редактировать на диаграмме нельзя — они связаны со значениями в базовой таблице и потому считаются *связанными элементами*.

Для каждого из присоединенных элементов оформления можно выполнить индивидуальное форматирование. Для этого надо в поле диаграммы щелкнуть дважды на поле присоединенного элемента — откроется соответствующее диалоговое окно форматирования (Формат легенды, Формат оси, Формат названия диаграммы, Формат области диаграммы и т. д.). Состав вкладок и других элементов управления этих диалоговых окон зависит от свойств конкретного присоединенного элемента. Так, например, средства форматирования осей диаграммы отличаются от средств форматирования ее названия.

Настройка элементов данных и элементов оформления — это как бы внутренние средства настройки диаграмм. Они определяют свойства диаграммы как объекта. Однако возможно также и редактирование объекта в целом в составе документа. Так, например, для выделенной диаграммы можно с помощью мыши изменять горизонтальный и вертикальный размеры объекта путем перетаскивания маркеров. При изменении размера диаграммы возможно автоматическое перемасштабирование ее элементов оформления.

11.5. Работа с графическими объектами

В документах *Microsoft Word* можно использовать два типа графических объектов: *рисунки* и *изображения*. На русском языке разница между этими терминами неочевидна, и мы поясним, что под ними понимается в текстовом процессоре *Word*. *Рисунки* — объекты векторной природы (линии, прямые и кривые, геометрические фигуры, стандартные и нестандартные). Простейшие средства для их создания есть в самом текстовом процессоре.

Изображения — растровые объекты. Текстовый процессор не имеет средств для их создания, поэтому они вставляются как внешние объекты из файла, подготовленного другими средствами (графическим редактором, с помощью сканера, цифровой камеры, графического планшета).

Рисунки всегда внедрены в документ — их можно редактировать непосредственно по месту. Изображения вставляют в документ методом связывания или внедрения. Их редактирование средствами текстового процессора возможно, но только в ограниченных пределах.

Работа с рисунками

Создание и редактирование рисунков. Для работы с векторными рисунками служит панель инструментов Рисование (Вид ▸ Панели инструментов ▸ Рисование). Основным средством этой панели, предназначенным для создания простейших объектов, является раскрывающийся список Автофигуры. В его категориях представлены заготовки для создания линий, прямых и кривых, простейших геометрических фигур, фигурных стрелок и выносных линий, чертежных элементов для блок-схем и функциональных схем и прочего. При создании и редактировании векторных объектов используют следующие приемы и средства.

1. Векторные объекты создают путем их выбора из категорий списка Автофигуры.
2. Их размер редактируют путем перетаскивания маркеров выделенного объекта в поле документа.

3. Удобным средством, упрощающим создание геометрических фигур, является вспомогательная координатная сетка. Командой Действия ▸ Сетка открывают диалоговое окно Привязка к сетке. В нем задают шаг сетки и способ отображения горизонтальных и вертикальных линий. Флажок Привязать к сетке обеспечивает точное позиционирование узловых точек фигур в узлах координатной сетки. Он удобен, если создаются простые (преимущественно прямолинейные) геометрические фигуры. При редактировании готовых фигур привязка к узлам сетки может создавать неудобства — в этом случае ее отключают или выполняют перемещение объектов при нажатой клавише ALT.
4. Толщина контурной линии и цвет заливки объекта относятся к свойствам объекта. Все свойства объектов можно редактировать в диалоговом окне Формат автофигуры, которое открывают командой Формат ▸ Автофигура, или через контекстное меню объекта, или двойным щелчком на самом объекте. В частности, для управления толщиной и формой контурных линий, а также параметрами заливки служат элементы управления вкладки Цвета и линии данного диалогового окна.
5. Поворотом объекта можно управлять дискретно и непрерывно. Для произвольного поворота фигуры используют команду Действия ▸ Повернуть/отразить ▸ Свободное вращение с панели инструментов Рисование. Для поворота на фиксированный угол значение угла вводят в поле счетчика Поворот на вкладке Размер диалогового окна Формат автофигуры.
6. Взаимодействие рисованного объекта с окружающим текстом может быть достаточно сложным. Так, например, текст может обтекать рисунок по заданной схеме, но он может лежать и поверх рисунка, и под ним. Выбор метода взаимодействия рисунка с текстом выполняют на вкладке Положение в диалоговом окне Формат автофигуры.

Создание надписей в поле рисунка. Рисованные объекты могут содержать текстовые элементы, например заголовки, буквенные или цифровые обозначения на схемах и чертежах. В принципе, необходимые надписи можно создать и основными средствами текстового процессора, но в этом случае очень трудно обеспечить точное положение рисунка относительно связанного с ним текста, особенно если текст не окончателен и может далее редактироваться и форматироваться. Для *Web*-страниц этот метод вообще неприемлем, поскольку они форматируются при каждом просмотре, причем непредсказуемым образом.

Для создания текстовых элементов, присоединенных к автофигурам или рисункам, служит специальное средство Надпись (Вставка ▸ Надпись). Создав автофигуру, рядом создают элемент Надпись. В поле надписи вводят необходимый текст, после чего надпись можно редактировать. Ее размер подгоняют под размер содержащегося в ней текста перетаскиванием маркеров. Прочие свойства надписи задают в диалоговом окне Формат надписи, которое для выделенной надписи открывают командой Формат ▸ Надпись. Элементы управления, представленные на вкладках этого окна, позволяют настроить:

- фоновый цвет (если задать параметр Нет заливки, надпись будет лежать на прозрачном фоне);

- цвет, тип и толщину обрамляющих линий (если при выборе цвета задать параметр Нет линий, то прочие параметры не имеют смысла);
- размеры внутренних полей между текстом и внешней рамкой поля Надпись (назначаются на вкладке Надпись).

Создав объект Надпись, его можно сгруппировать с рисунком, и тогда они будут представлять цельную композицию.

Для автофигур есть особое средство создания текстового оформления — текст может размещаться в поле автофигуры. Это выполняют командой Добавить текст в контекстном меню автофигуры. Если текст слишком велик, можно либо изменить размер автофигуры путем перетаскивания ее маркеров, либо изменить формат текста, уменьшив размер шрифта средствами панели Форматирование. Этот прием используют при создании блок-схем и функциональных схем устройств.

Работа с клипартами. Создание достаточно сложных композиций может быть очень трудоемким. В таких случаях используют готовые библиотеки (*коллекции*) рисунков (*клипартов*), в том числе и тематических. Такие библиотеки распространяются на отдельных компакт-дисках, их можно найти в Интернете, но базовая, простейшая коллекция может быть установлена вместе с текстовым процессором — она входит в комплект поставки пакета *Microsoft Office*.

Для вставки клипартов используют команду Вставка ▸ Рисунок ▸ Картинки. Соответствующая кнопка (Добавить картинку) имеется и на панели инструментов Рисование. При этом открывается Область задач в режиме Вставка картинки. Это название достаточно условное, поскольку клипарт — понятие расширенное. К клипартам относят не только графические объекты, но и звуковые клипы и видеоклипы — их тоже можно вставить в документ с помощью этого средства.

Для поиска графических клипартов раскройте список Искать объекты и оставьте флажки только в нужных категориях. Затем щелкните на кнопке Найти. На панели появятся изображения всех найденных клипартов (рис. 11.13). Разыскав нужный клипарт, его можно вставить в документ простым щелчком.

При работе с клипартами следует иметь в виду, что подобрать именно тот клипарт, который наилучшим образом соответствует характеру документа, можно далеко не всегда. Поэтому клипарты следует рассматривать не как готовые средства оформления, а как заготовки для их создания. Клипарты — это композиционные объекты. Их можно «разбирать» на составляющие, редактировать их элементы по отдельности, создавать композиции из объектов, взятых из разных клипартов. Все это выполняется путем редактирования клипартов, вставленных в документ.

Обычный порядок редактирования клипартов — следующий:

- клипарт выделяют щелчком левой кнопки мыши;
- открывают его контекстное меню щелчком правой кнопки;
- в контекстном меню выбирают команду Изменить рисунок — он открывается в режиме редактирования;
- в этом режиме работают с отдельными объектами, составляющими рисунок.

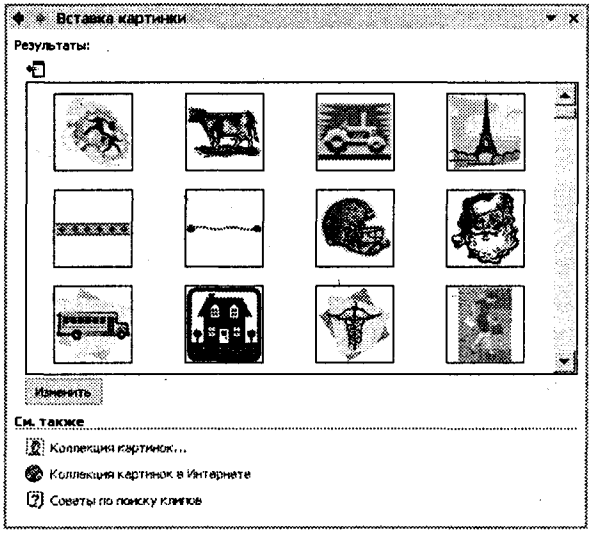


Рис. 11.13. Поиск и вставка клипартов

При работе с объектами клипарта используют команды разгруппировки и изменения порядка. Если из сложной композиции надо выделить один составляющий объект, то простейший прием состоит не в том, чтобы выделить все элементы, которые в него входят, а в том, чтобы удалить те, которые в него не входят. После каждого из удалений можно подавать отменяющую команду CTRL+Z, проверяя, что изменилось в составе рисунка. Если изменения желательны, их восстанавливают командой CTRL+Y, а если нет — переходят к выбору и удалению других элементов.

Комбинирование объектов, принадлежащих разным клипартам, выполняют путем копирования через буфер обмена *Windows* (CTRL+C и CTRL+V). При создании новых объектов из готовых клипартов часто приходится изменять размер итогового рисунка. Простейший способ для этого — воспользоваться кнопкой Подобрать размер на панели инструментов Полотно. При этой операции происходит подгонка границ рисунка по размеру содержимого.

Специальные средства оформления. Эти средства оформления представлены кнопками на панели инструментов Рисование. Они позволяют:

- управлять цветом заливки, цветом контура и цветом текста;
- управлять толщиной сплошных линий и параметрами штриха для штриховых линий;
- преобразовывать линии в стрелки и управлять формой их концов;
- создавать теневые эффекты;
- создавать трехмерные эффекты.

Для каждой из указанных кнопок открывается палитра, позволяющая настроить результат действия эффекта. Если к объекту применен теневой или трехмерный эффект, то редактировать результат этого эффекта непосредственно в поле доку-

мента нельзя, поскольку в отличие от контуров плоских объектов контуры трехмерных эффектов не являются объектами и не имеют управляющих маркеров. Поэтому для объектов, имеющих теневое или трехмерное оформление, используют иные приемы редактирования:

- выделяют объект в поле документа;
- используют кнопку Тень или Объем на панели инструментов Рисование;
- в открывшейся палитре выбирают элемент управления Настройка тени или Настройка объема;
- при этом открывается одноименная панель инструментов, посредством которых и редактируют специальные объекты.

Работа с изображениями

Под *изображениями* понимаются растровые графические объекты, исполненные посторонними программными средствами или полученные из внешнего источника. Они вставляются в документ методом связывания или внедрения. Общая команда для вставки таких объектов — Вставка ▸ Рисунок ▸ Из файла. По этой команде открывается стандартное диалоговое окно Добавление рисунка, в котором и производится выбор файла, содержащего изображение.

Выбор метода вставки. В текстовом процессоре *Microsoft Word XP* избранный рисунок можно вставить в документ тремя способами: *внедрением*, *связыванием* и *внедрением со связыванием*.

1. В первом случае объект войдет в документ и может передаваться вместе с ним.
2. Во втором случае он останется по месту своего хранения, а в документ войдет только указатель на первоисточник.
3. В третьем случае объект войдет в документ, но его связь с первоисточником сохранится. Это полезно, если предполагается возможность редактирования первоисточника и надо обеспечить синхронное редактирование и внедренного объекта.

Выбор метода вставки выполняют в диалоговом окне Добавление рисунка. В его правом нижнем углу есть раскрывающийся список, в котором следует выбрать один метод из трех возможных.

Изменение метода вставки. Если в качестве метода вставки было избрано внедрение, то ничего изменить уже нельзя. Пользователь документа, в который внедрено изображение, естественным образом лишен доступа к оригиналу. Если же при вставке был использован один из двух методов, подразумевающих связь с оригиналом, то метод изменить можно.

При выделении объекта, имеющего связь с оригиналом, в меню Правка активизируется пункт Связи, открывающий диалоговое окно Связи (рис. 11.14).

Элементы управления этого диалогового окна позволяют:

- обновить связь (если оригинал изменился);
- разорвать связь (и перейти к хранению объекта в документе);

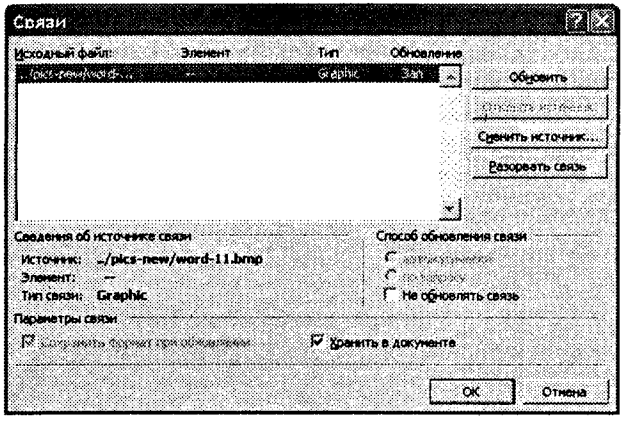


Рис. 11.14. Диалоговое окно Связи

- сменить источник (установить связь с другим объектом или с тем же объектом, но хранящимся в другом месте);
- перейти к методу одновременного внедрения и связывания путем установки флажка Хранить в документе.

Взаимодействие изображения с текстом. Основная часть инструментов для настройки свойств изображений в текстовом документе сосредоточена на панели инструментов Настройка изображения (Вид ▸ Панели инструментов ▸ Настройка изображения). Как правило, при выборе рисунка в тексте документа эта панель открывается автоматически.

По способу взаимодействия с текстом выделяют два основных типа изображений: *внедренные в строку* (*inline*) и *свободные* (*floating*). Изображения первого типа можно условно рассматривать как отдельные символы: при движении текста в процессе редактирования изображение перемещается вместе с ним и остается в том месте текста, куда его поместили. Положение свободного изображения на странице не связано с позицией ввода. Изображение взаимодействует с текстом посредством обтекания.

Для управления методом взаимодействия изображения с текстом служит вкладка Положение в диалоговом окне Формат рисунка, которое открывают командой Формат ▸ Рисунок или кнопкой Формат рисунка на панели инструментов Настройка изображения. Элемент управления В тексте обеспечивает внедрение изображения в текстовую строку. Прочие элементы служат для выбора одного из методов обтекания. Если изображение вставлено в документ как свободное, дополнительные средства настройки обтекания можно получить из меню, которое открывается кнопкой Обтекание текстом на панели инструментов Настройка изображения. В частности, здесь присутствует пункт Изменить контур обтекания, который позволяет создавать интересные варианты обтекания изображения по криволинейному контуру.

Приемы редактирования изображения. В текстовом процессоре *Microsoft Word XP* имеются два средства редактирования встроенного растрового изображения.

Первое средство — внутреннее, а второе — внешнее, подключаемое при установке процессора. Внутреннее средство представлено элементами управления панели инструментов Настройка изображения (Вид ▸ Панели инструментов ▸ Настройка изображения). Внешним средством редактирования изображений является редактор *Microsoft Photo Editor* 3.0. Он должен быть подключен при установке *Microsoft Word XP* точно так же, как редактор формул *Microsoft Equation* 3.0 и редактор диаграмм и графиков *Microsoft Chart*.

Внутреннее средство редактирования изображений имеет относительно малые возможности, и, если говорить строго, его не вполне корректно считать средством редактирования изображений. При его использовании оригинал изображения не меняется, а меняется только способ его отображения в документе. Фактически здесь редактируется не изображение, а фильтр, управляющий тем, как оно выглядит в документе.

На панели инструментов Настройка изображения средства настройки изображения представлены следующими кнопками:

- Увеличить контрастность;
- Уменьшить контрастность;
- Увеличить яркость;
- Уменьшить яркость;
- Обрезка;
- Установить прозрачный цвет.

Функция установки прозрачного цвета имеет особое значение для создания *Web*-страниц. Она позволяет назначить один (любой) из цветов изображения в качестве «прозрачного». При размещении такого графического объекта поверх других объектов (это выполняется настройкой метода обтекания) все объекты нижележащего слоя видны через те участки верхнего изображения, которые имеют цвет, назначенный прозрачным. Разумеется, изображения, используемые для такого представления, надо готовить особо. Они должны иметь большие участки, окрашенные однородным фоновым цветом. Для этого изображение либо предварительно обрабатывают в графическом редакторе, либо сразу снимают цифровой фотокамерой на однородном фоне (как правило, синего цвета).

Внешнее средство редактирования изображений (редактор *Microsoft Photo Editor 3.0*) рассчитано на изменение файла оригинала и потому применимо только к изображениям, внедренным в документ, но не связанным. Более того, вставку изображения в документ в этом случае надо выполнять не как обычно (Вставка ▸ Рисунок ▸ Из файла), а другим способом — Вставка ▸ Объект ▸ Microsoft Photo Editor 3.0 Photo. При этом открывается окно создания нового изображения Создание рисунка, в котором следует включить переключатель Открыть имеющийся.

Заранее подготовленное изображение открывается из файла и может редактироваться средствами редактора *Microsoft Photo Editor* 3.0. По окончании редактирования окно редактора закрывают, и изображение автоматически встраивается в текстовый документ. Если в дальнейшем потребуется продолжить его редактирование,

то при двойном щелчке на объекте изображение откроется непосредственно в редакторе *Microsoft Photo Editor 3.0*.

Практическое занятие

Упражнение 11.1. Создание сложных таблиц методом рисования

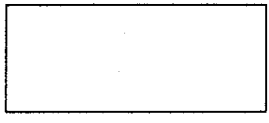
На рис. 11.15 представлен фрагмент технологической карты механической обработки детали. По своей сути технологическая карта является табличной формой сложной структуры. В данном упражнении мы рассмотрим процесс ее создания средствами текстового процессора *Microsoft Word*.

Переход	Содержание перехода	Инструмент (код и наименование)			Режим обработки					T_o	$T_в$
		вспомогательный	режущий	измерительный	T	i	S	n	V		
А											
1											
2											
3											

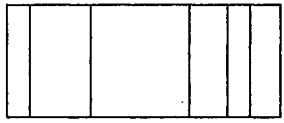
Рис. 11.15. Фрагмент карты механической обработки детали

1. Запустите текстовый процессор.
2. Создайте новый документ на базе обычного шаблона.
3. В качестве режима представления документа включите Режим разметки (Вид ▸ Разметка страницы), чтобы четко видеть границы полосы набора.
4. Откройте панель инструментов Таблицы и границы (Вид ▸ Панели инструментов ▸ Таблицы и границы).
5. Выберите инструмент Нарисовать таблицу.
6. Методом протягивания нарисуйте с его помощью прямоугольник, ширина которого равна ширине полосы набора. Высота прямоугольника может быть произвольной — его можно будет растянуть или сжать впоследствии. Для этого достаточно навести указатель мыши на нижнюю границу рамки и, когда указатель сменит форму, переместить рамку методом перетаскивания.

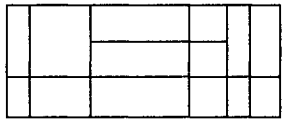
   Полученный прямоугольник представляет собой внешнюю границу таблицы. Для прочих границ она будет *опорной*, то есть они должны начинаться и заканчиваться на опорной границе.



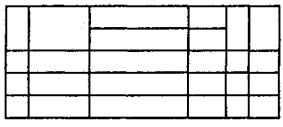
7. Проведите пять вертикальных линий. Это внутренние границы. Они опираются на внешние границы. Для горизонтальных границ, которые будут на них опираться, они будут выполнять функции опорных. На ширину столбцов не обращайте внимания — ее можно будет изменить впоследствии. Сейчас мы разрабатываем только структуру таблицы.



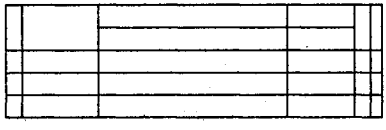
8. Убедитесь, что с помощью инструмента Ластик можно удалить любую из только что проведенных границ. Удаление выполняется одним щелчком. Внешние границы удалить нельзя.
9. Проведите две горизонтальные линии, как показано на рисунке.



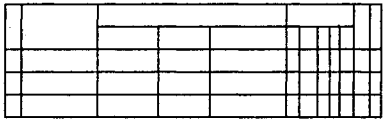
10. Убедитесь с помощью Ластика в том, что вертикальные линии, ставшие опорными для первой горизонтальной линии, не могут быть удалены.
11. Выделите всю таблицу. Для этого введите в нее указатель мыши и дайте команду Таблица ▸ Выделить ▸ Таблица.
12. Когда таблица выделена, можно задать высоту ее строк элементом управления Таблица ▸ Свойства таблицы ▸ Строка ▸ Высота. Добавьте в нижней части таблицы несколько строк командой Таблица ▸ Вставить ▸ Строки ниже. При необходимости впоследствии можно добавить столько строк, сколько надо.



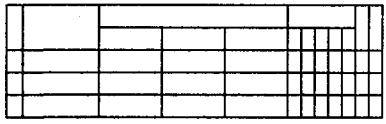
13. Методом перетаскивания вертикальных границ создайте нужное соотношение между шириной столбцов.



14. Проведите дополнительные вертикальные линии инструментом Нарисовать таблицу.



15. Выделите группы столбцов, которые должны иметь равную ширину. Для этого установите указатель мыши над верхней рамкой таблицы и в тот момент, когда он примет форму стрелки, направленной вниз, щелкните левой кнопкой.
16. Выделенные столбцы станут равными по ширине, если щелкнуть на кнопке Выровнять ширину столбцов на панели инструментов Таблицы и границы.



17. Если необходимо выровнять высоту строк, их следует выделить и использовать кнопку Выровнять высоту строк.
18. Заполните заголовки столбцов таблицы. Гарнитуру шрифта, его размер и начертание задайте с помощью инструментов панели Форматирование.
19. Обратите внимание на то, что в ячейках таблицы имеет значение не только горизонтальное выравнивание, но и вертикальное, поэтому для задания выравнивания заголовков средств панели Форматирование недостаточно. Нужный метод выравнивания (один из девяти) выбирают в палитре, которая открывается щелчком на раскрывающей кнопке Выравнивание в ячейке в центре панели Таблицы и границы.
20. При вводе заголовка первого столбца в образце использовано вертикальное расположение текста. Это типичный прием для оформления заголовков узких столбцов. Изменение направления текста выполняют с помощью кнопки Изменить направление текста на панели инструментов Таблицы и границы.
21. Завершив создание таблицы, сохраните документ *Word* в папке \Мои документы.

Мы научились создавать таблицы сложной структуры методом «рисования» и использовать автоматические средства управления шириной столбцов, высотой ячеек и их выравниванием.

Упражнение 11.2. Создание диаграмм на основе таблиц

Ниже представлена таблица с итогами испытания на износ образцов легированных сталей при трении скольжения под нагрузкой в условиях недостаточной смазки. Замеры величины износа образца производились восемь раз через каждые пятнадцать минут.

Пара трения	Износ верхнего образца, мг							
	15 мин	30 мин	45 мин	60 мин	75 мин	90 мин	105 мин	120 мин
40Х13/95Х18	11,2	7,6	4,2	1,8	1,1	1,2	1,1	1,2
40Х13/40ХН	17,4	12,5	9,5	7,4	5,3	4,8	4,5	4,4
40ХН/95Х18	12,1	6,4	3,1	2,2	1,7	1,6	1,6	1,6

В этом упражнении мы построим диаграмму на базе данной таблицы.

1. Запустите текстовый процессор.
2. Создайте новый документ на базе стандартного шаблона.
3. В качестве режима представления документа включите Режим разметки (Вид ▸ Разметка страницы), чтобы четко видеть границы полосы набора.
4. Командой Таблица ▸ Вставить ▸ Таблица создайте базовую таблицу, имеющую 5 строк и 9 столбцов.
5. Выделите две верхние ячейки первого столбца и объедините их командой Таблица ▸ Объединить ячейки.
6. Выделите ячейки первой строки для столбцов со второго по девятый и объедините их.
7. Заполните таблицу согласно прилагаемому образцу.
8. Установите указатель мыши в поле таблицы и выделите таблицу командой Таблица ▸ Выделить ▸ Таблица. Скопируйте выделенную таблицу в буфер обмена (Правка ** Копировать).
9. Вставьте базовую диаграмму командой Вставка ▸ Объект ▸ Microsoft Graph Chart. Рядом с диаграммой развернется ее базовая таблица.
10. Выделите содержимое базовой таблицы диаграммы щелчком на ячейке, образованной на пересечении заголовков строк и столбцов в левом верхнем углу.
11. Замените содержимое базовой таблицы содержимым своей таблицы командой вставки содержимого из буфера обмена (Правка ▸ Вставить).
12. Обратите внимание на то, как изменилась диаграмма: она пришла в соответствие с содержимым таблицы.
13. На диаграмме выделите область построения. Щелкните правой кнопкой мыши и в контекстном меню выберите пункт Тип диаграммы. Средствами открывшегося диалогового окна проверьте, как выглядят диаграммы других (стандартных и нестандартных) типов.
14. Закройте диалоговое окно Тип диаграммы. Сохраните документ *Word* в папке \Мои документы.

В этом упражнении мы освоили один из двух основных методов создания диаграмм — метод, основанный на использовании базовой таблицы, которая содержащится в документе.

Упражнение 11.3. Изучение эффективных приемов работы с графическими объектами

1. Запустите текстовый процессор.
2. Создайте новый документ на базе стандартного шаблона.
3. В качестве режима представления документа включите Режим разметки (Вид ▸ Разметка страницы), чтобы четко видеть границы полосы набора.
4. Введите несколько строк произвольного текста.

5. Командой Вставка ▸ Рисунок ▸ Из файла вставьте ниже текста рисунок из произвольного файла, например из файла \Windows\Японский мотив.bmp.
6. Выделите рисунок щелчком левой кнопки мыши — откроется панель инструментов Настройка изображения. Используя кнопку Формат рисунка, откройте одноименное диалоговое окно.
7. На вкладке Положение выберите вариант размещения В тексте. Передвиньте изображение методом перетаскивания, оценивая происходящее взаимодействие с текстом.
8. На вкладке Положение диалогового окна Формат рисунка выберите вариант размещения По контуру. Проверьте, как происходит взаимодействие с текстом при перемещении изображения.
9. Выделите изображение, скопируйте его в буфер обмена (CTRL+C) и создайте рядом его копию (CTRL+V).
10. Перемещая оба изображения, добейтесь их положения рядом, с выравниванием по верхнему краю.
11. Повторите перемещение изображений с выравниванием при нажатой клавише ALT. Убедитесь в том, что перемещение изображений происходит дискретно, с привязкой к узлам невидимой сетки, что позволяет выполнить выравнивание абсолютно точно.
12. Выделите одно из изображений. Используя угловой маркер, измените его размер методом перетаскивания.
13. Восстановите прежний размер изображения.
14. Повторите перетаскивание углового маркера, но при нажатой клавише CTRL. Обратите внимание на то, что характер изменения размера изображения изменился. В данном случае оно перемасштабируется «от центра».
15. Сохраните итоговый документ *Word* в папке \Мои документы.

Мы освоили два основных приема вставки изображения в текст — с внедрением в строку и со свободным размещением. Мы убедились, что использование клавиш CTRL и ALT при работе с изображениями в документе открывает дополнительные возможности оформления.

Упражнение 11.4. Создание графических заголовков

15 мин

Для создания художественных графических надписей, например заголовков, текстовый процессор *Microsoft Word XP* имеет специальное программное средство *WordArt*. Доступ к нему осуществляется двумя способами: либо через панель инструментов WordArt (Вид ▸ Панели инструментов ▸ WordArt), либо с помощью кнопки Добавить объект WordArt на панели инструментов Рисование.

Графические объекты, вставленные в текстовый документ средством *WordArt*, могут распечатываться вместе с документом на выводном печатающем устройстве, могут отображаться в составе электронного документа, распространяемого в формате *Microsoft Word*, и могут отображаться на *Web*-страницах. Однако при экспорте доку-

мента в форматы других программ, предназначенных для обработки документов, объекты *WordArt* не всегда воспроизводятся правильно, то есть при создании документов, в которых содержание играет более высокую роль, чем оформление, использовать художественные заголовки, выполненные средствами *WordArt*, не рекомендуется.

1. Запустите текстовый процессор.
2. Создайте новый документ на базе стандартного шаблона.
3. В качестве режима представления документа включите Режим разметки (Вид ▸ Разметка страницы), чтобы четко видеть границы полосы набора.
4. Введите несколько строк произвольного текста.
5. Командой Вид ▸ Панели инструментов ▸ WordArt включите отображение панели инструментов WordArt.
6. Щелкните на кнопке Добавить объект WordArt — произойдет запуск мастера создания объекта *WordArt*.
7. В окне Коллекция WordArt выберите желаемый стиль оформления надписи.
8. В диалоговом окне Изменение текста WordArt выберите желаемый шрифт, его размер, начертание и введите текст создаваемого заголовка (надписи).
9. После щелчка на кнопке OK произойдет вставка созданного объекта в текущий документ *Microsoft Word*.
10. Дальнейшее управление формой и расположением созданного объекта выполняют элементами управления панели инструментов WordArt. Проверьте, как протекают следующие операции (после каждой команды возвращайтесь к исходному состоянию комбинацией CTRL+Z):
    - изменение содержания надписи (Изменить текст);
    - изменение стиля оформления (Коллекция WordArt);
    - изменение характера взаимодействия с основным текстом (Формат объекта ▸ Положение);
    - изменение формы надписи (Форма WordArt);
    - выравнивание букв надписи по высоте (Выровнять буквы WordArt по высоте);
    - расположение текста надписи по вертикали (Вертикальный текст WordArt);
    - управление интервалом между символами (Межсимвольный интервал WordArt).
11. Закончив эксперименты, создайте заголовок по своему вкусу и сохраните документ *Word* в папке \Мои документы.

Мы научились создавать художественные заголовки, внедрять их в документы и редактировать «по месту». В то же время мы узнали, что для документов, передаваемых на последующую обработку, пользоваться этим средством не рекомендуется.

Глава 12. Обработка данных средствами электронных таблиц

Для представления данных в удобном виде используют таблицы. Компьютер позволяет представлять их в электронной форме, а это дает возможность не только отображать, но и обрабатывать данные. Класс программ, используемых для этой цели, называется *электронными таблицами*.

Особенность электронных таблиц заключается в возможности применения формул для описания связи между значениями различных ячеек. Расчет по заданным формулам выполняется автоматически. Изменение содержимого какой-либо ячейки приводит к пересчету значений всех ячеек, которые с ней связаны формульными отношениями и, тем самым, к обновлению всей таблицы в соответствии с изменившимися данными.

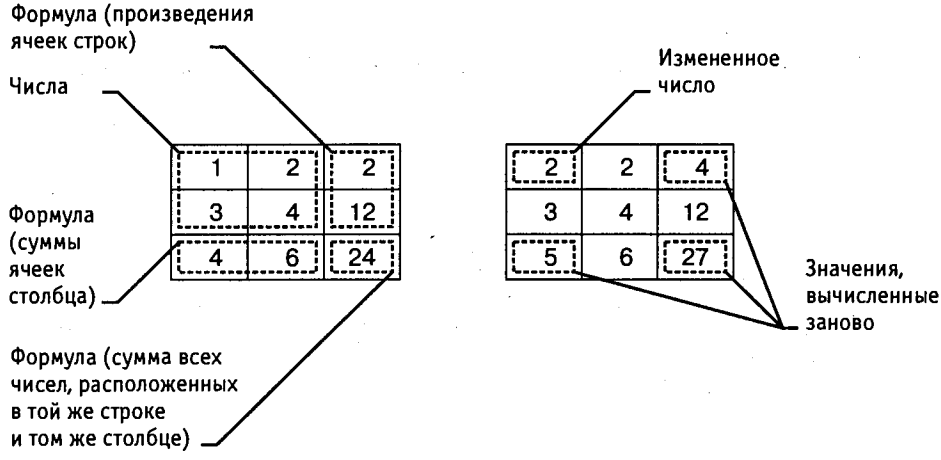


Рис. 12.1. При изменении содержания одной из ячеек таблицы все формулы пересчитываются и значения в ячейках, которые прямо или косвенно зависят от измененных, автоматически обновляются

Применение электронных таблиц упрощает работу с данными и позволяет получать результаты без проведения расчетов вручную или специального программирования. Наиболее широкое применение электронные таблицы нашли в экономических и бухгалтерских расчетах, но и в научно-технических задачах электронные таблицы можно использовать эффективно, например для:

- проведения однотипных расчетов над большими наборами данных;
- автоматизации итоговых вычислений;
- решения задач путем подбора значений параметров, табулирования формул;
- обработки результатов экспериментов;
- проведения поиска оптимальных значений параметров;
- подготовки табличных документов;
- построения диаграмм и графиков по имеющимся данным.

Одним из наиболее распространенных средств работы с документами, имеющими табличную структуру, является программа *Microsoft Excel.*

12.1. Основные понятия электронных таблиц

Программа *Microsoft Excel* предназначена для работы с таблицами данных, преимущественно числовых. При формировании таблицы выполняют ввод, редактирование и форматирование текстовых и числовых данных, а также формул. Наличие средств автоматизации облегчает эти операции. Созданная таблица может быть выведена на печать.

Рабочая книга и рабочий лист. Строки, столбцы, ячейки

Документ *Excel* называется *рабочей книгой.* Рабочая книга представляет собой набор *рабочих листов,* каждый из которых имеет табличную структуру и может содержать одну или несколько таблиц. В окне документа в программе *Excel* отображается только *текущий* рабочий лист, с которым и ведется работа (рис. 12.2). Каждый рабочий лист имеет *название,* которое отображается на ярлычке *листа,* отображаемом в его нижней части. С помощью ярлычков можно переключаться к другим рабочим листам, входящим в ту же самую рабочую книгу. Чтобы переименовать рабочий лист, надо дважды щелкнуть на его ярлычке.

Рабочий лист состоит из *строк* и *столбцов.* Столбцы озаглавлены прописными латинскими буквами и, далее, двухбуквенными комбинациями. Всего рабочий лист может содержать до 256 столбцов, пронумерованных от A до IV. Строки последовательно нумеруются цифрами, от 1 до 65 536 (максимально допустимый номер строки).

Ячейки и их адресация. На пересечении столбцов и строк образуются *ячейки* таблицы. Они являются минимальными элементами для хранения данных. Обозначение отдельной ячейки сочетает в себе номера столбца и строки (в этом порядке), на пересечении которых она расположена, например: A1 или DE234. Обозначение ячейки (ее номер) выполняет функции ее адреса. Адреса ячеек используются при

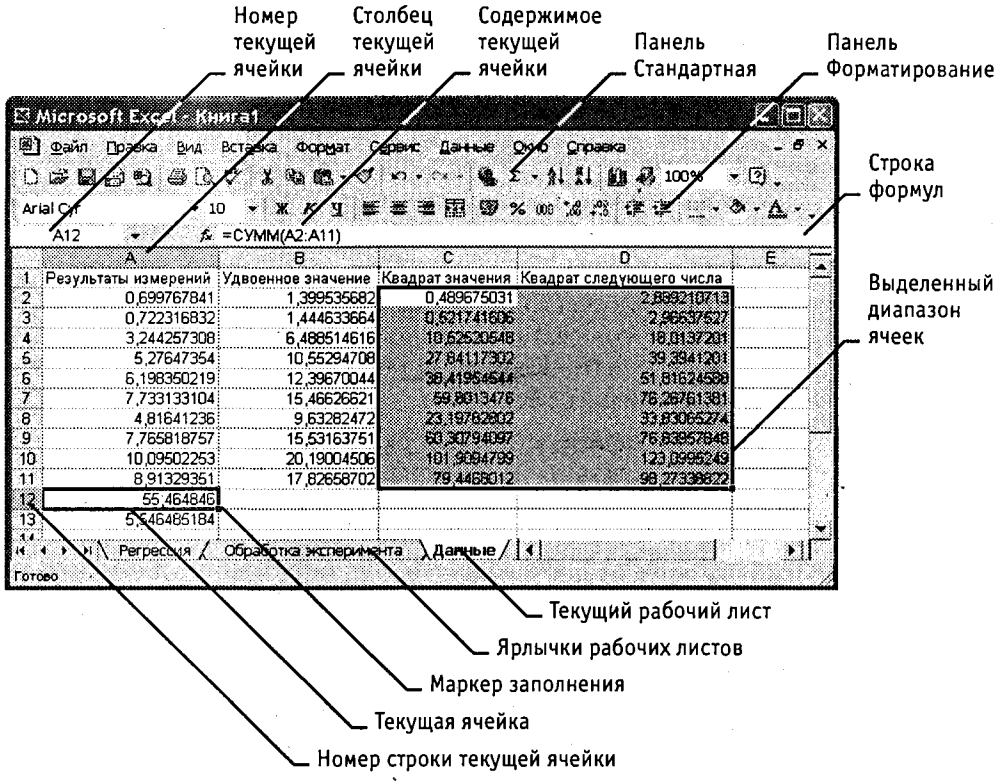


Рис. 12.2. Рабочий лист электронной таблицы Excel

записи формул, определяющих взаимосвязь между значениями, расположенными в разных ячейках.

Одна из ячеек всегда является *активной* и выделяется *рамкой активной ячейки*. Эта рамка в программе *Excel* играет роль курсора. Операции ввода и редактирования всегда производятся в активной ячейке. Переместить рамку активной ячейки можно с помощью курсорных клавиш или указателя мыши.

Диапазон ячеек. На данные, расположенные в соседних ячейках, можно ссылаться в формулах как на единое целое. Такую группу ячеек называют *диапазоном*. Наиболее часто используют прямоугольные диапазоны, образующиеся на пересечении группы последовательно идущих строк и группы последовательно идущих столбцов. Диапазон ячеек обозначают, указывая через двоеточие номера ячеек, расположенных в противоположных углах прямоугольника, например: A1:C15.

Если требуется выделить прямоугольный диапазон ячеек, это можно сделать протягиванием указателя от одной угловой ячейки до противоположной по диагонали. Рамка текущей ячейки при этом расширяется, охватывая весь выбранный диапазон. Чтобы выбрать столбец или строку целиком, следует щелкнуть на заголовке

столбца (строки). Протягиванием указателя по заголовкам можно выбрать несколько идущих подряд столбцов или строк.

Ввод, редактирование и форматирование данных

Отдельная ячейка может содержать данные, относящиеся к одному из трех типов: *текст*, *число* или *формула*, — а также оставаться пустой. Программа *Excel* при сохранении рабочей книги записывает в файл только прямоугольную область рабочих листов, примыкающую к левому верхнему углу (ячейка A1) и содержащую все заполненные ячейки.

Тип данных, размещаемых в ячейке, определяется автоматически при вводе. Если эти данные можно интерпретировать как число, программа *Excel* так и делает. В противном случае данные рассматриваются как текст. Ввод формулы всегда начинается с символа «=» (знака равенства).

Ввод текста и чисел. Ввод данных осуществляют непосредственно в текущую ячейку или в *строку формул*, располагающуюся в верхней части окна программы под панелями инструментов (см. рис. 12.2). Место ввода отмечается текстовым курсором. Если начать ввод нажатием алфавитно-цифровых клавиш, данные из текущей ячейки заменяются вводимым текстом. Если щелкнуть на строке формул или дважды на текущей ячейке, старое содержимое ячейки не удаляется и появляется возможность его редактирования. Вводимые данные в любом случае отображаются как в ячейке, так и в строке формул.

Чтобы завершить ввод, сохранив введенные данные, используют кнопку Ввод в строке формул или клавишу ENTER. Чтобы отменить внесенные изменения и восстановить прежнее значение ячейки, используют кнопку Отмена в строке формул или клавишу ESC. Для очистки текущей ячейки или выделенного диапазона проще всего использовать клавишу DELETE.

Форматирование содержимого ячеек. Текстовые данные по умолчанию выравниваются по левому краю ячейки, а числа — по правому. Чтобы изменить формат отображения данных в текущей ячейке или выбранном диапазоне, используют команду Формат ▸ Ячейки. Вкладки этого диалогового окна позволяют выбирать формат записи данных (количество знаков после запятой, указание денежной единицы, способ записи даты и прочее), задавать направление текста и метод его выравнивания, определять шрифт и начертание символов, управлять отображением и видом рамок, задавать фоновый цвет.

12.2. Содержание электронной таблицы

Формулы

Вычисления в таблицах программы *Excel* осуществляются при помощи *формул*. Формула может содержать числовые константы, ссылки на ячейки и *функции Excel*, соединенные знаками математических операций. Скобки позволяют изменять стандартный порядок выполнения действий. Если ячейка содержит формулу, то в рабочем листе отображается текущий результат вычисления этой формулы. Если сделать ячейку текущей, то сама формула отображается в строке формул.

Правило использования формул в программе *Excel* состоит в том, что, если значение ячейки *действительно* зависит от других ячеек таблицы, *всегда* следует использовать формулу, даже если операцию легко можно выполнить в «уме». Это гарантирует, что последующее редактирование таблицы не нарушит ее целостности и правильности производимых в ней вычислений.

Ссылки на ячейки

Формула может содержать *ссылки*, то есть адреса ячеек, содержимое которых используется в вычислениях. Это означает, что результат вычисления формулы зависит от числа, находящегося в другой ячейке. Ячейка, содержащая формулу, таким образом, является *зависимой*. Значение, отображаемое в ячейке с формулой, пересчитывается при изменении значения ячейки, на которую указывает ссылка.

Ссылку на ячейку можно задать разными способами. Во-первых, адрес ячейки можно ввести вручную. Другой способ состоит в щелчке на нужной ячейке или выборе диапазона, адрес которого требуется ввести. Ячейка или диапазон при этом выделяются пунктирной рамкой.

Все диалоговые окна программы *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 щелчка или протягивания (рис. 12.3).

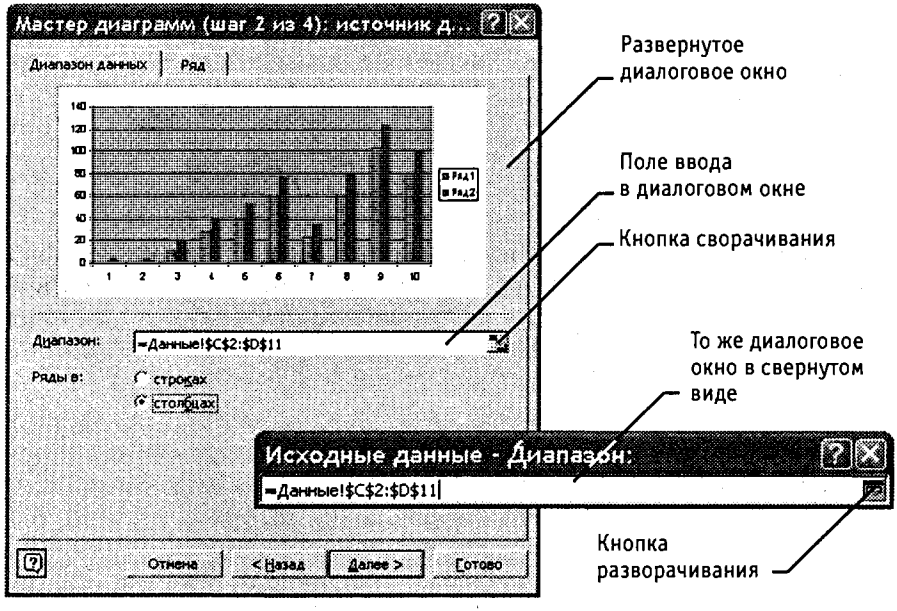


Рис. 12.3. Диалоговое окно в развернутом и свернутом виде

Для редактирования формулы следует дважды щелкнуть на соответствующей ячейке. При этом ячейки (диапазоны), от которых зависит значение формулы, выде-

ляются 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.

Абсолютные и относительные ссылки

По умолчанию, ссылки на ячейки в формулах рассматриваются как *относительные*. Это означает, что при копировании формулы адреса в ссылках автоматически изменяются в соответствии с относительным расположением исходной ячейки и создаваемой копии.

Пусть, например, в ячейке B2 имеется ссылка на ячейку A3. В относительном представлении можно сказать, что ссылка указывает на ячейку, которая располагается на один столбец левее и на одну строку ниже данной. Если формула будет скопирована в другую ячейку, то такое относительное указание ссылки сохранится. Например, при копировании формулы в ячейку EA27 ссылка будет продолжать указывать на ячейку, располагающуюся левее и ниже, в данном случае на ячейку DZ28.

При *абсолютной адресации* адреса ссылок при копировании не изменяются, так что ячейка, на которую указывает ссылка, рассматривается как нетабличная. Для изменения способа адресации при редактировании формулы надо выделить ссылку на ячейку и нажать клавишу F4. Элементы номера ячейки, использующие абсолютную адресацию, предваряются символом $. Например, при последовательных нажатиях клавиши F4 номер ячейки A1 будет записываться как A1, $A$1, A$1 и $A1. В двух последних случаях один из компонентов номера ячейки рассматривается как абсолютный, а другой — как относительный.

Копирование содержимого ячеек

Копирование и перемещение ячеек в программе *Excel* можно осуществлять методом перетаскивания или через буфер обмена. При работе с небольшим числом ячеек удобно использовать первый метод, при работе с большими диапазонами — второй.

Метод перетаскивания. Чтобы методом перетаскивания скопировать или переместить текущую ячейку (выделенный диапазон) вместе с содержимым, следует навести указатель мыши на рамку текущей ячейки (он примет вид стрелки с дополнительными стрелочками). Теперь ячейку можно перетащить в любое место рабочего листа (точка вставки помечается всплывающей подсказкой).

Для выбора способа выполнения этой операции, а также для более надежного контроля над ней рекомендуется использовать *специальное перетаскивание* с помощью правой кнопки мыши. В этом случае при отпускании кнопки мыши появляется специальное меню, в котором можно выбрать конкретную выполняемую операцию.

Применение буфера обмена. Передача информации через буфер обмена имеет в программе *Excel* определенные особенности, связанные со сложностью контроля над этой операцией. Вначале необходимо выделить копируемый (вырезаемый) диапазон и дать команду на его помещение в буфер обмена: Правка ▸ Копировать или Правка ▸ Вырезать. Вставка данных в рабочий лист возможна лишь немедленно после их помещения в буфер обмена. Попытка выполнить любую другую опера-

цию приводит к отмене начатого процесса копирования или перемещения. Однако утраты данных не происходит, поскольку «вырезанные» данные удаляются из места их исходного размещения только в момент выполнения вставки.

Место вставки определяется путем указания ячейки, соответствующей верхнему левому углу диапазона, помещенного в буфер обмена, или путем выделения диапазона, который по размерам в точности равен копируемому (перемещаемому). Вставка выполняется командой Правка ▸ Вставить. Для управления способом вставки можно использовать команду Правка ▸ Специальная вставка. В этом случае правила вставки данных из буфера обмена задаются в открывшемся диалоговом окне.

Автоматизация ввода

Так как таблицы часто содержат повторяющиеся или однотипные данные, программа *Excel* содержит средства автоматизации ввода. К числу предоставляемых средств относятся: *автозавершение*, *автозаполнение числами* и *автозаполнение формулами*.

Автозавершение. Для автоматизации ввода текстовых данных используется метод *автозавершения*. Его применяют при вводе в ячейки одного столбца рабочего листа текстовых строк, среди которых есть повторяющиеся. В ходе ввода текстовых данных в очередную ячейку программа *Excel* проверяет соответствие введенных символов строкам, имеющемся в этом столбце выше. Если обнаружено однозначное совпадение, введенный текст автоматически дополняется. Нажатие клавиши ENTER подтверждает операцию автозавершения, в противном случае ввод можно продолжать, не обращая внимания на предлагаемый вариант.

Можно прервать работу средства автозавершения, оставив в столбце пустую ячейку. И наоборот, чтобы использовать возможности средства автозавершения, заполненные ячейки должны идти подряд, без промежутков между ними.

Автозаполнение числами. При работе с числами используется метод *автозаполнения*. В правом нижнем углу рамки текущей ячейки имеется черный квадратик — *маркер заполнения*. При наведении на него указатель мыши (он обычно имеет вид толстого белого креста) приобретает форму тонкого черного крестика. Перетаскивание маркера заполнения рассматривается как операция «размножения» содержимого ячейки в горизонтальном или вертикальном направлении.

Если ячейка содержит число (в том числе дату, денежную сумму), то при перетаскивании маркера происходит копирование ячеек или их заполнение арифметической прогрессией. Для выбора способа автозаполнения следует производить специальное перетаскивание с использованием правой кнопки мыши.

Пусть, например, ячейка A1 содержит число 1. Наведите указатель мыши на маркер заполнения, нажмите правую кнопку мыши и перетащите маркер заполнения так, чтобы рамка охватила ячейки A1, B1 и C1, и отпустите кнопку мыши. Если теперь выбрать в открывшемся меню пункт Копировать ячейки, все ячейки будут содержать число 1. Если же выбрать пункт Заполнить, то в ячейках окажутся числа 1, 2 и 3.

Чтобы точно сформулировать условия заполнения ячеек, следует дать команду Правка ▸ Заполнить ▸ Прогрессия. В открывшемся диалоговом окне Прогрессия выбирается тип прогрессии, величина шага и предельное значение. После щелчка

на кнопке OK программа *Excel* автоматически заполняет ячейки в соответствии с заданными правилами.

Автозаполнение формулами. Эта операция выполняется так же, как автозаполнение числами. Ее особенность заключается в необходимости копирования ссылок на другие ячейки. В ходе автозаполнения во внимание принимается характер ссылок в формуле: относительные ссылки изменяются в соответствии с относительным расположением копии и оригинала, абсолютные остаются без изменений.

Для примера предположим, что значения в третьем столбце рабочего листа (столбце C) вычисляются как суммы значений в соответствующих ячейках столбцов A и B. Введем в ячейку C1 формулу =A1+B1. Теперь скопируем эту формулу методом автозаполнения во все ячейки третьего столбца таблицы. Благодаря относительной адресации формула будет правильной для всех ячеек данного столбца.

В таблице 12.1 приведены правила обновления ссылок при автозаполнении вдоль строки или вдоль столбца.

Таблица 12.1. Правила обновления ссылок при автозаполнении

Ссылка в исходной ячейке	Ссылка в следующей ячейке	
	При заполнении вправо	При заполнении вниз
A1 (относительная)	B1	A2
$A1 (абсолютная по столбцу)	$A1	$A2
A$1 (абсолютная по строке)	B$1	A$1
$A$1 (абсолютная)	$A$1	$A$1

Использование стандартных функций

Стандартные функции используются в программе *Excel* только в формулах. *Вызов функции* состоит в указании в формуле *имени функции*, после которого в скобках указывается *список параметров*. Отдельные параметры разделяются в списке точкой с запятой. В качестве параметра может использоваться число, адрес ячейки или произвольное выражение, для вычисления которого также могут использоваться функции.

В режиме ввода формулы в левой части строки формул, где раньше располагался номер текущей ячейки, появляется раскрывающийся список функций. Он содержит десять функций, которые использовались последними, а также пункт Другие функции.

Использование мастера функций. При выборе пункта Другие функции запускается Мастер функций, облегчающий выбор нужной функции. В раскрывающемся списке Категория выбирается категория, к которой относится функция (если определить категорию затруднительно, используют пункт Полный алфавитный перечень), а в списке Выберите функцию — конкретная функция данной категории. После щелчка на кнопке OK имя функции заносится в строку формул вместе со скобками, ограничивающими список параметров. Текстовый курсор устанавливается между этими скобками. Вызвать Мастер функций можно и проще, щелчком на кнопке Вставка функции в строке формул.

Аргументы функции. Как только имя функции выбрано, на экране появляется диалоговое окно Аргументы функции (в предыдущих версиях *Excel* это окно рассматривалось как *палитра формул*). Это окно, в частности, содержит значение, которое получится, если немедленно закончить ввод формулы (рис. 12.4).

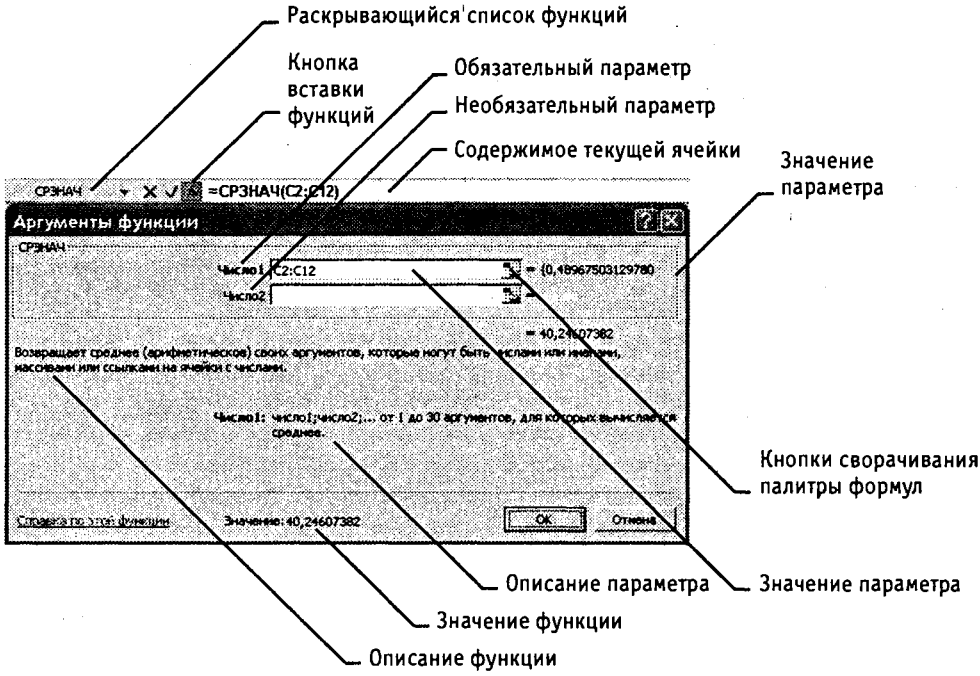


Рис. 12.4. Строка формул и диалоговое окно Аргументы функции

Правила вычисления формул, содержащих функции, не отличаются от правил вычисления более простых формул. Ссылки на ячейки, используемые в качестве параметров функции, также могут быть относительными или абсолютными, что учитывается при копировании формул методом автозаполнения.

12.3. Печать документов Excel

Экранное представление электронной таблицы в *Excel* значительно отличается от того, которое получилось бы при выводе данных на печать. Это связано с тем, что единый рабочий лист приходится разбивать на фрагменты, размер которых определяется форматом печатного листа. Кроме того, элементы оформления рабочего окна программы: номера строк и столбцов, условные границы ячеек — обычно не отображаются при печати.

Предварительный просмотр

Перед печатью рабочего листа следует перейти в режим *предварительного просмотра* (кнопка Предварительный просмотр на стандартной панели инструментов). Режим предварительного просмотра (рис. 12.5) не допускает редактирования документа,

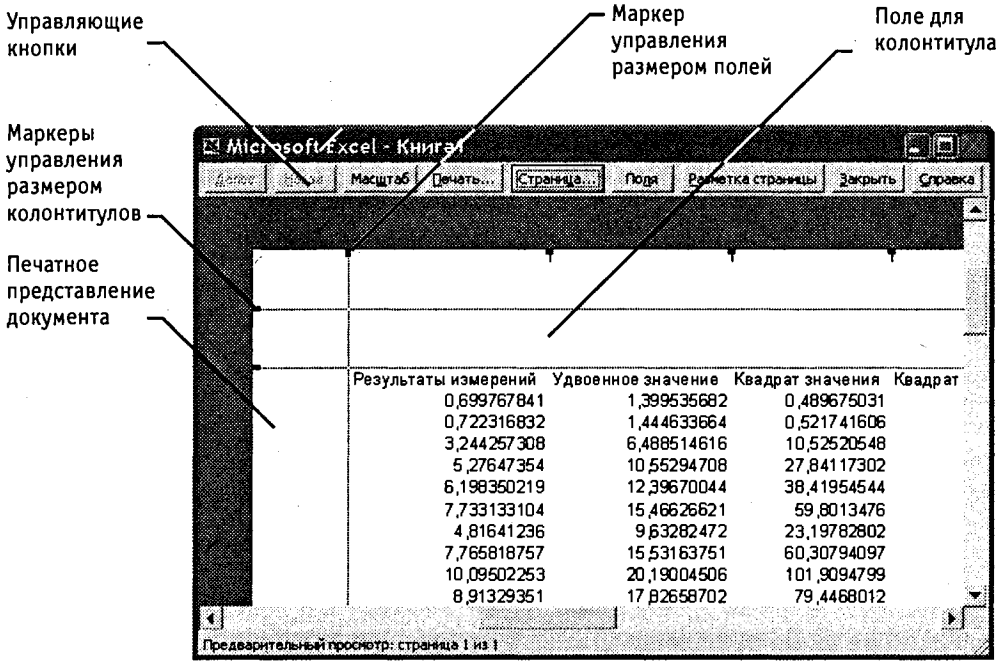


Рис. 12.5. Предварительный просмотр документа перед печатью

но позволяет увидеть его на экране точно в таком виде, в каком он будет напечатан. Кроме того, режим предварительного просмотра позволяет изменить свойства печатной страницы и параметры печати.

Управление в режиме предварительного просмотра осуществляется при помощи кнопок, расположенных вдоль верхнего края окна. Кнопка Страница открывает диалоговое окно Параметры страницы, которое служит для задания параметров страницы: ориентации листа, масштаба страницы (изменение масштаба позволяет управлять числом печатных страниц, необходимых для документа), размеров полей документа. Здесь же можно задать верхние и нижние колонтитулы для страницы. На вкладке Лист включается или отключается печать сетки и номеров строк и столбцов, а также выбирается последовательность разбиения на страницы рабочего листа, превосходящего размеры печатной страницы как по длине, так и по ширине.

Изменить величину полей страницы, а также ширину ячеек при печати можно также непосредственно в режиме предварительного просмотра, при помощи кнопки Поля. При щелчке на этой кнопке на странице появляются маркеры, указывающие границы полей страницы и ячеек. Изменить положение этих границ можно методом перетаскивания.

Завершить работу в режиме предварительного просмотра можно тремя способами, в зависимости от того, что планируется делать дальше. Щелчок на кнопке Закрыть позволяет вернуться к редактированию документа. Щелчок на кнопке Разметка

страницы служит для возврата к редактированию документа, но в режиме *разметки страницы*. В этом режиме документ отображается таким образом, чтобы наиболее удобно показать не содержимое ячеек таблицы, а *область печати* и границы страниц документа. Переключение между режимом разметки и обычным режимом можно также осуществлять через меню Вид (команды Вид ▸ Обычный и Вид ▸ Разметка страницы). Третий способ — начать печать документа.

Печать документа

Щелчок на кнопке Печать открывает диалоговое окно Печать, используемое для распечатки документа (его можно открыть и без предварительного просмотра — с помощью команды Файл ▸ Печать). Это окно содержит стандартные средства управления, применяемые для печати документов в любых приложениях.

Выбор области печати

Область печати — эта часть рабочего листа, которая должна быть выведена на печать. По умолчанию область печати совпадает с заполненной частью рабочего листа и представляет собой прямоугольник, примыкающий к верхнему левому углу рабочего листа и захватывающий все заполненные ячейки. Если часть данных не должна выводиться на бумагу, область печати можно задать вручную. Для этого надо выделить ячейки, которые должны быть включены в область печати, и дать команду Файл ▸ Область печати ▸ Задать. Если текущей является одна-единственная ячейка, то программа предполагает, что область печати не выделена, и выдает предупреждающее сообщение.

Если область печати задана, то программа отображает в режиме предварительного просмотра и распечатывает только ее. Границы области печати выделяются на рабочем листе крупным пунктиром (сплошной линией в режиме разметки). Для изменения области печати можно задать новую область или при помощи команды Файл ▸ Область печати ▸ Убрать вернуться к параметрам, используемым по умолчанию.

Границы отдельных печатных страниц отображаются на рабочем листе мелким пунктиром. В некоторых случаях требуется, чтобы определенные ячейки располагались вместе на одной и той же печатной странице или, наоборот, разделение печатных страниц происходило в определенном месте рабочего листа. Такая возможность реализуется путем задания границ печатных страниц вручную. Чтобы вставить разрыв страницы, надо сделать текущей ячейку, которая будет располагаться в левом верхнем углу печатной страницы, и дать команду Вставка ▸ Разрыв страницы. Программа *Excel* вставит принудительные разрывы страницы перед строкой и столбцом, в которых располагается данная ячейка. Если выбранная ячейка находится в первой строке или столбце A, то разрыв страницы задается только по одному направлению.

12.4. Применение электронных таблиц для расчетов

В научно-технической деятельности программу *Excel* трудно рассматривать как основной вычислительный инструмент. Однако ее удобно применять в тех случаях, когда требуется быстрая обработка больших объемов данных. Она полезна для выполнения таких операций, как статистическая обработка и анализ данных, реше-

ние задач оптимизации, построение диаграмм и графиков. Для такого рода задач применяют как основные средства программы *Excel*, так и дополнительные (надстройки).

Итоговые вычисления

Итоговые вычисления предполагают получение числовых характеристик, описывающих определенный набор данных в целом. Например, возможно вычисление суммы значений, входящих в набор, среднего значения и других статистических характеристик, количества или доли элементов набора, удовлетворяющих определенных условиям. Проведение итоговых вычислений в программе *Excel* выполняется при помощи встроенных функций. Особенность использования таких *итоговых функций* состоит в том, что при их задании программа пытается «угадать», в каких ячейках заключен обрабатываемый набор данных, и задать параметры функции автоматически.

В качестве параметра итоговой функции обычно задается некоторый диапазон ячеек, размер которого определяется автоматически. Выбранный диапазон рассматривается как отдельный параметр («массив»), и в вычислениях используются все ячейки, составляющие его.

Суммирование. Для итоговых вычислений применяют ограниченный набор функций, наиболее типичной из которых является функция суммирования (СУММ). Это единственная функция, для применения которой есть отдельная кнопка на стандартной панели инструментов (кнопка Автосумма). Диапазон суммирования, выбираемый автоматически, включает ячейки с данными, расположенные над текущей ячейкой (предпочтительнее) или слева от нее и образующие непрерывный блок. При неоднозначности выбора используется диапазон, непосредственно примыкающий к текущей ячейке.

Автоматический подбор диапазона не исключает возможности редактирования формулы. Можно переопределить диапазон, который был выбран автоматически, а также задать дополнительные параметры функции.

Функции для итоговых вычислений. Прочие функции для итоговых вычислений выбираются обычным образом, с помощью раскрывающегося списка в строке формул или с использованием мастера функций. Все эти функции относятся к категории Статистические. В их число входят функции ДИСП (вычисляет дисперсию), МАКС (максимальное число в диапазоне), СРЗНАЧ (среднее арифметическое значение чисел диапазона), СЧЕТ (подсчет ячеек с числами в диапазоне) и другие.

Функции, предназначенные для выполнения итоговых вычислений, часто применяют при использовании таблицы *Excel* в качестве базы данных, а именно на фоне фильтрации записей или при создании сводных таблиц.

Использование надстроек

Надстройки — это специальные средства, расширяющие возможности программы *Excel*. На практике именно надстройки делают программу *Excel* удобной для использования в научно-технической работе. Хотя эти средства считаются внешними,

дополнительными, доступ к ним осуществляется при помощи обычных команд строки меню (обычно через меню Сервис или Данные). Команда использования настройки обычно открывает специальное диалоговое окно, оформление которого не отличается от стандартных диалоговых окон программы *Excel*.

Подключить или отключить установленные надстройки можно с помощью команды Сервис ▸ Надстройки (рис. 12.6). Подключение надстроек увеличивает нагрузку на вычислительную систему, поэтому обычно рекомендуют подключать только те надстройки, которые реально используются.

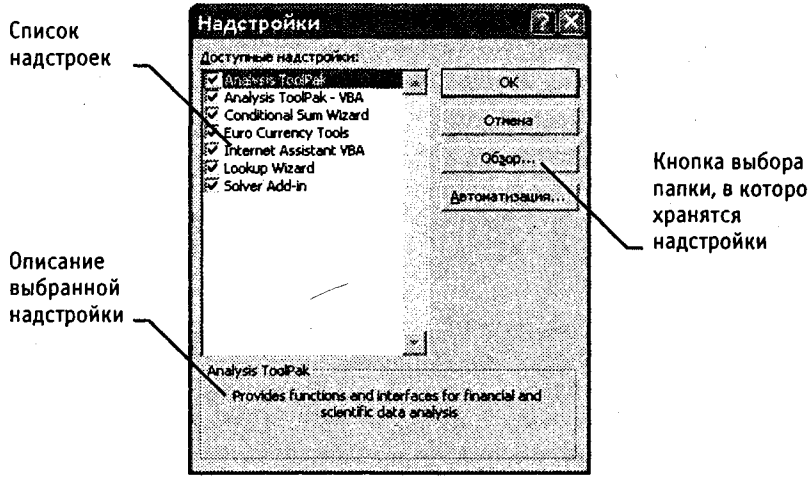


Рис. 12.6. Диалоговое окно для подключения и отключения надстроек

Вот основные надстройки, поставляемые вместе с программой *Excel*.

Пакет анализа (*Analysis ToolPak*). Обеспечивает дополнительные возможности анализа наборов данных. Выбор конкретного метода анализа осуществляется в диалоговом окне Data Analysis (Анализ данных), которое открывается командой Сервис ▸ Data Analysis (Анализ данных).

Мастер суммирования (*Conditional Sum Wizard*). Позволяет автоматизировать создание формул для суммирования данных в столбце таблицы. При этом ячейки могут включаться в сумму только при выполнении определенных условий. Запуск мастера осуществляется с помощью команды Сервис ▸ Conditional Sum (Частичная сумма).

Мастер подстановок (*Lookup Wizard*). Автоматизирует создание формулы для поиска данных в таблице по названию столбца и строки. Мастер позволяет произвести однократный поиск или предоставляет возможность ручного задания параметров, используемых для поиска. Вызывается командой Сервис ▸ Lookup (Поиск).

Поиск решения (*Solver Add-in*). Эта надстройка используется для решения задач оптимизации. Ячейки, для которых подбираются оптимальные значения и задаются ограничения, выбираются в диалоговом окне Solver Parameters (Поиск решения), которое открывают при помощи команды Сервис ▸ Solver (Поиск решения).

12.5. Построение диаграмм и графиков

В программе *Excel* термин «диаграмма» используется для обозначения всех видов графического представления числовых данных. Построение графического изображения производится на основе *ряда данных*. Так называют группу ячеек с данными в пределах отдельной строки или столбца. На одной диаграмме можно отображать несколько рядов данных.

Диаграмма представляет собой вставной объект, внедренный на один из листов рабочей книги. Она может располагаться на том же листе, на котором находятся данные, или на любом другом листе (часто для отображения диаграммы отводят отдельный лист). Диаграмма сохраняет связь с данными, на основе которых она построена, и при обновлении этих данных немедленно изменяет свой вид.

Для построения диаграммы обычно используют Мастер диаграмм, запускаемый щелчком на кнопке Мастер диаграмм на стандартной панели инструментов. Часто удобно заранее выделить область, содержащую данные, которые будут отображаться на диаграмме, но задать эту информацию можно и в ходе работы мастера.

Выбор типа диаграммы

На первом этапе работы мастера выбирают форму диаграммы. Доступные формы перечислены в списке Тип на вкладке Стандартные. Для выбранного типа диаграммы справа указывается несколько вариантов представления данных (палитра Вид), из которых следует выбрать наиболее подходящий. На вкладке Нестандартные отображается набор полностью сформированных типов диаграмм с готовым форматированием. После задания формы диаграммы следует щелкнуть на кнопке Далее.

Выбор данных

Второй этап работы мастера служит для выбора данных, по которым будет строиться диаграмма (рис. 12.7). Если диапазон данных был выбран заранее, то в области предварительного просмотра в верхней части окна мастера появится приблизительное отображение будущей диаграммы. Если данные образуют единый прямоугольный диапазон, то их удобно выбирать при помощи вкладки Диапазон данных. Если данные не образуют единой группы, то информацию для отрисовки отдельных рядов данных задают на вкладке Ряд. Предварительное представление диаграммы автоматически обновляется при изменении набора отображаемых данных.

Оформление диаграммы

Третий этап работы мастера (после щелчка на кнопке Далее) состоит в выборе оформления диаграммы. На вкладках окна мастера задаются:

- название диаграммы, подписи осей (вкладка Заголовки);
- отображение и маркировка осей координат (вкладка Оси);
- отображение сетки линий, параллельных осям координат (вкладка Линии сетки);
- описание построенных графиков (вкладка Легенда);

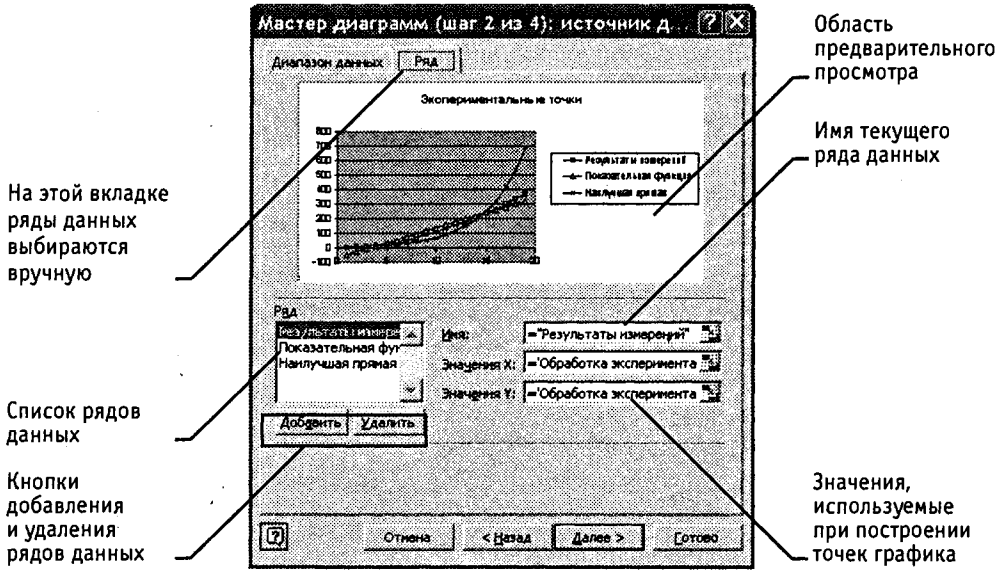


Рис. 12.7. Выбор данных, отображаемых на диаграмме

- отображение надписей, соответствующих отдельным элементам данных на графике (вкладка Подписи данных);
- представление данных, использованных при построении графика, в виде таблицы (вкладка Таблица данных).

В зависимости от типа диаграммы некоторые из перечисленных вкладок могут отсутствовать.

Размещение диаграммы

На последнем этапе работы мастера (после щелчка на кнопке Далее) указывается, следует ли использовать для размещения диаграммы новый рабочий лист или один из имеющихся. Обычно этот выбор важен только для последующей печати документа, содержащего диаграмму. После щелчка на кнопке Готово диаграмма строится автоматически и вставляется на указанный рабочий лист (рис. 12.8).

Редактирование диаграммы

Готовую диаграмму можно изменить. Она состоит из набора отдельных элементов, таких, как сами графики (ряды данных), оси координат, заголовок диаграммы, область построения и прочее. При щелчке на элементе диаграммы он выделяется маркерами, а при наведении на него указателя мыши — описывается всплывающей подсказкой. Открыть диалоговое окно для форматирования элемента диаграммы можно через меню Формат (для выделенного элемента) или через контекстное меню (команда Формат). Различные вкладки открывшегося диалогового окна позволяют изменять параметры отображения выбранного элемента данных.

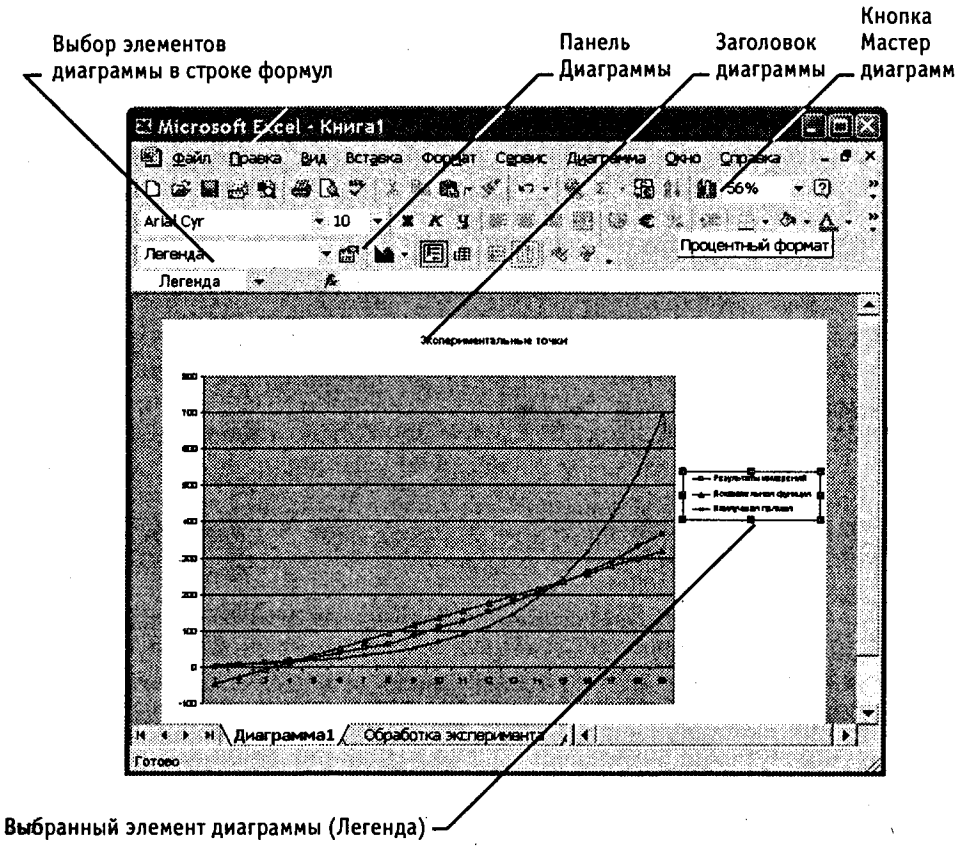


Рис. 12.8. Готовая диаграмма Excel

Если требуется внести в диаграмму существенные изменения, следует вновь воспользоваться мастером диаграмм. Для этого следует открыть рабочий лист с диаграммой или выбрать диаграмму, внедренную в рабочий лист с данными. Запустив мастер диаграмм, можно изменить текущие параметры, которые рассматриваются в окнах мастера как заданные по умолчанию.

Чтобы удалить диаграмму, можно удалить рабочий лист, на котором она расположена (Правка ▸ Удалить лист), или выбрать диаграмму, внедренную в рабочий лист с данными, и нажать клавишу DELETE.

Практическое занятие

Упражнение 12.1. Обработка данных

1. Запустите программу *Excel* (Пуск ▸ Программы ▸ Microsoft Excel).
2. Создайте новую рабочую книгу (кнопка Создать на стандартной панели инструментов).

3. Дважды щелкните на ярлычке текущего рабочего листа и дайте этому рабочему листу имя Данные.
4. Дайте команду Файл ▸ Сохранить как и сохраните рабочую книгу под именем book.xls.
5. Сделайте текущей ячейку A1 и введите в нее заголовок Результаты измерений.
6. Введите произвольные числа в последовательные ячейки столбца A, начиная с ячейки A2.
7. Введите в ячейку B1 строку Удвоенное значение.
8. Введите в ячейку C1 строку Квадрат значения.
9. Введите в ячейку D1 строку Квадрат следующего числа.
10. Введите в ячейку B2 формулу =2*A2.
11. Введите в ячейку C2 формулу =A2*A2.
12. Введите в ячейку D2 формулу =B2+C2+1.
13. Выделите протягиванием ячейки B2, C2 и D2.
14. Наведите указатель мыши на маркер заполнения в правом нижнем углу рамки, охватывающей выделенный диапазон. Нажмите левую кнопку мыши и перетащите этот маркер, чтобы рамка охватила столько строк в столбцах B, C и D, сколько имеется чисел в столбце A.
15. Убедитесь, что формулы автоматически модифицируются так, чтобы работать со значением ячейки в столбце A текущей строки.
16. Измените одно из значений в столбце A и убедитесь, что соответствующие значения в столбцах B, C и D в этой же строке были автоматически пересчитаны.
17. Введите в ячейку E1 строку Масштабный множитель.
18. Введите в ячейку E2 число 5.
19. Введите в ячейку F1 строку Масштабирование.
20. Введите в ячейку F2 формулу =A2*E2.
21. Используйте метод автозаполнения, чтобы скопировать эту формулу в ячейки столбца F, соответствующие заполненным ячейкам столбца A.
22. Убедитесь, что результат масштабирования оказался неверным. Это связано с тем, что адрес E2 в формуле задан относительной ссылкой.
23. Щелкните на ячейке F2, затем в строке формул. Установите текстовый курсор на ссылку E2 и нажмите клавишу F4. Убедитесь, что формула теперь выглядит как =A2*$E$2, и нажмите клавишу ENTER.
24. Повторите заполнение столбца F формулой из ячейки F2.
25. Убедитесь, что благодаря использованию абсолютной адресации значения ячеек столбца F теперь вычисляются правильно. Сохраните рабочую книгу book.xls.

Мы научились вводить текстовые и числовые данные в электронные таблицы Excel. Мы узнали, как производится ввод и вычисление формул. Мы также выяснили, как осуществляется копирование формул методом автозаполнения, и определили, в каких случаях следует использовать относительные и абсолютные ссылки.

Упражнение 12.2. Применение итоговых функций

1. Запустите программу *Excel* (Пуск ▸ Программы ▸ Microsoft Excel) и откройте рабочую книгу book.xls, созданную ранее.
2. Выберите рабочий лист Данные.
3. Сделайте текущей первую свободную ячейку в столбце A.
4. Щелкните на кнопке Автосумма на стандартной панели инструментов.
5. Убедитесь, что программа автоматически подставила в формулу функцию СУММ и правильно выбрала диапазон ячеек для суммирования. Нажмите клавишу ENTER.
6. Сделайте текущей следующую свободную ячейку в столбце A.
7. Щелкните на кнопке Вставка функции в строке формул.
8. В раскрывающемся списке Категория выберите пункт Статистические.
9. В списке Функция выберите функцию СРЗНАЧ и щелкните на кнопке OK.
10. Переместите методом перетаскивания окно Аргументы функции, если оно заслоняет нужные ячейки. Обратите внимание, что автоматически выбранный диапазон включает все ячейки с числовым содержимым, включая и ту, которая содержит сумму. Выделите правильный диапазон методом протягивания и нажмите клавишу ENTER.
11. Используя порядок действий, описанный в пп. 6–10, вычислите минимальное число в заданном наборе (функция МИН), максимальное число (МАКС), количество элементов в наборе (СЧЕТ).
12. Сохраните рабочую книгу book.xls.

Мы познакомились с некоторыми итоговыми функциями. Мы научились использовать итоговые функции для вычисления значений, характеризующих набор данных. Мы выяснили, как автоматически определяется диапазон значений, обрабатываемых функцией, и как изменить его вручную.

Упражнение 12.3. Подготовка и форматирование прайс-листа

1. Запустите программу *Excel* (Пуск ▸ Программы ▸ Microsoft Excel) и откройте рабочую книгу book.xls.
2. Выберите щелчком на ярлычке неиспользуемый рабочий лист или создайте новый (Вставка ▸ Лист). Дважды щелкните на ярлычке нового листа и переименуйте его как Прейскурант.
3. В ячейку A1 введите текст Прейскурант и нажмите клавишу ENTER.
4. В ячейку A2 введите текст Курс пересчета: и нажмите клавишу ENTER. В ячейку B2 введите текст 1 у.е.= и нажмите клавишу ENTER. В ячейку C2 введите текущий курс пересчета и нажмите клавишу ENTER.

5. В ячейку A3 введите текст Наименование товара и нажмите клавишу ENTER. В ячейку B3 введите текст Цена (у.е.) и нажмите клавишу ENTER. В ячейку C3 введите текст Цена (руб.) и нажмите клавишу ENTER.
6. В последующие ячейки столбца A введите названия товаров, включенных в прейскурант.
7. В соответствующие ячейки столбца B введите цены товаров в условных единицах.
8. В ячейку C4 введите формулу: =B4*$C$2, которая используется для пересчета цены из условных единиц в рубли.
9. Методом автозаполнения скопируйте формулы во все ячейки столбца C, которым соответствуют заполненные ячейки столбцов A и B. Почему при таком копировании получатся верные формулы?
10. Измените курс пересчета в ячейке C2. Обратите внимание, что все цены в рублях при этом обновляются автоматически.
11. Выделите методом протягивания диапазон A1:C1 и дайте команду Формат ▸ Ячейки. На вкладке Выравнивание задайте выравнивание по горизонтали По центру и установите флажок Объединение ячеек.
12. На вкладке Шрифт задайте размер шрифта равный 14 пунктам и в списке Начертание выберите вариант Полужирный. Щелкните на кнопке OK.
13. Щелкните правой кнопкой мыши на ячейке B2 и выберите в контекстном меню команду Формат ячеек. Задайте выравнивание по горизонтали По правому краю и щелкните на кнопке OK.
14. Щелкните правой кнопкой мыши на ячейке C2 и выберите в контекстном меню команду Формат ячеек. Задайте выравнивание по горизонтали По левому краю и щелкните на кнопке OK.
15. Выделите методом протягивания диапазон B2:C2. Щелкните на раскрывающей кнопке рядом с кнопкой Границы на панели инструментов Форматирование и задайте для этих ячеек толстую внешнюю границу (кнопка в правом нижнем углу открывшейся палитры).
16. Дважды щелкните на границе между заголовками столбцов A и B, B и C, C и D. Обратите внимание, как при этом изменяется ширина столбцов A, B и C.
17. Посмотрите, устраивает ли вас полученный формат таблицы. Щелкните на кнопке Предварительный просмотр на стандартной панели инструментов, чтобы увидеть, как документ будет выглядеть при печати.
18. Щелкните на кнопке Печать и напечатайте документ.
19. Сохраните рабочую книгу book.xls.

Мы научились форматировать документ Excel. При этом мы использовали такие средства, как изменение ширины столбцов, объединение ячеек, управление выравниванием текста, создание рамок ячеек. Мы выяснили, что в готовом документе заданные и вычисленные ячейки отображаются одинаково. Мы познакомились с использованием средства предварительного просмотра и произвели печать документа.

Упражнение 12.4. Построение экспериментального графика

1. Запустите программу *Excel* (Пуск ▸ Программы ▸ Microsoft Excel) и откройте рабочую книгу book.xls, созданную ранее.
2. Выберите щелчком на ярлычке неиспользуемый рабочий лист или создайте новый (Вставка ▸ Лист). Дважды щелкните на ярлычке листа и переименуйте его как Обработка эксперимента.
3. В столбец A, начиная с ячейки A1, введите произвольный набор значений независимой переменной.
4. В столбец B, начиная с ячейки B1, введите произвольный набор значений функции.
5. Методом протягивания выделите все заполненные ячейки столбцов A и B.
6. Щелкните на значке Мастер диаграмм на стандартной панели инструментов.
7. В списке Тип выберите пункт Точечная (для отображения графика, заданного парами значений). В палитре Вид выберите средний пункт в первом столбце (маркеры, соединенные гладкими кривыми). Щелкните на кнопке Далее.
8. Так как диапазон ячеек был выделен заранее, мастер диаграмм автоматически определяет расположение рядов данных. Убедитесь, что данные на диаграмме выбраны правильно. На вкладке Ряд в поле Имя укажите: Результаты измерений. Щелкните на кнопке Далее.
9. Выберите вкладку Заголовки. Убедитесь, что заданное название ряда данных автоматически использовано как заголовок диаграммы. Замените его, введя в поле Название диаграммы заголовок Экспериментальные точки. Щелкните на кнопке Далее.
10. Установите переключатель Отдельном. По желанию, задайте произвольное имя добавляемого рабочего листа. Щелкните на кнопке Готово.
11. Убедитесь, что диаграмма построена и внедрена в новый рабочий лист. Рассмотрите ее и щелкните на построенной кривой, чтобы выделить ряд данных.
12. Дайте команду Формат ▸ Выделенный ряд. Откройте вкладку Вид.
13. На панели Линия откройте палитру Цвет и выберите красный цвет. В списке Тип линии выберите пунктир.
14. На панели Маркер выберите в списке Тип маркера треугольный маркер. В палитрах Цвет и Фон выберите зеленый цвет.
15. Щелкните на кнопке OK, снимите выделение с ряда данных и посмотрите, как изменился вид графика.
16. Сохраните рабочую книгу.

Мы научились строить графики на основе данных, содержащихся на рабочем листе, настраивать формат диаграммы, задавать отображаемые данные и оформлять получающуюся диаграмму. Мы также узнали, как можно изменить формат готовой диаграммы.

Упражнение 12.5. Анализ данных с использованием метода наименьших квадратов

Задача. Для заданного набора пар значений независимой переменной и функции определить наилучшие линейное приближение в виде прямой с уравнением $y = ax + b$ и показательное приближение в виде линии с уравнением $y = b \cdot a^x$.

1. Запустите программу *Excel* (Пуск ▸ Программы ▸ Microsoft Excel) и откройте рабочую книгу book.xls, созданную ранее.
2. Щелчком на ярлычке выберите рабочий лист Обработка эксперимента.
3. Сделайте ячейку C1 текущей и щелкните на кнопке Вставка функции в строке формул.
4. В окне мастера функций выберите категорию Ссылки и массивы и функцию ИНДЕКС. В новом диалоговом окне выберите первый вариант набора параметров.
5. Установите текстовый курсор в первое поле для ввода параметров в окне Аргументы функции и выберите в раскрывающемся списке в строке формул пункт Другие функции.
6. С помощью мастера функций выберите функцию ЛИНЕЙН категории Статистические.
7. В качестве первого параметра функции ЛИНЕЙН выберите диапазон, содержащий значения функции (столбец B).
8. В качестве второго параметра функции ЛИНЕЙН выберите диапазон, содержащий значения независимой переменной (столбец A).
9. Переместите текстовый курсор в строке формул, чтобы он стоял на имени функции ИНДЕКС. В качестве второго параметра функции ИНДЕКС задайте число 1. Щелкните на кнопке OK в окне Аргументы функции.

Функция ЛИНЕЙН возвращает коэффициенты уравнения прямой в виде массива из двух элементов. С помощью функции ИНДЕКС выбирается нужный элемент.

10. Сделайте текущей ячейку D1. Повторите операции, описанные в пп. 3–9, чтобы в итоге в этой ячейке появилась формула: =ИНДЕКС(ЛИНЕЙН(B1:B20;A1:A20);2). Ее можно ввести и вручную (посимвольно). Теперь в ячейках C1 и D1 вычислены, соответственно, коэффициенты a и b уравнения наилучшей прямой.
11. Сделайте текущей ячейку C2. Повторите операции, описанные в пп. 3–9, или введите вручную следующую формулу:

    =ИНДЕКС(ЛГРФПРИБЛ(B1:B20;A1:A20);1)

12. Сделайте текущей ячейку D2. Повторите операции, описанные в пп. 3–9, или введите вручную следующую формулу:

    =ИНДЕКС(ЛГРФПРИБЛ(B1:B20;A1:A20);2)

    Теперь ячейки C2 и D2 содержат, соответственно, коэффициенты a и b уравнения наилучшего показательного приближения.

Для интерполяции или экстраполяции оптимальной кривой без явного определения ее параметров можно использовать функции ТЕНДЕНЦИЯ (для линейной зависимости) и РОСТ (для показательной зависимости).

13. Для построения наилучшей прямой другим способом дайте команду Сервис ▸ Data Analysis (Анализ данных).
14. Откроется одноименное диалоговое окно. В списке Analysis Tools (Инструменты анализа) выберите пункт Regression (Регрессия), после чего щелкните на кнопке OK.
15. В поле Input Y Range (Входной интервал Y) укажите методом протягивания диапазон, содержащий значения функции (столбец B).
16. В поле Input X Range (Входной интервал X) укажите методом протягивания диапазон, содержащий значения независимой переменной (столбец A).
17. Установите переключатель New Worksheet (Новый рабочий лист) и задайте для него имя Результат расчета.
18. Щелкните на кнопке OK и по окончании расчета откройте рабочий лист Результат расчета. Убедитесь, что вычисленные коэффициенты (см. ячейки B17 и B18) совпали с полученными первым методом.
19. Сохраните рабочую книгу book.xls.

Мы научились анализировать с помощью программы Excel экспериментальные данные с использованием метода наименьших квадратов. Мы применили для вычислений разные средства программы Excel. Мы получили информацию, необходимую для построения графиков нужных приближений.

Упражнение 12.6. Применение таблиц подстановки

Задача. Построить графики функций, коэффициенты которых определены в предыдущем упражнении.

1. Запустите программу *Excel* (Пуск ▸ Программы ▸ Microsoft Excel) и откройте рабочую книгу book.xls.
2. Выберите щелчком на ярлычке рабочий лист Обработка эксперимента.
3. Так как программа *Excel* не позволяет непосредственно строить графики функций, заданных формулами, необходимо сначала табулировать формулу, то есть создать таблицу значений функций для заданных значений переменной. Сделайте текущей ячейку C3 и занесите в нее значение 0. Эта ячейка будет использоваться как ячейка ввода, на которую будут ссылаться формулы.
4. Методом протягивания выделите значения в столбце A. Дайте команду Правка ▸ Копировать, чтобы перенести эти данные в буфер обмена. Сделайте текущей ячейку F2 и дайте команду Правка ▸ Вставить, чтобы скопировать заданные значения независимой переменной в столбец F, начиная со второй строки.
5. В ячейку G1 введите формулу =C3*$C$1+$D$1. Здесь C3 — ячейка ввода, а в качестве других ссылок используются вычисленные методом наименьших квадратов коэффициенты уравнения прямой.

6. В ячейку H1 введите формулу =$D$2*$C$2^C3 для вычисления значения показательной функции. В программе *Excel* можно табулировать несколько функций одной переменной в рамках единой операции.
7. Выделите прямоугольный диапазон, включающий столбцы F, G и H и строки от строки 1, содержащей формулы, до последней строки с данными в столбце F.
8. Дайте команду Данные ▸ Таблица подстановки. Выберите поле Подставлять значения по строкам в и щелкните на ячейке ввода C3.
9. Щелкните на кнопке OK, чтобы заполнить пустые ячейки в столбцах G и H выделенного диапазона значениями формул в ячейках первой строки для значений независимой переменной, выбранных из столбца F.
10. Переключитесь на рабочий лист Диаграмма1 (если используемое по умолчанию название листа с диаграммой было изменено, используйте свое название).
11. Щелкните на кнопке Мастер диаграмм на стандартной панели инструментов и пропустите первый этап щелчком на кнопке Далее.
12. Выберите вкладку Ряд и щелкните на кнопке Добавить. В поле Имя укажите: Наилучшая прямая. В поле Значения X укажите диапазон ячеек с данными в столбце F, а в поле Значения Y укажите диапазон ячеек в столбце G.
13. Еще раз щелкните на кнопке Добавить. В поле Имя укажите: Показательная функция. В поле Значения X укажите диапазон ячеек с данными в столбце F, а в поле Значения Y укажите диапазон ячеек в столбце H.
14. Щелкните на кнопке Готово, чтобы перестроить диаграмму в соответствии с новыми настройками.
15. Сохраните рабочую книгу book.xls.

Мы научились создавать таблицу подстановки, содержащую значения заданных формул для нужных значений независимой переменной. Мы применили эту возможность программы Excel для построения графиков функций, заданных формулами. Мы также научились редактировать ранее построенную диаграмму, нанося на нее дополнительные графики.

Упражнение 12.7. Решение уравнений средствами программы Excel

Задача. Найти решение уравнения $x^3 - 3x^2 + x = -1$.

1. Запустите программу *Excel* (Пуск ▸ Программы ▸ Microsoft Excel) и откройте рабочую книгу book.xls, созданную ранее.
2. Создайте новый рабочий лист (Вставка ▸ Лист), дважды щелкните на его ярлычке и присвойте ему имя Уравнение.
3. Занесите в ячейку A1 значение 0.
4. Занесите в ячейку B1 левую часть уравнения, используя в качестве независимой переменной ссылку на ячейку A1. Соответствующая формула может, например, иметь вид =A1^3-3*A1^2+A1.

5. Дайте команду Сервис ▸ Подбор параметра.
6. В поле Установить в ячейке укажите B1, в поле Значение задайте –1, в поле Изменяя значение ячейки укажите A1.
7. Щелкните на кнопке OK и посмотрите на результат подбора, отображаемый в диалоговом окне Результат подбора параметра. Щелкните на кнопке OK, чтобы сохранить полученные значения ячеек, участвовавших в операции.
8. Повторите расчет, задавая в ячейке A1 другие начальные значения, например 0,5 или 2. Совпали ли результаты вычислений? Чем можно объяснить различия?
9. Сохраните рабочую книгу book.xls.

Мы научились численно решать с помощью программы Excel уравнения, содержащие одно неизвестное и задаваемые формулой. Мы выяснили, что при наличии нескольких корней результат решения уравнения зависит от того, какое число было выбрано в качестве начального приближения.

Упражнение 12.8. Решение задач оптимизации

Задача. Завод производит электронные приборы трех видов (прибор A, прибор B и прибор C), используя при сборке микросхемы трех типов (тип 1, тип 2 и тип 3). Расход микросхем задается следующей таблицей:

	Прибор A	Прибор B	Прибор C
Тип 1	2	5	1
Тип 2	2	0	4
Тип 3	2	1	1

Стоимость изготовленных приборов одинакова.

Ежедневно на склад завода поступает 400 микросхем типа 1 и по 500 микросхем типов 2 и 3. Каково оптимальное соотношение дневного производства приборов различного типа, если производственные мощности завода позволяют использовать запас поступивших микросхем полностью?

1. Запустите программу *Excel* (Пуск ▸ Программы ▸ Microsoft Excel) и откройте рабочую книгу book.xls, созданную ранее.
2. Создайте новый рабочий лист (Вставка ▸ Лист), дважды щелкните на его ярлычке и присвойте ему имя Организация производства.
3. В ячейки A2, A3 и A4 занесите дневной запас комплектующих — числа 400, 500 и 500 соответственно.
4. В ячейки C1, D1 и E1 занесите нули — в дальнейшем значения этих ячеек будут подобраны автоматически.
5. В ячейках диапазона C2:E4 разместите таблицу расхода комплектующих.
6. В ячейках B2:B4 нужно указать формулы для расчета расхода комплектующих по типам. В ячейке B2 формула будет иметь вид =$C$1*C2+$D$1*D2+$E$1*E2,

а остальные формулы можно получить методом автозаполнения (обратите внимание на использование абсолютных и относительных ссылок).

7. В ячейку F1 занесите формулу, вычисляющую общее число произведенных приборов: для этого выделите диапазон C1:E1 и щелкните на кнопке Автосумма на стандартной панели инструментов.
8. Дайте команду Сервис ▸ Solver (Поиск решения) — откроется диалоговое окно Solver Parameters (Поиск решения).
9. В поле Set Target Cell (Установить целевую) укажите ячейку, содержащую оптимизируемое значение (F1). Установите переключатель Equal To Max (Равной максимальному значению) (требуется максимальный объем производства).
10. В поле By Changing Cells (Изменяя ячейки) задайте диапазон подбираемых параметров — C1:E1.
11. Чтобы определить набор ограничений, щелкните на кнопке Add (Добавить). В диалоговом окне Add Constraint (Добавление ограничения) в поле Cell Reference (Ссылка на ячейку) укажите диапазон B2:B4. В качестве условия задайте <=. В поле Constraint (Ограничение) задайте диапазон A2:A4. Это условие указывает, что дневной расход комплектующих не должен превосходить запасов. Щелкните на кнопке OK.
12. Снова щелкните на кнопке Add (Добавить). В поле Cell Reference (Ссылка на ячейку) укажите диапазон C1:E1. В качестве условия задайте >=. В поле Constraint (Ограничение) задайте число 0. Это условие указывает, что число производимых приборов неотрицательно. Щелкните на кнопке OK.
13. Снова щелкните на кнопке Add (Добавить). Cell Reference (Ссылка на ячейку) укажите диапазон C1:E1. В качестве условия выберите пункт int (цел). Это условие не позволяет производить доли приборов. Щелкните на кнопке OK.
14. Щелкните на кнопке Solve (Выполнить). По завершении оптимизации откроется диалоговое окно Solver Results (Результаты поиска решения).
15. Установите переключатель Keep Solver Solution (Сохранить найденное решение), после чего щелкните на кнопке OK.
16. Проанализируйте полученное решение. Кажется ли оно очевидным? Проверьте его оптимальность, экспериментируя со значениями ячеек C1:E1. Чтобы восстановить оптимальные значения, можно в любой момент повторить операцию поиска решения.
17. Сохраните рабочую книгу book.xls.

Мы узнали, как использовать программу Excel для решения сложных задач оптимизации. Мы научились формулировать условия задачи табличным образом, формировать ограничения, которым должно удовлетворять решение, и производить поиск оптимального набора переменных. Мы также выяснили, что даже для несложной задачи оптимизации найти оптимальное решение подбором практически невозможно.

Компьютерные программы создают *программисты* — люди, обученные процессу их составления (*программированию*). Мы знаем, что программа — это логически упорядоченная последовательность команд, необходимых для управления компьютером (выполнения им конкретных операций), поэтому программирование сводится к созданию последовательности команд, необходимой для решения определенной задачи.

20.1. Языки программирования

Машинный код процессора

Процессор компьютера — это большая интегральная микросхема. Все команды и данные он получает в виде электрических сигналов. Фактически процессор можно рассматривать как огромную совокупность достаточно простых электронных элементов — транзисторов. Транзистор имеет три вывода. На два крайних подается напряжение, необходимое для создания в транзисторе электрического тока, а на средний вывод — напряжение, с помощью которого можно управлять внутренним сопротивлением транзистора, а значит, управлять и током, и напряжением на его выводах.

В электронике транзисторы имеют три применения: для создания усилителей, в электронных схемах, обладающих автоколебательными свойствами, и в электронных переключателях. Последний способ и применяется в цифровой вычислительной технике. В процессоре компьютера транзисторы сгруппированы в микроэлементы, называемые *триггерами* и *вентилями*. Триггеры имеют два устойчивых состояния (*открыт — закрыт*) и переключаются из одного состояния в другое электрическими сигналами. Этим устойчивым состояниям соответствуют математические понятия 0 или 1. Вентили немного сложнее — они могут иметь несколько входов (напряжение на выходе зависит от комбинаций напряжений на входах) и служат для простейших арифметических и логических операций.

Команды, поступающие в процессор по его шинам, на самом деле являются электрическими сигналами, но и их тоже можно представить как совокупности нулей и единиц, то есть числами. Разным командам соответствуют разные числа. Поэтому реально программа, с которой работает процессор, представляет собой последовательность чисел, называемую *машинным кодом*.

Алгоритм и программа

Управлять компьютером нужно по определенному *алгоритму*. Алгоритм — это точно определенное описание способа решения задачи в виде конечной (по времени) последовательности действий. Такое описание еще называется *формальным*. Для представления алгоритма в виде, понятном компьютеру, служат *языки программирования*. Сначала всегда разрабатывается алгоритм действий, а потом он записывается на одном из таких языков. В итоге получается текст программы — полное, законченное и детальное описание алгоритма на языке программирования. Затем этот текст программы специальными служебными приложениями, которые называются *трансляторами*, либо переводится в машинный код, либо исполняется.

Что такое язык программирования

Самому написать программу в машинном коде весьма сложно, причем эта сложность резко возрастает с увеличением размера программы и трудоемкости решения нужной задачи. Условно можно считать, что машинный код приемлем, если размер программы не превышает нескольких десятков байтов и нет потребности в операциях ручного ввода/вывода данных.

Поэтому сегодня практически все программы создаются с помощью языков программирования. Теоретически программу можно написать и средствами обычного человеческого (естественного) языка — это называется программированием на *метаязыке* (подобный подход обычно используется на этапе составления алгоритма), но автоматически перевести такую программу в машинный код пока невозможно из-за высокой неоднозначности естественного языка.

Языки программирования — искусственные языки. От естественных они отличаются ограниченным числом «слов», значение которых понятно транслятору, и очень строгими правилами записи команд (*операторов*). Совокупность подобных требований образует *синтаксис* языка программирования, а *смысл* каждой команды и других конструкций языка — его *семантику*. Нарушение формы записи программы приводит к тому, что транслятор не может понять назначение оператора и выдает сообщение о синтаксической ошибке, а правильно написанное, но не отвечающее алгоритму использование команд языка приводит к семантическим ошибкам (называемым еще логическими ошибками или ошибками времени выполнения).

Процесс поиска ошибок в программе называется *тестированием*, процесс устранения ошибок — *отладкой*.

Компиляторы и интерпретаторы

С помощью языка программирования создается не готовая программа, а только ее текст, описывающий ранее разработанный алгоритм. Чтобы получить работающую

программу, надо этот текст либо автоматически перевести в машинный код (для этого служат программы-*компиляторы*) и затем использовать отдельно от исходного текста, либо сразу выполнять команды языка, указанные в тексте программы (этим занимаются программы-*интерпретаторы*).

Интерпретатор берет очередной оператор языка из текста программы, анализирует его структуру и затем сразу исполняет (обычно после анализа оператор транслируется в некоторое промежуточное представление или даже машинный код для более эффективного дальнейшего исполнения). Только после того, как текущий оператор успешно выполнен, интерпретатор перейдет к следующему. При этом, если один и тот же оператор должен выполняться в программе многократно, интерпретатор всякий раз будет выполнять его так, как будто встретил впервые. Вследствие этого, программы, в которых требуется осуществить большой объем повторяющихся вычислений, могут работать медленно. Кроме того, для выполнения такой программы на другом компьютере там также должен быть установлен интерпретатор — ведь без него текст программы является просто набором символов.

По-другому можно сказать, что интерпретатор моделирует некую виртуальную вычислительную машину, для которой базовыми инструкциями служат не элементарные команды процессора, а операторы языка программирования.

Компиляторы полностью обрабатывают весь текст программы (он иногда называется *исходный код*). Они просматривают его в поисках синтаксических ошибок (иногда несколько раз), выполняют определенный смысловой анализ и затем автоматически переводят (*транслируют*) на машинный язык — генерируют машинный код. Нередко при этом выполняется *оптимизация* с помощью набора методов, позволяющих повысить быстродействие программы (например, с помощью инструкций, ориентированных на конкретный процессор, путем исключения ненужных команд, промежуточных вычислений и т. д.). В результате законченная программа получается компактной и эффективной, работает в сотни раз быстрее программы, выполняемой с помощью интерпретатора, и может быть перенесена на другие компьютеры с процессором, поддерживающим соответствующий машинный код.

Основной недостаток компиляторов — трудоемкость трансляции языков программирования, ориентированных на обработку данных сложной структуры, часто заранее неизвестной или динамически меняющейся во время работы программы. Тогда в машинный код приходится вставлять множество дополнительных проверок, анализировать наличие ресурсов операционной системы, динамически их захватывать и освобождать, формировать и обрабатывать в памяти компьютера сложные объекты, что на уровне жестко заданных машинных инструкций осуществить довольно трудно, а для ряда задач практически невозможно.

С помощью интерпретатора, наоборот, допустимо в любой момент остановить работу программы, исследовать содержимое памяти, организовать диалог с пользователем, выполнить сколь угодно сложные преобразования данных и при этом постоянно контролировать состояние окружающей программно-аппаратной среды, благодаря чему достигается высокая надежность работы. Интерпретатор при выполнении каждого оператора проверяет множество характеристик операцион-

ной системы и при необходимости максимально подробно информирует разработчика о возникающих проблемах. Кроме того, интерпретатор очень удобен для использования в качестве инструмента изучения программирования, так как позволяет понять принципы работы любого отдельного оператора языка.

В реальных системах программирования перемешаны технологии и компиляции, и интерпретации. В процессе отладки программа может выполняться по шагам, а результирующий код не обязательно будет машинным — он даже может быть исходным кодом, написанным на другом языке программирования (это существенно упрощает процесс трансляции, но требует компилятора для конечного языка), или промежуточным машинно-независимым кодом абстрактного процессора, который в различных компьютерных архитектурах станет выполняться с помощью интерпретатора или компилироваться в соответствующий машинный код.

Уровни языков программирования

Разные типы процессоров имеют разные наборы команд. Если язык программирования ориентирован на конкретный тип процессора и учитывает его особенности, то он называется *языком программирования низкого уровня*. В данном случае «низкий уровень» не значит «плохой». Имеется в виду, что операторы языка близки к машинному коду и ориентированы на конкретные команды процессора.

Языком самого низкого уровня является *язык ассемблера*, который просто представляет каждую команду машинного кода, но не в виде чисел, а с помощью символьных условных обозначений, называемых *мнемониками*. Однозначное преобразование одной машинной инструкции в одну команду ассемблера называется *транслитерацией*. Так как наборы инструкций для каждого модели процессора отличаются, конкретной компьютерной архитектуре соответствует свой язык ассемблера, и написанная на нем программа может быть использована только в этой среде.

С помощью языков низкого уровня создаются очень эффективные и компактные программы, так как разработчик получает доступ ко всем возможностям процессора. С другой стороны, при этом требуется очень хорошо понимать устройство компьютера, затрудняется отладка больших приложений, а результирующая программа не может быть перенесена на компьютер с другим типом процессора. Подобные языки обычно применяют для написания небольших системных приложений, драйверов устройств, модулей стыковки с нестандартным оборудованием, когда важнейшими требованиями становятся компактность, быстродействие и возможность прямого доступа к аппаратным ресурсам. В некоторых областях, например в машинной графике, на языке ассемблера пишутся библиотеки, эффективно реализующие требующие интенсивных вычислений алгоритмы обработки изображений.

Языки программирования высокого уровня значительно ближе и понятнее человеку, нежели компьютеру. Особенности конкретных компьютерных архитектур в них не учитываются, поэтому создаваемые программы на уровне исходных текстов легко переносимы на другие платформы, для которых создан транслятор этого языка. Разрабатывать программы на языках высокого уровня с помощью понятных и мощных команд значительно проще, а ошибок при создании программ допускается гораздо меньше.

Поколения языков программирования

Языки программирования принято делить на пять *поколений*. В первое поколение входят языки, созданные в начале 50-х годов, когда первые компьютеры только появились на свет. Это был первый язык ассемблера, созданный по принципу «одна инструкция — одна строка».

Расцвет второго поколения языков программирования пришелся на конец 50-х — начало 60-х годов. Тогда был разработан символический ассемблер, в котором появилось понятие переменной. Он стал первым полноценным языком программирования. Благодаря его возникновению заметно возросли скорость разработки и надежность программ.

Появление третьего поколения языков программирования принято относить к 60-м годам. В это время родились универсальные языки высокого уровня, с их помощью удается решать задачи из любых областей. Такие качества новых языков, как относительная простота, независимость от конкретного компьютера и возможность использования мощных синтаксических конструкций, позволили резко повысить производительность труда программистов. Понятная большинству пользователей структура этих языков привлекла к написанию небольших программ (как правило, инженерного или экономического характера) значительное число специалистов из некомпьютерных областей. Подавляющее большинство языков этого поколения успешно применяется и сегодня.

С начала 70-х годов по настоящее время продолжается период языков четвертого поколения. Эти языки предназначены для реализации крупных проектов, повышения их надежности и скорости создания. Они обычно ориентированы на специализированные области применения, где хороших результатов можно добиться, используя не универсальные, а проблемно-ориентированные языки, оперирующие конкретными понятиями узкой предметной области. Как правило, в эти языки встраиваются мощные операторы, позволяющие одной строкой описать такую функциональность, для реализации которой на языках младших поколений потребовались бы тысячи строк исходного кода.

Рождение языков пятого поколения произошло в середине 90-х годов. К ним относятся также системы автоматического создания прикладных программ с помощью визуальных средств разработки, без знания программирования. Главная идея, которая закладывается в эти языки, — возможность автоматического формирования результирующего текста на универсальных языках программирования (который потом требуется откомпилировать). Инструкции же вводятся в компьютер в максимально наглядном виде с помощью методов, наиболее удобных для человека, не знакомого с программированием.

Обзор языков программирования высокого уровня

FORTRAN (Фортран). Это первый компилируемый язык, созданный Джимом Бэкусом в 50-е годы. Программисты, разрабатывавшие программы исключительно на ассемблере, выражали серьезное сомнение в возможности появления высокопроизводительного языка высокого уровня, поэтому основным критерием при раз-

работке компиляторов Фортрана являлась эффективность исполняемого кода. Хотя в Фортране впервые был реализован ряд важнейших понятий программирования, удобство создания программ было принесено в жертву возможности получения эффективного машинного кода. Однако для этого языка было создано огромное количество библиотек, начиная от статистических комплексов и кончая пакетами управления спутниками, поэтому Фортран продолжает активно использоваться во многих организациях, а сейчас ведутся работы над очередным стандартом Фортрана *F2k*, который появится в 2000 году. Имеется стандартная версия Фортрана *HPF* (*High Performance Fortran*) для параллельных суперкомпьютеров со множеством процессоров.

COBOL (Кобол). Это компилируемый язык для применения в экономической области и решения бизнес-задач, разработанный в начале 60-х годов. Он отличается большой «многословностью» — его операторы иногда выглядят как обычные английские фразы. В Коболе были реализованы очень мощные средства работы с большими объемами данных, хранящимися на различных внешних носителях. На этом языке создано очень много приложений, которые активно эксплуатируются и сегодня. Достаточно сказать, что наибольшую зарплату в США получают программисты на Коболе.

Algol (Алгол). Компилируемый язык, созданный в 1960 году. Он был призван заменить Фортран, но из-за более сложной структуры не получил широкого распространения. В 1968 году была создана версия Алгол 68, по своим возможностям и сегодня опережающая многие языки программирования, однако из-за отсутствия достаточно эффективных компьютеров для нее не удалось своевременно создать хорошие компиляторы.

Pascal (Паскаль). Язык Паскаль, созданный в конце 70-х годов основоположником множества идей современного программирования Никлаусом Виртом, во многом напоминает Алгол, но в нем ужесточен ряд требований к структуре программы и имеются возможности, позволяющие успешно применять его при создании крупных проектов.

Basic (Бейсик). Для этого языка имеются и компиляторы, и интерпретаторы, а по популярности он занимает первое место в мире. Он создавался в 60-х годах в качестве учебного языка и очень прост в изучении.

C (Си). Данный язык был создан в лаборатории *Bell* и первоначально не рассматривался как массовый. Он планировался для замены ассемблера, чтобы иметь возможность создавать столь же эффективные и компактные программы и в то же время не зависеть от конкретного типа процессора.

Си во многом похож на Паскаль и имеет дополнительные средства для прямой работы с памятью (*указатели*). На этом языке в 70-е годы написано множество прикладных и системных программ и ряд известных операционных систем (*Unix*).

C++ (Си++). Си++ — это объектно-ориентированное расширение языка Си, созданное Бьярном Страуструпом в 1980 году. Множество новых мощных возможностей, позволивших резко повысить производительность программистов, наложи-

лось на унаследованную от языка Си определенную низкоуровневость, в результате чего создание сложных и надежных программ потребовало от разработчиков высокого уровня профессиональной подготовки.

Java (Джава, Ява). Этот язык был создан компанией *Sun* в начале 90-х годов на основе Си++. Он призван упростить разработку приложений на основе Си++ путем исключения из него всех низкоуровневых возможностей. Но главная особенность этого языка — компиляция не в машинный код, а в платформно-независимый байт-код (каждая команда занимает один байт). Этот байт-код может выполняться с помощью интерпретатора — виртуальной *Java*-машины *JVM* (*Java Virtual Machine*), версии которой созданы сегодня для любых платформ. Благодаря наличию множества *Java*-машин программы на *Java* можно переносить не только на уровне исходных текстов, но и на уровне двоичного байт-кода, поэтому по популярности язык Ява сегодня занимает второе место в мире после Бейсика.

Особое внимание в развитии этого языка уделяется двум направлениям: поддержке всевозможных мобильных устройств и микрокомпьютеров, встраиваемых в бытовую технику (технология *Jini*) и созданию платформно-независимых программных модулей, способных работать на серверах в глобальных и локальных сетях с различными операционными системами (технология *Java Beans*). Пока основной недостаток этого языка — невысокое быстродействие, так как язык Ява интерпретируемый.

C# (Си Шарп). В конце 90-х годов в компании *Microsoft* под руководством Андерса Хейльсберга был разработан язык C#. В нем воплотились лучшие идеи Си и Си++, а также достоинства *Java*. Правда, C#, как и другие технологии *Microsoft*, ориентирован на платформу *Windows*. Однако формально он не отличается от прочих универсальных языков, а корпорация даже планирует его стандартизацию. Язык C# предназначен для быстрой разработки *.NET*-приложений, и его реализация в системе *Microsoft Visual Studio .NET* содержит множество особенностей, привязывающих C# к внутренней архитектуре *Windows* и платформы *.NET*.

Языки программирования баз данных

Эта группа языков отличается от алгоритмических языков прежде всего решаемыми задачами. База данных — это файл (или группа файлов), представляющий собой упорядоченный набор *записей*, имеющих единообразную структуру и организованных по единому шаблону (как правило, в табличном виде). База данных может состоять из нескольких таблиц. Удобно хранить в базах данных различные сведения из справочников, картотек, журналов бухгалтерского учета и т. д.

При работе с базами данных чаще всего требуется выполнять следующие операции:

- создание/модификация свойств/удаление таблиц в базе данных;
- поиск, отбор, сортировка информации по запросам пользователей;
- добавление новых записей;
- модификация существующих записей;
- удаление существующих записей.

Первые базы данных появились очень давно, как только появилась потребность в обработке больших массивов информации и выборки групп записей по определенным признакам. Для этого был создан *структурированный язык запросов SQL* (*Structured Query Language*). Он основан на мощной математической теории и позволяет выполнять эффективную обработку баз данных, манипулируя не отдельными записями, а *группами* записей.

Для управления большими базами данных и их эффективной обработки разработаны СУБД (Системы Управления Базами Данных). Практически в каждой СУБД помимо поддержки языка *SQL* имеется также свой уникальный язык, ориентированный на особенности этой СУБД и не переносимый на другие системы. Сегодня в мире насчитывается три ведущих производителя СУБД: *Microsoft* (*SQL Server*), *IBM* (*DB2*) и *Oracle*. Их продукты нацелены на поддержку одновременной работы тысяч пользователей в сети, а базы данных могут храниться в распределенном виде на нескольких серверах. В каждой из этих СУБД реализован собственный диалект *SQL*, ориентированный на особенности конкретного сервера, поэтому *SQL*-программы, подготовленные для разных СУБД, друг с другом, как правило, несовместимы.

С появлением персональных компьютеров были созданы так называемые настольные СУБД. Родоначальником современных языков программирования баз данных для ПК принято считать СУБД *dBase II*, язык которой был интерпретируемым. Затем для него были созданы компиляторы, появились СУБД *FoxPro* и *Clipper*, поддерживающие диалекты этого языка. Сегодня самой распространенной настольной СУБД стала система *Microsoft Access*.

Языки программирования для Интернета

С активным развитием глобальной сети было создано немало реализаций популярных языков программирования, адаптированных специально для Интернета. Все они отличаются характерными особенностями: языки являются интерпретируемыми, интерпретаторы для них распространяются бесплатно, а сами программы — в исходных текстах. Такие языки называют *скрипт-языками*.

HTML. Общеизвестный язык для оформления документов. Он очень прост и содержит элементарные команды форматирования текста, добавления рисунков, задания шрифтов и цветов, организации ссылок и таблиц. Все *Web*-страницы написаны на языке *HTML* или используют его расширения.

Perl. В 80-х годах Ларри Уолл разработал язык *Perl*. Он задумывался как средство эффективной обработки больших текстовых файлов, генерации текстовых отчетов и управления задачами. По мощности *Perl* значительно превосходит языки типа Си. В него введено много часто используемых функций работы со строками, массивами, всевозможные средства преобразования данных, управления процессами, работы с системной информацией и др.

PHP. Расмус Лердорф, активно использовавший *Perl*-скрипты, в 1995 году решил улучшить этот язык, упростив его и дополнив встроенными средствами доступа к базам данных. В результате появилась разработка *Personal Contents Page/Forms Interpreter* (*PHP/FI*). Уже через пару лет программы на ее основе использовались на 50 тыс. сайтов. В 1997 году ее значительно усовершенствовали Энди Гутманс и

Зив Сураски, и под названием *PHP* 3.0 этот язык быстро завоевал популярность у создателей динамических сайтов во всем мире.

Tcl/Tk. В конце 80-х годов Джон Аустираут придумал популярный скрипт-язык *Tcl* и библиотеку *Tk*. В *Tcl* он попытался воплотить видение идеального скрипт-языка. Язык *Tcl* ориентирован на автоматизацию рутинных процессов и состоит из мощных команд, предназначенных для работы с абстрактными нетипизированными объектами. Он независим от типа системы и при этом позволяет создавать программы с графическим интерфейсом.

VRML. В 1994 году был создан язык *VRML* для организации виртуальных трехмерных интерфейсов в Интернете. Он позволяет описывать в текстовом виде различные трехмерные сцены, освещение и тени, текстуры (покрытия объектов), создавать свои миры, путешествовать по ним, «облетать» со всех сторон, вращать в любых направлениях, масштабировать, регулировать освещенность и т. д.

XML. В августе 1996 года *WWW*-консорциум, ответственный за стандарты на Интернет-технологии, приступил к подготовке универсального языка разметки структуры документов, базировавшегося на достаточно давно созданной в *IBM* технологии *SGML*. Новый язык получил название *XML*. Сегодня он служит основой множества системных, сетевых и прикладных приложений, позволяя представлять в прозрачном для пользователей и программ текстовом виде различные аспекты внутренней структуры иерархически организованных документов. В недалеком будущем он может стать заменой *HTML*.

Языки моделирования

При создании программ и формировании структур баз данных нередко применяются формальные способы их представления — *формальные нотации*, с помощью которых можно визуально представить (изобразить с помощью мыши) таблицы баз данных, поля, объекты программы и взаимосвязи между ними в системе, имеющей специализированный редактор и генератор исходных текстов программ на основе созданной модели. Такие системы называются *CASE-системами*. В них активно применяются нотации *IDEF*, а в последнее время все большую популярность завоевывает язык графического моделирования *UML*.

Прочие языки программирования

PL/I (ПЛ/1). В середине 60-х годов компания *IBM* решила взять все лучшее из языков Фортран, Кобол и Алгол. В результате в 1964 году на свет появился новый компилируемый язык программирования, который получил название *Programming Language One*. В этом языке было реализовано множество уникальных решений, полезность которых удается оценить только спустя 33 года, в эпоху крупных программных систем. По своим возможностям ПЛ/1 значительно мощнее многих других языков (Си, Паскаля). Например, в ПЛ/1 присутствует уникальная возможность указания точности вычислений — ее нет даже у Си++ и Явы. Этот язык и сегодня продолжает поддерживаться компанией *IBM*.

Smalltalk (Смолток). Работа над этим языком началась в 1970 году в исследовательской лаборатории корпорации *XEROX*, а закончились спустя 10 лет, воплотив-

шись в окончательном варианте интерпретатора *SMALLTALK-80*. Данный язык оригинален тем, что его синтаксис очень компактен и базируется исключительно на понятии объекта. В этом языке отсутствуют операторы или данные. Все, что входит в Смолток, является объектами, а сами объекты общаются друг с другом исключительно с помощью сообщений (например, появление выражения I + 1 вызывает посылку объекту I сообщения «+», то есть «прибавить», с параметром 1, который считается не числом-константой, а тоже объектом). Больше никаких управляющих структур, за исключением «оператора» ветвления (на самом деле функции, принадлежащей стандартному объекту), в языке нет, хотя их можно очень просто смоделировать. Сегодня версия *VisualAge for Smalltalk* активно развивается компанией *IBM*.

LISP (Лисп). Интерпретируемый язык программирования, созданный в 1960 году Джоном Маккарти. Ориентирован на структуру данных в форме списка и позволяет организовывать эффективную обработку больших объемов текстовой информации.

Prolog (Пролог). Создан в начале 70-х годов Аланом Колмероэ. Программа на этом языке, в основу которого положена математическая модель теории исчисления предикатов, строится из последовательности фактов и правил, а затем формулируется утверждение, которое Пролог будет пытаться доказать с помощью введенных правил. Человек только описывает структуру задачи, а внутренний «мотор» Пролога сам ищет решение с помощью методов поиска и сопоставления.

Ada (Ада). Назван по имени леди Огасты Ады Байрон, дочери английского поэта Байрона и его отдаленной родственницы Анабеллы Милбэнк. В 1980 году сотни экспертов Министерства обороны США отобрали из 17 вариантов именно этот язык, разработанный небольшой группой под руководством Жана Ишбиа. Он удовлетворил на то время все требования Пентагона, а к сегодняшнему дню в его развитие вложены десятки миллиардов долларов. Структура самого языка похожа на Паскаль. В нем имеются средства строгого разграничения доступа к различным уровням спецификаций, доведена до предела мощность управляющих конструкций.

Forth (Форт). Результат попытки Чарльза Мура в 70-х годах создать язык, обладающий мощными средствами программирования, который можно эффективно реализованным на компьютерах с небольшими объемами памяти, а компилятор мог бы выдавать очень быстрый и компактный код, то есть служил заменой ассемблеру. Однако сложности восприятия программного текста, записанного в непривычной форме, сильно затрудняли поиск ошибок, и с появлением Си язык Форт оказался забытым.

Вопросы для самоконтроля

1. Что такое язык программирования?
2. В чем различие компиляторов и интерпретаторов?
3. Объясните термины «язык низкого уровня» и «язык высокого уровня».
4. Расскажите о поколениях языков программирования.
5. Какие языки программирования активно используются сегодня?

20.2. Системы программирования

Средства создания программ

В самом общем случае для создания программы на выбранном языке программирования нужно иметь следующие компоненты.

1. *Текстовый редактор.* Так как текст программы записывается с помощью ключевых слов, обычно происходящих от слов английского языка, и набора стандартных символов для записи всевозможных операций, то формировать этот текст можно в любом редакторе, получая в итоге текстовый файл с *исходным текстом* программы. Лучше использовать специализированные редакторы, которые ориентированы на конкретный язык программирования и позволяют в процессе ввода текста выделять ключевые слова и идентификаторы разными цветами и шрифтами. Подобные редакторы созданы для всех популярных языков и дополнительно могут автоматически проверять правильность синтаксиса программы непосредственно во время ее ввода.
2. Исходный текст с помощью *программы-компилятора* переводится в машинный код. Если обнаружены синтаксические ошибки, то результирующий код создан не будет.

   На этом этапе уже возможно получение готовой программы, но чаще всего в ней не хватает некоторых компонентов, поэтому компилятор обычно выдает промежуточный *объектный код* (двоичный файл, стандартное расширение .OBJ).
3. Исходный текст большой программы состоит, как правило, из нескольких *модулей* (файлов с исходными текстами), потому что хранить все тексты в одном файле неудобно — в них сложно ориентироваться. Каждый модуль компилируется в отдельный файл с объектным кодом, которые затем надо объединить в одно целое.

   Кроме того, к ним надо добавить машинный код подпрограмм, реализующих различные стандартные функции (например, вычисляющих математические функции *sin* или *ln*). Такие функции содержатся в *библиотеках* (файлах со стандартным расширением .LIB), которые поставляются вместе с компилятором. Сгенерированный код модулей и подключенные к нему стандартные функции надо не просто объединить в одно целое, а выполнить такое объединение с учетом требований операционной системы, то есть получить на выходе программу, отвечающую определенному формату.

   Объектный код обрабатывается специальной программой — *редактором связей* или *сборщиком*, который выполняет связывание объектных модулей и машинного кода стандартных функций, находя их в библиотеках, и формирует на выходе работоспособное приложение — *исполнимый код* для конкретной платформы.

   Если по каким-то причинам один из объектных модулей или нужная библиотека не обнаружены (например, неправильно указан каталог с библиотекой), то сборщик сообщает об ошибке и готовой программы не получается.

4. Исполнимый код — это законченная программа, которую можно запустить на любом компьютере, где установлена операционная система, для которой эта программа создавалась. Как правило, итоговый файл имеет расширение .EXE.

Интегрированные системы программирования

Итак, для создания программы нужны:

- текстовый редактор;
- компилятор;
- редактор связей;
- библиотеки функций.

Как правило, в стандартную поставку входят как минимум три последних компонента, но хорошая *интегрированная система* включает в себя и специализированный текстовый редактор, причем почти все этапы создания программы в ней автоматизированы: после того как исходный текст введен, его компиляция и сборка выполняются одним нажатием клавиши. Это очень удобно, так как не требует ручной настройки множества параметров запуска компилятора и редактора связей, указывания им нужных файлов вручную и т. д. Процесс компиляции обычно демонстрируется на экране: показывается, сколько строк исходного текста откомпилировано, или выдаются сообщения о найденных ошибках.

В современных интегрированных системах имеется еще один компонент — *отладчик*, который позволяет анализировать работу программы во время ее выполнения. С его помощью можно последовательно выполнять отдельные операторы исходного текста *по шагам*, наблюдая при этом, как меняются значения различных переменных. Без отладчика разработать крупное приложение очень сложно.

Среды быстрого проектирования

В последние несколько лет в программировании (особенно в программировании для операционной системы *Windows*) наметился так называемый *визуальный подход*. До этого серьезным препятствием для разработки графических приложений была сложность создания различных элементов управления и контроля их работы. Достаточно взглянуть на окно любой *Windows*-программы. В нем имеется множество стандартных элементов управления (кнопки, пункты меню, списки, переключатели и т. д.). Очень трудоемко вручную описывать процесс создания этих элементов в соответствии с требованиями *Windows*, на глазок определять координаты, отслеживать их состояние с помощью специальных команд. Например, для простой программы, складывающей два числа, потребуется один оператор (одна строка исходного текста) для выполнения нужного вычисления и сотни строк кода для подготовки приложения к работе в *Windows*, создания кнопки и пары полей ввода.

Этот процесс автоматизирован в *средах быстрого проектирования* (*Rapid Application Development*, *RAD-среды*). Все необходимые элементы оформления и управления создаются и обслуживаются не путем ручного программирования, а с помощью готовых визуальных *компонентов*, которые с помощью мыши «перетаскиваются» в проектируемое окно. Их свойства и поведение затем настраиваются с помощью

простых редакторов, визуально показывающих характеристики соответствующих элементов. При этом вспомогательный исходный текст программы, ответственный за создание и работу этих элементов, генерируется *RAD*-средой автоматически, что позволяет сосредоточиться только на логике решаемой задачи. В результате программирование во многом заменяется на проектирование — подобный подход называется еще *визуальным программированием.*

Компоненты достаточно легко создавать самостоятельно, поэтому в мире сегодня распространяются тысячи бесплатных и платных компонентов для наиболее известных *RAD*-сред, из них формируются библиотеки компонентов — *объектные репозитории.* Компоненты выступают в роли «строительных кирпичиков», позволяющих собирать готовое приложение с богатыми возможностями, написав всего десяток строк исходного кода, и такой *компонентный подход* к созданию программ считается очень перспективным, потому что без лишних усилий и на законных основаниях допускает *повторное использование* чужого труда.

Архитектура программных систем

В то время как большинство автономных приложений: офисные программы, среды разработки, системы подготовки текстов и изображений — выполняются на одном компьютере, крупные информационные комплексы (например, система автоматизации предприятия) состоят из десятков и сотен отдельных программ, которые взаимодействуют друг с другом по сети, выполняясь на разных компьютерах. В таких случаях говорят, что они работают в различной *программной архитектуре.* Она делится на следующие группы.

Автономные приложения. Работают на одном компьютере.

Приложения в файл-серверной архитектуре. Компьютеры пользователей системы объединены в сеть, при этом на каждом из них (на *клиентском месте*) запущены копии одной и той же программы, которые обращаются за данными к *серверу* — специальному компьютеру, который хранит файлы, одновременно доступные всем пользователям (как правило, это базы данных). Сервер обладает повышенной надежностью, высоким быстродействием, большим объемом памяти, на нем установлена специальная *серверная* версия операционной системы.

При одновременном обращении нескольких программ к одному файлу, например, с целью его обновления, могут возникнуть проблемы, связанные с неоднозначностью определения его содержимого. Поэтому каждое изменение общедоступного файла выделяется в *транзакцию* — элементарную операцию по обработке данных, имеющую фиксированные начало, конец (успешное или неуспешное завершение) и ряд других характеристик.

Особенность этой архитектуры в том, что все вычисления выполняются на клиентских местах, что требует наличия на них достаточно производительных ПК (это так называемые системы с *толстым клиентом* — программой, которая выполняет всю обработку получаемой от сервера информации).

Приложения в клиент-серверной архитектуре. Эта архитектура похожа на предыдущую, только сервер помимо простого обеспечения одновременного доступа к дан-

ным способен еще выполнять программы (обычно выполняются СУБД — тогда сервер называется *сервером баз данных*), которые берут на себя определенный объем вычислений (в файл-серверной архитектуре он реализуется полностью на клиентских местах). Благодаря этому удается повысить общую надежность системы, так как сервер работает значительно более устойчиво, чем ПК, и снять лишнюю нагрузку с клиентских мест, на которых удается использовать дешевые компьютеры. Запускаемые на них приложения реально осуществляют небольшие объемы вычислений, а иногда занимаются только отображением получаемой от сервера информации, поэтому они называются *тонкими клиентами*.

Приложения в многозвенной архитектуре. Недостаток предыдущей архитектуры в том, что резко возрастает нагрузка на сервер, а если он выходит из строя, то работа всей системы останавливается. Поэтому в некоторых случаях в систему добавляется так называемый *сервер приложений*, на котором выполняется вся вычислительная работа. Другой сервер баз данных обрабатывает запросы пользователей, на третьем может быть установлена специальная программа — *монитор транзакций*, которая оптимизирует обработку транзакций и балансирует нагрузку на серверы. В большинстве практических случаев все серверы соединены последовательно — *позвенно*, и выход из строя одного звена если и не останавливает всю работу, то, по крайней мере, резко снижает производительность системы.

Приложения в распределенной архитектуре. Чтобы избежать недостатков рассмотренных архитектур, были придуманы специальные технологии, позволяющие создавать программу в виде набора компонентов, которые можно запускать на любых серверах, связанных в сеть (компоненты как бы распределены по сети). Основное преимущество подобного подхода в том, что при выходе из строя любого компьютера специальные *программы-мониторы*, которые следят за корректностью работы компонентов и позволяют им «переговариваться» между собой, сразу перезапуская временно пропавший компонент на другом компьютере. При этом общая надежность всей системы становится очень высокой, а вычислительная загрузка распределяется между серверами оптимальным образом.

Доступ к возможностям любого компонента, предназначенного для общения с пользователем, осуществляется с произвольного клиентского места. При этом, так как все вычисления происходят на серверах, появляется возможность создавать *сверхтонкие клиенты* — программы, только отображающие получаемую из сети информацию и требующие минимальных компьютерных ресурсов. Благодаря этому доступ к компонентной системе возможен не только с ПК, но и с небольших мобильных устройств.

Частный случай компонентного подхода — доступ к серверным приложениям из браузеров через Интернет.

Сегодня наиболее популярны три компонентные технологии — *CORBA* консорциума *OMG*, *Java Beans* компании *Sun* и *COM+/.NET* корпорации *Microsoft*. Эти технологии будут определять развитие информационной индустрии в ближайшие десятилетия.

Основные системы программирования

Из универсальных языков программирования сегодня наиболее популярны следующие:

- Бейсик (*Basic*) — для освоения требует начальной подготовки (общеобразовательная школа);
- Паскаль (*Pascal*) — требует специальной подготовки (школы с углубленным изучением предмета и общетехнические вузы);
- Си++ (C++), Ява (*Java*), Си Шарп (C#) — требуют профессиональной подготовки (специализированные средние и высшие учебные заведения).

Для каждого из этих языков программирования сегодня имеется немало систем программирования, выпускаемых различными фирмами и ориентированных на различные модели ПК и операционные системы. Наиболее популярны следующие визуальные среды быстрого проектирования программ для *Windows*:

- *Basic*: *Microsoft Visual Basic*;
- *Pascal*: *Borland Delphi*;
- *C++*: *Microsoft Visual C++*;
- *Java*: *Borland JBuilder*;
- *C#*: *Microsoft Visual Studio .NET*, *Borland C#Builder*.

Для разработки серверных и распределенных приложений можно использовать систему программирования *Microsoft Visual C++*, продукты фирмы *Borland*, практически любые средства программирования на *Java*.

В дальнейшем будут рассматриваться возможности, характерные для Бейсика, Паскаля и Си++.

Вопросы для самоконтроля

1. Что нужно для создания программы?
2. Что такое среды быстрого проектирования?
3. Объясните понятие «архитектура программной системы».
4. Опишите основные типы программных архитектур.
5. Какая программная архитектура обеспечивает работу Интернета?

20.3. Алгоритмическое (модульное) программирование

Алгоритм — это формальное описание способа решения задачи путем разбиения ее на конечную по времени последовательность действий (элементарных операций). Под словом «формальное» подразумевается, что описание должно быть абсолютно полным и учитывать все возможные ситуации, которые могут встретиться по ходу решения. Под элементарной операцией понимается действие, которое по заранее определенным критериям (например, очевидности) не имеет смысла детализировать.

Основная идея алгоритмического программирования — разбиение программы на последовательность модулей, каждый из которых выполняет одно или несколько

действий. Единственное требование к модулю — чтобы его выполнение всегда начиналось с первой команды и всегда заканчивалось на самой последней (то есть, чтобы нельзя было попасть на команды модуля извне и передать управление из модуля на другие команды в обход заключительной).

Алгоритм на выбранном языке программирования записывается с помощью команд описания данных, вычисления значений и управления последовательностью выполнения программы.

Переменные и константы

Реальные данные, с которыми работает программа, — это *числа*, *строки* и *логические величины* (аналоги 1 и 0, «да» и «нет», «истина» и «ложь»). Эти типы данных называют *базовыми*.

Каждая единица информации хранится в ячейках памяти компьютера, имеющих свои адреса. На практике заранее неизвестно, в каких конкретно ячейках памяти во время работы программы будут записаны ее данные, поэтому в языках программирования введено понятие переменной, позволяющее отвлечься от конкретных адресов и обращаться к содержимому памяти с помощью *идентификатора* или *имени* — как правило, последовательности, содержащей английские буквы, цифры, символы подчеркивания и начинающейся не с цифры. Например:

```
Hello
_SumOfReal
x1
H8_G7_F6
```

Это имя будет указывать на *значение*, о реальном адресе и способе хранения которого можно забыть. В процессе работы программы содержимое соответствующих ячеек можно менять, обращаясь к переменной по имени. Лучше выбирать такие названия, которые отражают назначение данной переменной.

Кроме имени и значения, переменная обычно имеет *тип*, определяющий, какая информация хранится в данной переменной (число, строка и т. д.). В зависимости от объема памяти, отведенного для хранения значения переменной, оно должно укладываться в *допустимый диапазон*. Например, значение типа «байт» имеет диапазон от 0 до 255.

Переменные с указанием их типа можно вводить в программу с помощью специальных команд *описания* (*объявления*, *декларации*). Это позволяет компилятору организовать эффективное хранение и обработку данных и повышает ясность исходных тестов. Каждый тип описывается своим ключевым словом. Значения переменных разных типов допускается преобразовывать друг в друга в соответствии с соглашениями языка программирования. Такой процесс называется *приведением типов*.

Переменные могут существовать на всем протяжении работы программы — тогда они называются *статическими*, а могут создаваться и уничтожаться на разных этапах ее функционирования — такие переменные называются *динамическими*. Все

остальные данные в программе, значение которых не меняется на протяжении ее работы, называются *константами* или *постоянными*. Константы, как и переменные, обычно имеют тип. Данные можно указывать явно:

```
123
2.87
"это строка"
```

или для удобства обозначать их идентификаторами. Например, число π, равное 3,1416, можно обозначить как pi и везде вместо числа применять идентификатор. Только изменять значение pi нельзя, так как это не переменная, а константа.

Числовые данные

Числа обычно бывают двух видов: *целые* и *дробные*. Если число отрицательное, перед ним ставится знак «–»; если положительное, то знак «+» можно ставить, а можно и опускать. Вычисления над целыми числами выполняются *точно*, вычисления над дробными числами — *приближенно*. При записи дробных чисел в качестве десятичного разделителя используется точка:

```
1.28
3.333321
```

Очень большие или очень маленькие числа записываются специальным образом. Для них дополнительно указывается *мантисса* — число со знаком, являющееся степенью числа 10. Мантисса записывается справа от числа через букву e (или E). Пробелы в такой записи не допускаются.

Например, число 100 (единица, умноженная на 10 во второй степени) запишется так:

```
1e+2
```

число 0,003 (тройка, умноженная на 10 в минус третьей степени) так:

```
3e-3
```

число со 120 нулями — так:

```
1E+120
```

Допускается дробная запись числа с мантиссой:

```
31.4e-1
```

Тип числа	Бейсик	Паскаль	Си++
целое	INTEGER	integer	int
дробное	DOUBLE	real	float

Арифметические операции

Для записи арифметических действий используются арифметические операторы. В некоторых языках программирования они считаются не операторами, а *опера-*

циями, предназначенными для вычисления значения выражения, но не влияющими на другие значения и не сказывающимися на ходе выполнения программы.

К основным арифметическим операциям относятся:

+ (сложение)
- (вычитание)
* (умножение)
/ (деление)

Такая форма записи отвечает общепринятым соглашениям и принята в большинстве языков программирования.

Каждая арифметическая операция имеет свой *приоритет*. Операции с более высоким приоритетом (умножение и деление) будут выполняться раньше, чем операции с более низким приоритетом (сложение и вычитание). Изменить порядок вычисления выражения можно с помощью круглых скобок.

```
b*2 + c/3
b*(2 + c) - 3
```

Скобки допускается вкладывать друг в друга произвольное число раз. При этом использование квадратных или фигурных скобок, как правило, не допускается.

```
((y+2)*3 + 1) / 2
```

Арифметические выражения

С помощью арифметических операций формируются арифметические выражения, которые состоят из операций и *операндов* (переменных и констант).

Выражение

```
i1 + 2
```

состоит из одной операции «+» и двух операндов — переменной i1 и числовой константы 2.

Каждое выражение имеет значение, которое определяется в момент выполнения оператора, содержащего это выражение. Если на момент вычисления выражения i1+2 в переменной i1 хранится число 3, то значение этого выражения будет равно 5 (3+2).

Логические выражения

При создании программ не обойтись без *логических выражений*. Они отличаются тем, что результат их вычислений может принимать только одно из двух допустимых значений — true (истина, да, включено) и false (ложь, нет, выключено). Чаще всего значение false ассоциируется с нулем, а значение true — с числом 1 или просто ненулевым значением.

При записи логических выражений используются *операции сравнения* и *логические операции*. Операции сравнения сличают значения правого и левого операндов. Результатом сравнения является true, если оно удачно, и false в противном случае.

В таблице даны примеры записи операций сравнения для разных языков.

Операция	Варианты написания	
	Бейсик, Паскаль	Си++
Равно	=	==
Не равно	<>	!=
Меньше	<	<
Меньше или равно	<=	<=
Больше	>	>
Больше или равно	>=	>=

```
Pi == 3.14
x > 0
a1 <> b1
```

В одном выражении может потребоваться проверка нескольких подобных условий. Например, надо определить, больше ли значение переменной X, чем 0 и меньше ли, чем 10. Условия могут быть связаны с помощью логических операций, наиболее активно используемые из которых — это И и ИЛИ. В компьютерной графике также часто применяется так называемое исключающее ИЛИ и операция отрицания НЕ. Для нее требуется только один операнд, указывающийся справа от знака операции. Эта операция просто меняет значение своего операнда на противоположное.

1 операнд	2 операнд	И	ИЛИ	исключающее ИЛИ	НЕ (только первый операнд)
true	true	true	true	false	false
true	false	false	true	true	false
false	true	false	true	true	true
false	false	false	false	false	true

В следующей таблице приведен синтаксис записи логических операций.

Логическая операция	Бейсик	Паскаль	Си++
И	AND	and	&&
ИЛИ	OR	or	||
НЕ	NOT	not	!

Приоритеты всех логических операций ниже, чем приоритеты операций сравнения, поэтому сравнения всегда выполняются первыми. А логические операции вычисляются в следующем порядке: сначала НЕ, потом И, потом ИЛИ. При необходимости этот порядок может быть изменен с помощью скобок.

Примеры логических выражений:

```
x1 >= 1 && x1 <= 10
(R > 3.14) and (R < 3.149)
(Value < Oldvalue) OR (Value <> 0)
```

Логический тип

Бейсик	Паскаль	Си++
Базового типа нет. Используется числовой тип INTEGER	boolean	bool

Строчные выражения

Строки в языках программирования всегда заключаются в кавычки. В Си++ и Бейсике для этого используются двойные кавычки, в Паскале — одинарные.

```
"это строка Бейсика или Си++"
'это строка Паскаля'
```

Строка может быть *пустой* — не содержать ни одного символа.

Например:

```
''
""
```

Как правило, строки можно сравнивать друг с другом на эквивалентность (равно и не равно). В некоторых языках программирования допускаются также сравнения типа «больше» или «меньше» — при этом происходит последовательное сравнение значений символов (каждый символ представляется в компьютере конкретным числом).

Кроме того, часто допускается также *операция сцепления строк*, записываемая с помощью символа «+». Например:

```
"123" + "4567" — получится "1234567"
"абв " + "abc " + " эюя" — получится "абв abc  эюя"
```

Тип «строка»

Бейсик	Паскаль	Си++
STRING	string	Базового типа «строка» нет

Указатели

Некоторые языки программирования допускают в явном виде работу с *указателями* — адресами физической памяти. При этом в них имеется специальная *операция получения адреса* конкретной переменной, что позволяет работать с памятью напрямую, примерно так, как это происходит в языках ассемблера. Такая возможность позволяет добиваться высокой эффективности работы программы, но часто приводит к ошибкам, если указатель вдруг получает неверное значение и при его использовании начинает портиться область памяти, предназначенная совсем для других целей.

Сложные данные

Структуры. До сих пор рассматривались базовые типы данных: числа, строки, логические величины — и операции над базовыми данными. Однако для повышения производительности труда программистов и повышения качества их работы необходимо, чтобы язык программирования имел средства, позволяющие описывать данные в виде, максимально приближенном к их реальным аналогам. Например,

чтобы организовать обработку данных по студентам, в программе удобно не просто описать десяток различных переменных, а объединить их в *структуру* (или *запись*) «студент», состоящую из *полей* разного типа «имя», «пол», «год рождения», «группа» и т. д.

Современные языки программирования позволяют применять такие *сложные типы данных*, составляющиеся из базовых и определенных ранее сложных типов. В результате удается организовывать структуры данных произвольной сложности: списки, деревья и т. п. При этом структура объединяет группу разных данных под одним названием.

Получить доступ к отдельным составляющим (полям) этой структуры можно по их именам. В рассматриваемых языках программирования такой доступ осуществляется указанием имени структуры и имени поля через точку. Если подобным способом происходит обращение к полю, которое само является структурой, то выделение нужного поля продолжается приписыванием справа имени вложенного поля через точку.

Синтаксис описания структуры

Бейсик	Паскаль	Си++

TYPE имя-структуры
поле AS тип
...
END TYPE
	record
	поле: тип;
	...
	end;
		struct имя
		{
		тип поле;
		...
		};

Вот примеры описания структур.

Бейсик:

```
TYPE Student
Name AS STRING
Sex AS INTEGER
BirthYear AS INTEGER
END TYPE
```

Паскаль:

```
Record
Name: string;
Sex: boolean;
BirthYear: integer;
end;
```

Си++:

```
struct Student
{
bool Sex;
int BirthYear;
};
```

Доступ к содержимому структуры:

```
Student.BirthYear = 1980;
```

Массивы. Доступ к элементам структуры осуществляется по имени ее составляющих. В одних случаях это значительно повышает наглядность исходных текстов и упрощает процесс программирования, но имеется немало ситуаций, когда надо организовать обработку больших объемов данных одного типа, при этом создавать структуры с сотнями и тысячами полей неразумно. Поэтому в дополнение к структурам в языки программирования введено понятие *массива*, сложного типа данных, доступ к элементам которого происходит по их положению, по номеру или индексу. Например, можно описать массив, состоящий из тысячи элементов численного типа, и затем обратиться к десятому или сотому элементу по его номеру.

При описании массива обычно указывается его *размер* (число элементов) или верхняя и нижняя *границы* — диапазон, в рамках которого можно обращаться к элементам массива.

Синтаксис описания массива

Бейсик	DIM имя (число элементов) AS тип
Паскаль	array[нижняя_граница .. верхняя_граница] of тип;
Си++	тип имя[число-элементов];

В Бейсике нижней границей считается 1, в Си++ — 0, в Паскале она указывается явно.

Вот примеры описания массивов.

Бейсик:

```
DIM IntArray(1000) AS INTEGER
```

Паскаль:

```
array[1..1000] of integer
```

Си++:

```
int IntArray[1000];
```

Доступ к элементу массива осуществляется по его номеру. Этот номер указывается в круглых (Бейсик) или квадратных (Паскаль, Си++) скобках сразу за именем массива (такое действие называется *индексированием*):

```
IntArray( 12 )
IntArray[ i+1 ]
```

Массивы, границы которых явно заданы в команде описания, называются *статическими*. Их размер известен заранее и не меняется на всем протяжении работы программы.

В последних версиях компилируемых языков программирования реализуются так называемые *динамические* массивы, размер которых может меняться во время выполнения программы. В ряде случаев это весьма удобно, так как позволяет экономно расходовать память, захватывая ее по мере необходимости. Недостаток динамических массивов в том, что организовать эффективную работу с ними, исполь-

зуя компиляторы, сложно. Приходится выполнять множество проверок, связанных с расходованием памяти компьютера, что понижает общую эффективность приложения. Динамические массивы в Паскале начали поддерживаться совсем недавно, с активным распространением новых мощных ПК, а в интерпретируемых языках типа Бейсика это было сделано довольно давно.

Во многих языках программирования строки рассматриваются как массивы символов. Их допускается индексировать как обычные массивы.

Правила работы со сложными типами

Отличие базовых типов от сложных в том, что в базовых типах нельзя выделить составные части. При этом поле структуры или элемент массива считаются обычными переменными, и их использование в любых операторах ничем не отличается от использования переменных базовых типов.

В развитых языках программирования допускаются массивы, состоящие из структур, и структуры, состоящие из массивов. При этом возможны достаточно сложные формы записи, например:

```
a[0].Items.Strings[4].value
```

Массив a состоит из структур, в описании которых есть поле Items, являющееся тоже структурой, имеющей поле Strings, которое, в свою очередь, представляет собой массив структур, имеющих поле value.

Описание переменных

Чтобы переменную можно было использовать в программе, ее надо предварительно описать, указав ее тип. Пока переменная не описана, обращаться к ней нельзя (хотя в некоторых языках, например в Бейсике и Фортране, считается, что все переменные, не объявленные явно, имеют числовой тип). После того как переменная описана, к ней можно обращаться, но она обычно исходно имеет *неопределенное значение*, поэтому ее надо предварительно *инициализировать* — присвоить ей начальное значение.

Синтаксис команд описания данных

Бейсик	Паскаль	Си++
DIM имя AS тип	var имя: тип;	тип имя;

Вот примеры описания переменных.

Бейсик:

```
DIM X AS DOUBLE
```

Паскаль:

```
var x: real;
var Str: record
         P1: integer;
         S: string;
         end;
```

Си++:

```
float x;
int a[20];
```

При описании переменных одного типа в Паскале и Си++ их можно указывать через запятую.

Паскаль:

```
var xx, z2: integer;
```

Си++:

```
int xx, yy[10], z2;
```

Новые типы данных

При определении нескольких переменных со сложной структурой удобно описывать каждую переменную, многократно используя одну и ту же запись структуры. Если, например, в нее потребуется внести изменение (добавить новое поле, изменить тип существующего и т. д.), то придется делать это несколько раз, рискуя ошибиться и пропустить одно из описаний, особенно если они сделаны в разных местах программы.

Чтобы избежать этой проблемы и позволить программистам активно применять нужные структуры данных, в современных языках программирования разрешено определять собственные типы данных, которые допускается использовать в командах описания наравне с базовыми типами.

Синтаксис описания нового типа

Бейсик	Паскаль	Си++
Аналогичен описанию структуры, которое уже является описанием нового типа	type имя = описание;	typedef struct имя-структуры { поля-структуры; } имя; Имя структуры надо указывать только из-за требований синтаксиса. Реально оно нигде не применяется

Название нового типа можно использовать во всех последующих командах описания переменных.

Паскаль:

```
type TMyArray = array[0..99] of integer;
type TMyRecord = record
                 Item1: integer;
                 Item2: string;
                 end;
var MyArray: TMyArray;
var R: TMyRecord;
```

Си++:

```
typedef struct name1
{
int i;
float x;
} TNewStruct;
TNewStruct NewStruct;
```

Разделение операторов

Если записать подряд несколько операторов и не указать, где кончается один и начинается другой, то в процессе компиляции возникнет множество проблем с выделением отдельных операторов. Поэтому операторы в Паскале и Си++ отделяются друг от друга точкой с запятой «;» (каждый оператор в этих языках должен заканчиваться таким символом), а в Бейсике — двоеточием «:» или переходом на новую строку.

Блок операторов

Часто в программе возникает необходимость выполнить группу операторов (например, в зависимости от какого-либо условия). Такая группа объединяется в *блок* с помощью специальных скобок начала и конца блока, называемых *логическими скобками*.

В Бейсике явного понятия «блок операторов» нет, в Паскале для этого используются ключевые слова begin и end, а в Си++ — фигурные скобки «{» и «}».

Область действия переменных

Команды описания переменных могут встречаться в разных местах программы. При этом считается, что объявленные в них переменные являются локальными и их *область действия* — текущий блок, в котором они описаны. Как только встречается логическая скобка, закрывающая блок (например, «}»), соответствующая переменная перестает существовать, а выделенная для нее память освобождается.

Некоторые переменные описываются вне блоков и доступны из любого места программы.

Оператор присваивания

Оператор присваивания позволяет изменять текущее значение переменной. Синтаксис его очень простой. В левой части оператора присваивания указывается имя переменной, значение которой изменяется, а справа — выражение, значение которого будет записано в переменную. При этом старое значение, хранившееся в ней, безвозвратно пропадет.

Сам оператор присваивания записывается знаком «=» в Бейсике и Си++ и комбинацией двух знаков «:=» в Паскале (пробел между ними не допускается).

Например:

```
Result = 5
```

В переменную Result запишется число 5. Знак «=» означает именно присваивание, а не сравнение, которое может использоваться только в логических выражениях.

Другой пример:

```
X = X + 1
```

Сначала вычисляется значение выражения X+1, и затем оно заносится в переменную X. Допустима и такая запись:

```
X = X = X
```

Прежде всего выполняется сравнение в правой части (X = X), его значение всегда будет true, и значением переменной X, соответственно, тоже станет true. Для повышения наглядности оператора присваивания в Паскале принята специальная форма его записи:

```
X := X = X;
```

Примеры.

Бейсик:

```
a23 = a22(12) + 1: b1 = b1 - 1
```

Паскаль:

```
a := b*2 + c; d := (e[8] - f)*2.2;
```

Си++:

```
x[5] = y/3.33; y = z[0] - 0.001;
```

Комментарии

При составлении программы очень полезно комментировать различные участки кода, чтобы потом, обратившись к ним, сразу понять, что конкретно выполняется в том или ином месте программы. Забыть смысл того, что было сделано совсем недавно, можно очень быстро — за несколько недель, а в больших проектах и за несколько дней.

Эта проблема становится особенно актуальной, когда группой специалистов разрабатывается объемное приложение, и разобраться в сотнях тысяч строк своего и чужого исходного текста очень сложно.

Языки программирования допускают использование *комментариев* — частей исходных текстов, выделяемых с помощью специальных обозначений и пропускаемых компилятором при анализе текста программы.

Комментарии могут начинаться и заканчиваться особыми символами и охватывать несколько строк кода, а могут записываться только в конце строки — при этом считается, что весь остаток строки является комментарием.

Для обозначения комментариев в одном и том же языке программирования могут использоваться разные символы, поэтому возможно возникновение *вложенных комментариев*. Допустимость такого вложения задается, как правило, в настройках компилятора.

Синтаксис комментария

	Бейсик	Паскаль	Си++
Однострочный комментарий	REM или '	//	//
Многострочный комментарий	нет	{ } или (* *)	/* */

```
X = 5 'комментарий до конца строки
X := 5; // комментарий до конца строки
/*
это комментарий
языка Си++
*/
{
это комментарий
языка Паскаль
(* а это вложенный комментарий *)
}
```

Условный оператор (условные вычисления)

С помощью одного оператора присваивания можно создавать достаточно сложные расчетные программы, однако реализовать абсолютное большинство алгоритмов, просто последовательно выполняя операторы присваивания, невозможно. Постоянно приходится изменять порядок выполнения последовательности вычислений в зависимости от определенных *условий*. Эти условия записываются в виде логических выражений и всегда принимают одно из двух значений — true или false (истинно или ложно). При этом происходит *разветвление* программы — выполнение в дальнейшем может продолжиться с разных операторов.

Синтаксис условного оператора примерно одинаков во всех языках программирования — он представляет собой конструкцию:

если условие истинно

то выполнить оператор-1

иначе выполнить оператор-2

После ключевого слова IF (*если*) следует условие, и если оно истинно, то выполняется оператор или блок операторов, следующих за ключевым словом THEN (*то*); если же оно ложно, то выполняется оператор или блок операторов, следующих за ключевым словом ELSE (*иначе*).

Синтаксис условного оператора

Бейсик	Паскаль	Си++

IF условие THEN
оператор-1
ELSE
оператор-2
END IF
	if условие then
	оператор-1
	else
	оператор-2;
		if(условие)
		оператор-1
		else
		оператор-2;

Примеры.

Бейсик:

```
IF A <> 0 THEN
  A = 0
ELSE
  A = -1
END IF
```

Паскаль:

```
if a <> 0 then a := 0
          else a := -1;
```

Си++:

```
if( a <> 0 ) a = 0
else a = -1;
```

Вторую часть условного оператора, выполняющуюся в случае, если условие ложно, всегда можно опускать.

Бейсик:

```
IF x < 0 THEN
  y = x / 2
  x = 1
END IF
```

Паскаль:

```
if x < 0 then
  begin
  y := x / 2;
  x := 1;
  end
```

Си++:

```
if( x < 0 )
  {
  y = x / 2;
  x = 1;
  };
```

Повторяющиеся вычисления (операторы цикла)

С помощью условных операторов и операторов присваивания теоретически можно реализовать сколь угодно сложный алгоритм. Однако на практике при необходимости организовать обработку тысяч элементов массива (например, присвоить каж-

дому элементу начальное значение) вручную набирать тысячу операторов присваивания крайне тяжело.

Поэтому в языках программирования имеются средства для организации повторных вычислений, называемые *операторами цикла*. Они бывают двух видов: с фиксированным числом повторений и условные операторы цикла.

Каждый оператор цикла состоит из *заголовка цикла*, определяющего число повторений, и *тела цикла* — повторяемого оператора или блока операторов.

Первый вид оператора цикла

При решении задачи примерно в половине случаев заранее известно, сколько раз понадобится выполнить тело цикла. Так бывает, как правило, при обработке массивов, размер которых всегда или известен заранее, или легко определяется.

Заголовок такого оператора состоит из трех частей — *инициализации переменной-счетчика* или *параметра цикла* (присваивания ей начального значения), *определения конечного значения счетчика*, по достижении которого тело цикла надо выполнить в последний раз, и *приращения счетчика*, определяющего, на сколько будет меняться значение счетчика после каждого выполнения тела цикла.

Синтаксис оператора цикла

Бейсик	FOR счетчик = начальное_значение TO конечное_значение STEP приращение тело_цикла группа_операторов NEXT Если приращение не указывать, то считается, что оно равно 1
Паскаль	for счетчик := начальное_значение to конечное_значение do оператор или блок операторов; Приращение всегда равно 1
Си++	for(счетчик = начальное_значение; условие_завершения; счетчик = счетчик + приращение) оператор или блок операторов;

Примеры инициализации тысячи элементов массива a.

Бейсик:

```
FOR I = 1 TO 1000
   A(I) = 0
NEXT
```

Паскаль:

```
for i := 1 to 1000 do
   a[i] := 0;
```

Си++:

```
for( i = 0; i < 1000; i = i + 1 )
   a[i] = 0;
```

В последнем примере счетчик будет принимать значения от 0 до 999, потому что нумерация элементов массива в Си++ начинается с нуля.

Второй вид оператора цикла

Не менее часто встречаются ситуации, когда число повторений заранее неизвестно — надо выполнять цикл, пока не произойдет некоторое событие (пользователь нажмет на кнопку, точность вычислений уложится в заданный порог и т. д.). В таких ситуациях заголовок цикла упрощается. В нем указывается только условие (логическое выражение) — пока его значение равно true, цикл будет выполняться.

Синтаксис оператора цикла

Бейсик	Паскаль	Си++
DO WHILE условие группа операторов LOOP	while условие do оператор или группа операторов;	while(условие) оператор или группа операторов;

Бейсик:

```
DO WHILE A > B
  A = A - 0.01
LOOP
```

Паскаль:

```
while a > b do
   a := a - 0.01;
```

Си++:

```
while( a > b )
   a = a - 0.01;
```

Зацикливание

При использовании условных операторов цикла программиста подстерегает одна опасность. Как показывает практика, достаточно легко сделать ошибку и неверно задать условие окончания цикла, которое всегда будет истинным, — при этом тело цикла станет выполняться бесконечно. Подобная ситуация называется *зацикливанием*.

Например:

```
a = 0; b = 1;
while( a < b )
   a = a - 0.01;
```

Так как исходное значение переменной a меньше, чем значение переменной b, и это значение будет только уменьшаться, то подобный цикл никогда не закончится.

В некоторых случаях программисты специально применяют подобный трюк, чтобы организовать бесконечный цикл, в котором будут приниматься и обрабатываться

внешние сообщения (события). Тогда использование условного оператора цикла может выглядеть так:

```
while true do
  begin
// тело цикла
  end;
```

Контроль над выходом из цикла при наступлении определенного события при этом полностью возлагается на программиста.

В Бейсике есть специальная форма оператора цикла, позволяющая явно описывать такие бесконечные циклы:

```
DO
    ' тело цикла
LOOP
```

Исключения

Управление порядком выполнения программы может происходить не только с помощью условных операторов и операторов цикла, но и при возникновении *исключений* — ситуаций в программе или операционной системе, требующих немедленного реагирования. Например, при выполнении оператора присваивания и вычислении выражения произошло деление на ноль. Программа остановилась, так как не знает, что ей делать дальше, — ведь получено ошибочное значение. Чаще всего выполнение программы просто прекращается по ошибке, но современные системы разработки позволяют программисту явно контролировать возникновение самых разных исключений (они еще называются *исключительными ситуациями*, требующими немедленного вмешательства) и указывать, какие операторы следует выполнять при их возникновении.

Параллельные вычисления

Еще одна область программирования, в которой возможно изменение явно указанного порядка выполнения операторов, — это область параллельных вычислений. С появлением недорогих ПК с несколькими процессорами возникла возможность *распараллеливания* программы — одновременного выполнения ее независимых частей на разных процессорах, что теоретически позволяет получить выигрыш в быстродействии, линейно зависящий от числа процессоров. Однако на практике это очень сложная задача, которая требует правильного выделения независимых модулей кода (так называемых *процессов*), выполнение которых не скажется на результатах работы других процессов. Так как момент окончания работы того или иного процесса заранее неизвестен, то в программе надо предусмотреть действия, связанные с синхронизацией обработки получаемых результатов. Их выполнение может потребоваться в самые неожиданные моменты, поэтому изменение линейной последовательности работы операторов неизбежно.

Ввод и вывод

Чтобы получать от человека информацию для обработки и показывать результаты своей работы, программа должна иметь средства для организации *интерактивного* общения с пользователем (общения в реальном масштабе времени — человек щелкнул мышкой на кнопке и сразу получил ответ) и средства для ввода данных из файлов и сохранения данных в файлах. Интерактивное общение реализуется с помощью *RAD*-систем, позволяющих быстро спроектировать пользовательский интерфейс. Ввод и вывод информации осуществляется в разных языках по-разному. В Паскале и Бейсике есть операторы для такой работы, в Си++ они выделены в специальные библиотеки. Введен также специальный тип данных «файл» (FILE).

Работа с файлами всегда происходит в три этапа.

1. Файл *открывается* в одном из выбранных режимов (он рассматривается как последовательность строк или двоичных чисел, разрешается только считывать из него данные или только записывать и т. д.). Файл может состоять из последовательности одинаковых блоков, каждый из которых будет представлять собой копию структуры данных определенного типа, описанного в программе. Каждый такой блок называется *записью*.
2. Выполняется считывание, обновление или удаление записей в файле.
3. Файл *закрывается*. Если этого не сделать, то он останется открытым и в дальнейшем к нему нельзя будет обратиться из других программ.

Каждый из этих пунктов реализуется в каждом из языков программирования по-своему. Некоторые пункты требуют для своей реализации нескольких операторов.

Вопросы для самоконтроля

1. Какие типы данных считаются базовыми?
2. Приведите примеры арифметических и логических выражений.
3. Напишите формулу для вычисления среднего арифметического и среднего геометрического значений двух переменных.
4. В чем различие структуры и массива?
5. Зачем нужны комментарии?
6. С помощью условных операторов выполните проверку неравенства $x < y < z$.
7. Из каких частей состоит оператор цикла?
8. Назовите достоинства и недостатки параллельных вычислений.
9. Как организуется работа с файлами?

20.4. Структурное программирование

Подпрограммы

В предыдущем разделе рассматривались основные операторы и типы данных, необходимые для составления программ. При этом предполагалось, что текст программы

представляет собой линейную последовательность операторов присваивания, цикла и условных операторов. Таким способом можно решать не очень сложные задачи и составлять программы, содержащие несколько сот строк кода. После этого понятность исходного текста резко падает из-за того, что общая структура алгоритма теряется за конкретными операторами языка, выполняющими слишком детальные, элементарные действия. Возникают многочисленные вложенные условные операторы и операторы циклов, логика становится совсем запутанной, при попытке исправить один ошибочный оператор вносится несколько новых ошибок, связанных с особенностями работы этого оператора, результаты выполнения которого нередко учитываются в самых разных местах программы. Поэтому набрать и отладить длинную линейную последовательность операторов практически невозможно.

При создании средних по размеру приложений (несколько тысяч строк исходного кода) используется *структурное программирование*, идея которого заключается в том, что структура программы должна отражать структуру решаемой задачи, чтобы алгоритм решения был ясно виден из исходного текста. Для этого надо иметь средства для создания программы не только с помощью трех простых операторов, но и с помощью средств, более точно отражающих конкретную структуру алгоритма. С этой целью в программирование введено понятие *подпрограммы* — набора операторов, выполняющих нужное действие и не зависящих от других частей исходного кода. Программа разбивается на множество мелких подпрограмм (занимающих до 50 операторов — критический порог для быстрого понимания цели подпрограммы), каждая из которых выполняет одно из действий, предусмотренных исходным заданием. Комбинируя эти подпрограммы, удается формировать итоговый алгоритм уже не из простых операторов, а из законченных блоков кода, имеющих определенную смысловую нагрузку, причем обращаться к таким блокам можно по названиям. Получается, что подпрограммы — это новые операторы или операции языка, определяемые программистом.

Возможность применения подпрограмм относит язык программирования к классу *процедурных* языков.

Нисходящее проектирование

Наличие подпрограмм позволяет вести проектирование и разработку приложения *сверху вниз* — такой подход называется *нисходящим проектированием*. Сначала выделяется несколько подпрограмм, решающих самые глобальные задачи (например, инициализация данных, главная часть и завершение), потом каждый из этих модулей детализируется на более низком уровне, разбиваясь в свою очередь на небольшое число других подпрограмм, и так происходит до тех пор, пока вся задача не окажется реализованной.

Такой подход удобен тем, что позволяет человеку постоянно мыслить на предметном уровне, не опускаясь до конкретных операторов и переменных. Кроме того, появляется возможность некоторые подпрограммы не реализовывать сразу, а временно откладывать, пока не будут закончены другие части. Например, если имеется необходимость вычисления сложной математической функции, то выделяется

отдельная подпрограмма такого вычисления, но реализуется она временно одним оператором, который просто присваивает заранее выбранное значение (например, 5). Когда все приложение будет написано и отлажено, тогда можно приступить к реализации этой функции.

Немаловажно, что небольшие подпрограммы значительно проще отлаживать, что существенно повышает общую надежность всей программы.

Очень важная характеристика подпрограмм — это возможность их *повторного использования*. С интегрированными системами программирования поставляются большие библиотеки стандартных подпрограмм, которые позволяют значительно повысить производительность труда за счет использования чужой работы по созданию часто применяемых подпрограмм.

Рассмотрим пример, демонстрирующий методику нисходящего проектирования. Имеется массив Ocenki, состоящий из N ($N > 2$) судейских оценок (каждая оценка положительна). В некоторых видах спорта принято отбрасывать самую большую и самую маленькую оценки, чтобы избежать влияния необъективного судейства, а в зачет спортсмену идет среднее арифметическое из оставшихся оценок. Решим эту задачу, постепенно детализируя алгоритм (без привязки к конкретному языку программирования).

1. Процесс решения наиболее просто описывается подпрограммами:

   ```
   Ввести_оценки_в_массив;
   Удалить_самую_большую_оценку;
   Удалить_самую_маленькую_оценку;
   Рассчитать_среднее_арифметическое_оставшихся_оценок;
   Вывести_результаты;
   ```

   Теперь можно приступить к детализации каждой их этих подпрограмм.

2. Удалить_самую_большую_оценку;

   Как удалить самую большую оценку из статического массива? Вместо нее можно просто записать значение 0, а при подсчете среднего арифметического нулевые значения не учитывать.

   ```
   I = Номер_самого_большого_элемента_в_массиве;
   Ocenki[ I ] = 0;
   ```

3. Удалить_самую_маленькую_оценку;

   ```
   I = Номер_самого_маленького_элемента_в_массиве;
   Ocenki( I ) = 0;
   ```

   При реализации подпрограммы Номер_самого_маленького_элемента_в_массиве надо учесть, что искать придется самое маленькое из *положительных* значений (больших нуля).

4. Рассчитать_среднее_арифметическое_оставшихся_оценок;

Здесь потребуется оператор цикла, вычисляющий сумму всех элементов массива Ocenki.

```
SUM = 0
FOR I = 1 TO N
   SUM = SUM + Ocenki( I )
NEXT
SUM = SUM / (N - 2)
```

В последнем операторе происходит вычисление среднего арифметического всех оценок. Сумма элементов массива делится на число элементов, уменьшенное на 2, потому что две оценки, самую большую и самую маленькую, учитывать не надо.

Если бы эта задача решалась последовательно, то уже на этапе удаления оценок могли возникнуть определенные проблемы.

Реализацию подпрограмм Номер_самого_большого_элемента_в_массиве и Номер_самого_маленького_элемента_в_массиве выполните самостоятельно.

Процедуры и функции

Подпрограммы бывают двух видов — *процедуры* и *функции*. Отличаются они тем, что процедура просто выполняет группу операторов, а функция вдобавок вычисляет некоторое значение и передает его обратно в главную программу (*возвращает* значение). Это значение имеет определенный тип (говорят, что функция *имеет* такой-то тип).

В Си++ понятия «процедура» нет — там имеются только функции, а если никакого значения функция не вычисляет, то считается, что она возвращает значение типа «никакое» (void).

Параметры подпрограмм

Чтобы работа подпрограммы имела смысл, ей надо получить данные из внешней программы, которая эту подпрограмму *вызывает*. Данные передаются подпрограмме в виде *параметров* или *аргументов*, которые обычно описываются в ее заголовке так же, как переменные.

Управление последовательностью вызова подпрограмм

Подпрограммы вызываются, как правило, путем простой записи их названия с нужными параметрами. В Бейсике есть оператор CALL для явного указания того, что происходит вызов подпрограммы.

Подпрограммы активизируются *только* в момент их вызова. Операторы, находящиеся внутри подпрограммы, выполняются, только если эта подпрограмма явно вызвана. Пока выполнение подпрограммы полностью не закончится, оператор главной программы, следующий за командой вызова подпрограммы, выполняться не будет.

Подпрограммы могут быть *вложенными* — допускается вызов подпрограммы не только из главной программы, но и из любых других подпрограмм.

В некоторых языках программирования допускается вызов подпрограммы из себя самой. Такой прием называется *рекурсией* и потенциально опасен тем, что может привести к зацикливанию — бесконечному самовызову.

Структура подпрограммы

Подпрограмма состоит из нескольких частей: заголовка с параметрами, тела подпрограммы (операторов, которые будут выполняться при ее вызове) и завершения подпрограммы.

Локальные переменные, объявленные внутри подпрограммы, имеют областью действия только ее тело.

Функции

	Бейсик	**Паскаль**	**Си++**
Заголовок функции	FUNCTION имя (список_параметров) Тип возвращаемого значения определяется специальным символом после имени функции	function имя (список_параметров): тип_функции;	тип_функции имя(список_параметров)

Тело	Последовательность операторов	begin
		последовательность операторов
		end;
			{
			последовательность операторов
			};

Завершение	END FUNCTION	нет	нет

Процедуры

	Бейсик	**Паскаль**	**Си++**
Заголовок процедуры	SUB имя (список_параметров)	procedure имя (список_параметров);	void имя(список_параметров)

Тело	Последовательность операторов	begin
		последовательность операторов
		end;
			{
			последовательность операторов
			};

Завершение	END SUB	нет	нет

Как функция возвращает значение

После того как функция рассчитала нужное значение, ей требуется явно вернуть его в вызывающую программу. Для этого может использоваться специальный оператор (return в Си++) или особая форма оператора присваивания, когда в левой части указывается имя функции, а справа — возвращаемое значение.

Далее приведены примеры функции, вычисляющей значение квадрата аргумента.

Бейсик:

```
FUNCTION SQR% (X AS INTEGER)
SQR% = X*X
END FUNCTION
```

Паскаль:

```
function SQR(X: integer): integer;
begin
SQR := X*X
end;
```

Си++:

```
int SQR(int x)
{
return x*x;
};
```

Формальные и фактические параметры

Во время создания подпрограммы заранее не известно, какие конкретно параметры она может и будет получать. Поэтому в качестве переменных, выступающих в роли ее аргументов в заголовке, могут использоваться произвольные допустимые названия, даже совпадающие с уже имеющимися. Компилятор все равно поймет, что это не одно и то же.

Параметры, которые указываются в заголовке подпрограммы, называются *формальными*. Они нужны только для описания тела подпрограммы. А параметры (конкретные значения), которые указываются в момент вызова подпрограммы, называются *фактическими* параметрами. При выполнении операторов подпрограммы формальные параметры как бы временно заменятся на фактические.

Пример.

```
int a, y;
a = 5;
y = SQR(a);
```

Программа вызывает функцию SQR() с одним фактическим параметром a. Внутри подпрограммы формальный параметр x получает значение переменной a и возводится в квадрат. Результат возвращается обратно в программу и присваивается переменной y.

Событийно-ориентированное программирование

С активным распространением системы *Windows* и появлением визуальных *RAD*-сред широкую популярность приобрел событийный подход к созданию программ — *событийно-ориентированное программирование*.

Идеология системы *Windows* основана на событиях. Щелкнул человек на кнопке, выбрал пункт меню, нажал на клавишу или кнопку мыши — в *Windows* генерируется подходящее *сообщение*, которое отсылается окну соответствующей программы.

Структура программы, созданной с помощью событийного программирования, следующая. Главная часть представляет собой один бесконечный цикл, который опрашивает *Windows*, следя за тем, не появилось ли новое сообщение. При его обнаружении вызывается подпрограмма, ответственная за *обработку* соответствующего события (обрабатываются не все события, их сотни, а только нужные), и подобный цикл опроса продолжается, пока не будет получено сообщение «Завершить работу».

События могут быть *пользовательскими*, возникшими в результате действий пользователя, *системными*, возникающими в операционной системе (например, сообщения от таймера), и *программными*, генерируемыми самой программой (например, обнаружена ошибка и ее надо обработать).

Событийное программирование является развитием идей нисходящего проектирования, когда постепенно определяются и детализируются реакции программы на различные события.

Вопросы для самоконтроля

1. С какой целью применяют подпрограммы?
2. Чем характеризуются процедурные языки программирования?
3. В чем состоит идея нисходящего проектирования?
4. Что общего и в чем отличия процедуры и функции?
5. Определите значение выражения `F(1,2) + F(10,0.1)`, если функция `F(a,b)` рассчитывается как `a*a + b*b`.
6. В чем различие между событийным и структурным программированием?
7. Как организуется обработка программных событий?

20.5. Объектно-ориентированное программирование

Понятие объекта

Развитие идей структурного и событийного программирования существенно подняло производительность труда программистов и позволило в разумные сроки (несколько месяцев) создавать приложения объемом в сотни тысяч строк. Однако такой объем уже приблизился к пределу возможностей человека, и потребовались новые технологии разработки программ.

В начале 80-х годов в программировании возникло новое направление, основанное на понятии *объекта*. До того времени основные ограничения на возможность создания больших систем накладывала разобщенность в программе данных и методов их обработки.

Реальные объекты окружающего мира обладают тремя базовыми характеристиками: они имеют набор свойств, способны разными методами изменять эти свойства и реагировать на события, возникающие как в окружающем мире, так и внутри самого

объекта. Именно в таком виде в языках программирования и реализовано понятие *объекта* как совокупности *свойств* (структур данных, характерных для этого объекта), *методов* их обработки (подпрограмм изменения свойств) и *событий*, на которые данный объект может реагировать и которые приводят, как правило, к изменению свойств объекта.

Появление возможности создания объектов в программах качественно повлияло на производительность труда программистов. Максимальный объем приложений, которые стали доступны для создания группой программистов из 10 человек, за несколько лет увеличился до миллионов строк кода, при этом одновременно удалось добиться высокой надежности программ и, что немаловажно, повторно использовать ранее созданные объекты в других задачах.

Класс

Объекты могут иметь идентичную структуру и отличаться только значениями свойств. В таких случаях в программе создается новый тип, основанный на единой структуре объекта (по аналогии с тем, как создаются новые типы для структур данных). Он называется *классом*, а каждый конкретный объект, имеющий структуру этого класса, называется *экземпляром класса*.

Описание нового класса

Описание нового класса похоже на описание новой структуры данных, только к полям (свойствам) добавляются методы — подпрограммы.

В Си++ и Паскале для описания класса используется ключевое слово class.

Паскаль:

```
class TMyClass
Item1: integer;
Item2: string;
function GetSum(n: integer): integer;
procedure Initialize;
end;
```

Си++:

```
class TMyClass
{
int Item1;
int Item2;
int GetSum(int n);
void Initialize();
};
```

При определении подпрограмм, принадлежащих конкретному классу, его методов, в заголовке подпрограммы перед ее названием явно указывается, к какому классу

она принадлежит. Название класса от названия метода отделяют специальные символы (точка в Паскале или два двоеточия в Си++).

Паскаль:

```
procedure TMyClass.Initialize;
begin
Item1 := 1;
Item2 := «»;
end;
```

Си++:

```
void TMyClass::Initialize()
{
Item1 = 1;
Item2 = 0;
}
```

Класс — это тип данных, такой же, как любой другой базовый или сложный тип. На его основе можно описывать конкретные объекты (экземпляры классов).

Паскаль:

```
var C1, C2: TMyClass;
```

Си++:

```
TMyClass C1, C2;
```

Доступ к свойствам объектов и к их методам осуществляется так же, как к полям записей, через точку:

```
C1.Item1 := 5;
C2.Initialize;
x := C1.GetSum(21);
```

Объектно-ориентированное программирование базируется на трех ключевых концепциях — инкапсуляции, наследовании и полиморфизме. Объединение данных с методами в одном типе (классе) называется *инкапсуляцией*. Помимо объединения, инкапсуляция позволяет ограничивать доступ к данным объектов и реализации методов классов. В результате у программистов появляется возможность использования готовых классов в своих приложениях на основе только описаний этих классов.

Наследование

Важнейшая характеристика класса — возможность создания на его основе новых классов с *наследованием* всех его свойств и методов и добавлением собственных. Класс, не имеющий предшественника, называется *базовым*.

Например, класс «животное» имеет свойства «название», «размер», методы «идти» и «размножаться». Созданный на его основе класс «кошка» наследует все эти свойства и методы, к которым дополнительно добавляется свойство «окраска» и метод «пить».

Наследование позволяет создавать новые классы, повторно используя уже готовый исходный код и не тратя времени на его переписывание.

Полиморфизм

В большинстве случаев методы базового класса у классов-наследников приходится переопределять — объект класса «кошка» выполняет метод «идти» совсем не так, как объект класса «амеба». Все переопределяемые методы по написанию (названию) будут совпадать с методами базового объекта, однако компилятор по типу объекта (его классу) распознает, какой конкретно метод надо использовать, и не вызовет для объекта класса «кошка» метод «идти» класса «животное». Такое свойство объектов переопределять методы наследуемого класса и корректно их использовать называется *полиморфизмом*.

Визуальное программирование

Технологии объектного, событийного и структурного программирования сегодня объединены в *RAD*-системах, которые содержат множество готовых классов, представленных в виде визуальных *компонентов*, которые добавляются в программу одним щелчком мыши. Программисту надо только спроектировать внешний вид окон своего приложения и определить обработку основных событий — какие операторы будут выполняться при нажатии на кнопки, при выборе пунктов меню или щелчках мышкой. Весь вспомогательный исходный код среда сгенерирует сама, позволяя программисту полностью сосредоточиться только на реализации алгоритма.

Вопросы для самоконтроля

1. Для чего в языки программирования было введено понятие класса?
2. В чем различие между классом и объектом?
3. Поясните понятие инкапсуляции на бытовых примерах.
4. Для чего применяется механизм наследования?
5. Как полиморфизм модифицирует принцип наследования?
6. Опишите использование принципов объектно-ориентированного программирования в средах быстрого проектирования.

20.6. Проектирование программ

Программирование как вид деятельности

Появление первых компьютеров породило программирование как *науку*. Разрабатывались первые математические теории обработки информации, средства доказательства правильности программ, оптимизации кода, создания эффективных компиляторов, формального тестирования и т. д. Затем, с появлением универсальных языков программирования третьего поколения, эти аспекты стали менее актуальными — исследования шли и идут в основном в области автоматической генерации исходных текстов и повышения эффективности компиляторов. Программирование превратилось в *искусство* — миллионы людей, не имевших специального образования, получили возможности применять компьютеры для решения собственных

прикладных задач, что потребовало от них мастерства создавать правильно работающие программы. Искусством программирование остается и сегодня для профессиональных разработчиков и любителей, создающих программы в одиночку или в небольших компаниях, где все решает индивидуальное мастерство.

Вместе с тем, при росте спроса со стороны государственных и частных организаций на все более и более сложные системы автоматизации предприятий, надежные операционные среды, комплексы глобального телекоммуникационного управления, возникла необходимость в постановке процесса разработки программного обеспечения (ПО) на поток, превращения программирования в *ремесло*. Было разработано несколько методологий и стандартов, позволивших эффективно организовывать труд сотен программистов средней квалификации, точно укладываться в отпущенные сроки и средства и не зависеть от настроения нескольких талантливых ведущих специалистов. Отрицательная сторона подобных методологий — отсутствие творческого элемента в работе и своеобразная конвейерная «потогонная» система промышленного производства программ, которая, будучи внедренной в организации, в условиях жесточайшего дефицита программистов во всем мире может только отпугнуть сотрудников.

Потенциальные возможности человека

Объем проекта, строк исходного кода	Тип программы	Время создания	Вероятность успешного завершения	Число программистов
100	Утилиты для временных нужд	1 день	100%	1
1000	Небольшие приложения и дополнения, вносимые в готовые системы	до 1 месяца	100%	1
10 000	Типичная средняя программа, разрабатываемая на заказ	до 6 месяцев	85%	1 (предел возможностей среднего программиста)
100 000	Большинство современных коммерческих автономных и небольших клиент-серверных приложений	1 год	85% для групп, 35% для одиночки	10
1 млн	Крупные системы автоматизации	1,5–5 лет	50% для группы, 0% для одиночки	100
10 млн	Операционные системы (Microsoft Windows, IBM VMS), большие военные комплексы. Предел сегодняшних возможностей. Стоимость подобной разработки может равняться стоимости большого стадиона или крупного корабля	5–8 лет	35%	до тысячи

Экономические аспекты программирования

Когда на свет появились первые компьютеры, одна минута их работы стоила очень дорого, а задачи решались достаточно простые, поэтому в расходах на подготовку программ труд разработчиков составлял небольшую часть. С появлением ПК и ростом спроса на большие программные системы практически всю расходную часть проекта стала составлять зарплата программистов. Как видно из таблицы, большой процент таких проектов заканчивается неудачно, а расходы на них очень велики, поэтому проблемы создания качественного программного обеспечения точно в срок и в рамках бюджета сегодня самые важные и над созданием эффективных методологий производства ПО трудятся специалисты во всех развитых странах.

Этапы разработки программ

Программы небольшого и среднего размера (несколько тысяч строк) создаются, как правило, в два этапа. Сначала необходимо точно установить, что надо сделать, продумать соответствующий алгоритм, определить структуры данных, объекты и взаимодействие между ними (это этап *системного анализа*), а затем выразить этот алгоритм в виде, понятном машине (этап *кодирования*). Если же разрабатывается крупный проект объемом от десятков тысяч до миллионов строк кода, тогда приходится применять специальные методологии проектирования, охватывающие *период разработки ПО*.

Период разработки ПО

Рассмотрим классический период разработки ПО.

1. Формируются и анализируются *требования* к проекту. Этот этап самый важный, так как неправильное формулирование требований приводит к выполнению ненужной работы, а недооценка сложности вызывает перерасход средств и времени. Сегодня около 60% крупных проектов завершаются неудачей именно из-за ошибок на стадии подготовки требований.

   На основе требований по различным методикам определяется примерный объем проекта и его трудоемкость, рассчитываются будущие трудозатраты и определяется его *стоимость*. Так как требования к проекту во время работы над ним могут уточняться и меняться, а выполнение требований надо отслеживать, применяются специальные программы для управления требованиями.

   Часто заказчик не в состоянии точно выразить, чего он хочет, и задача специалистов по системному анализу — помочь ему выразить свои требования в виде, пригодном для формализации. После согласования требований подписывается *контракт на разработку ПО*. В дальнейшем любые отклонения от сформулированных требований к продукту (как со стороны заказчика, так и со стороны исполнителя) рассматриваются как нарушение контракта.

   Каждый этап требует от заказчика вложения собственных трудозатрат и привлечения высокопрофессиональных специалистов, поэтому он обязательно должен оплачиваться — бесплатно выполнять сложную работу никто не будет. Однако, хотя первый этап самый важный, заказчик редко понимает эту важ-

ность и не готов платить достаточно большие суммы. Примерный объем работ на этом этапе — 5% от объема всего проекта.

2. Начинается предпроектное обследование объекта автоматизации. С помощью *CASE*-средств составляется формальная модель его работы, модель базы данных, объектов и потоков информации. На этом этапе привлекаются специалисты заказчика и эксперты, хорошо знакомые с предметной областью, для которой составляется задача.

   Примерный объем работ второго этапа — 10% от общего.

3. На основе формальной модели составляется подробное *техническое задание* для программистов, *спецификации* отдельных модулей, таблицы баз данных, другая сопроводительная документация. Готовится подробный *календарный план работ*, где указываются все сроки, конкретные исполнители и выполняемые объемы работ (понедельно, помесячно). Для составления планов работ имеется немало хороших программ, ориентированных как на небольшие группы программистов, так и на коллективы из сотен сотрудников, выполняющих тысячи различных работ в рамках общего плана.

   Примерный объем работ третьего этапа — 10% от общего.

4. Выбирается методология разработки ПО и начинается разработка (кодирование). Крупные компании имеют собственные методологии, ориентированные на конкретные задачи (как правило, это задачи автоматизации предприятий), однако все методологии имеют общие черты и более-менее серьезно различающихся известно около двух десятков.

   Мы коснулись, в частности, методологий структурного (нисходящего) и объектного проектирования. Хотя объектный подход, безусловно, один из самых передовых, он подразумевает высокую квалификацию всех исполнителей без исключения и поэтому распространен не очень широко.

   Достаточно популярна методология *итерационного проектирования*, ориентированная на использование *RAD*-средств и систем автоматической генерации исходных текстов на основе созданной формальной модели. Такой подход хорош тем, что позволяет быстро создать первый *работающий* прототип программы, когда еще требования к ней окончательно не определены, а в дальнейшем, на следующих итерациях (их обычно требуется от двух до пяти), постепенно детализировать и реализовывать конкретные возможности, пропущенные по каким-то причинам на предыдущей итерации. Эта методология немного отличается от нисходящего проектирования тем, что применяется, когда окончательные требования неизвестны и могут меняться, а основные работающие функции нужны заказчику как можно быстрее (заказчик чаще хочет получить приложение, законченное на 80%, сегодня, чем законченное на 100% завтра). При нисходящем проектировании основная структура задачи должна быть определена заранее.

   Принятие решения о выборе подходящей методологии — очень ответственный процесс. Человек, принимающий такое решение, должен обладать богатым опытом и знаниями в области создания ПО. Многое зависит от инфраструктуры

заказчика — какие у него компьютеры, операционные системы, каковы их ресурсы. В соответствии с этим выбираются и средства разработки. Иногда бывает, что лучше всего подходит, например, итерационная методология, но для операционной системы заказчика нет хорошей *RAD*-среды и приходится остановиться на менее эффективной методологии.

В процессе разработки необходимо:

- непрерывно поддерживать *обратную связь* с заказчиком, чтобы следить за правильностью реализации требований;
- непрерывно контролировать ход работ в соответствии с планом и при отклонениях от него принимать экстренные меры.

Примерный объем этих работ — 10% от общего объема проекта.

5. Когда программа закончена (готова работоспособная *альфа-версия*), она поступает к *тестерам* компании-исполнителя, которые начинают проверять ее на наличие ошибок и сообщать о найденных ошибках программистам. Анализируется, в частности, устойчивость работы программы при вводе недопустимых или критических значений, при отсутствии информации, при неверных действиях, при сбоях аппаратуры, в стрессовых режимах и т. п. Когда число ошибок, выявляемых за определенный срок (неделя, месяц), снижается ниже экспериментально подобранного уровня (на основе аналогичных проектов), начинается *бета-тестирование* программы у заказчика. К такому тестированию привлекается максимально возможное число сотрудников, и программа уже начинает частично функционировать в рабочем режиме.

   Примерный объем этих работ — 10% от общего объема проекта.

6. После того как заказчик удовлетворен качеством продукта, начинается его *внедрение* — подготовка к окончательному запуску в эксплуатацию. Если приложение многопользовательское, нередко требуется сформировать и настроить локальную сеть, установить серверы, инсталлировать вспомогательные программы. Очень много проблем возникает при переходе со старых программ (например, бухгалтерского учета, расчета зарплаты, работы с персоналом), которые создавались разными сотрудниками и покупались в разных фирмах (так называемая *лоскутная автоматизация*), к новой интегрированной системе. Необходимо выверить, ввести или перенести множество жизненно важной информации: всю бухгалтерскую и финансовую отчетность, данные о хранимом оборудовании и множество других сведений. Этот этап самый трудоемкий и «нудный» и занимает порой до 90% времени всего проекта. Для систем автоматизации больших предприятий он растягивается на годы.

7. После того как новая система готова к работе, сотрудников организации заказчика нужно *обучить* работе с этой системой, потому что книг о ней не написано, да и содержится в такой системе, внедренной на конкретном предприятии, множество нюансов, связанных со спецификой работы.

   Примерный объем трудозатрат на обучение — 5% от общего объема проекта.

8. После того как заказчик подписывает *акт приемки*, проект считается завершенным, но связь с исполнителем не теряется. Особенно в первое время у пользователей системы постоянно будет возникать множество вопросов по работе с ней. Неизбежно и возникновение ошибок, которые требуется устранять. Кроме того, исполнитель может выпускать новые версии системы, и старая система потребует *обновления*. Сотрудничество с заказчиком по обслуживанию системы называется *сопровождением*. Оно бесплатно на определенный гарантийный срок (например, год).

   Реально объем непосредственного программирования и отладки (тестирования) в цикле разработки невелик. Он составляет 10-20% от общего объема работ.

Контроль качества

Чем крупнее проект, тем больше в нем ошибок. При этом слишком затягивать этап тестирования нельзя — нарушаются сроки, растет недовольство потребителей, и на рынок выпускается «сырая» система с множеством ошибок, которые устраняются уже в процессе эксплуатации выпуском многочисленных «заплаток».

Современные технологии создания надежного ПО предусматривают *непрерывный сквозной контроль качества* разрабатываемого продукта на всех этапах жизненного цикла — от анализа требований до внедрения и сопровождения, а не только на этапе тестирования. Качество каждой работы в плане формализуется числовой величиной с помощью специальных методик, но его можно контролировать, только оптимальным образом организовав работу большой группы аналитиков и программистов. Для этого надо иметь возможность *отслеживать* вносимые в проект *изменения* — изменения требований, формальных моделей, сопроводительной документации, версий исходных текстов, хода выполнения календарного плана по разработке, тестированию, внедрению, сопровождению, а также *контролировать и управлять* всеми этапами периода создания программы — *процессом* разработки ПО. Для этого предназначены *системы конфигурационного управления* — сложные и дорогие (десятки и сотни тысяч долларов) продукты, однако без них крупный проект, скорее всего, обречен на неудачу.

Системы не очень сложного конфигурационного управления, охватывающие контроль версий исходных текстов и ряд других аспектов работы группы программистов, встроены, в частности, в такие системы, как *Delphi* 7 и *Visual Studio .NET*.

Стандарты качества ПО

Компания может организовать у себя очень эффективный процесс разработки ПО, однако заказчик вполне обоснованно может ей не поверить. Существует международная система сертификации компаний по стандарту качества *ISO* 9000, которая гарантирует, что данная компания выполняет программные проекты в срок и с высоким качеством. Процесс сертификации сложен и требует подготовительной работы в течение нескольких лет.

В университете Карнеги-Меллона в США несколько лет назад была разработана специальная методология *CMM* (*Capability Maturity Model for Software*), позволя-

ющая сертифицировать компании по одному из 5 уровней «зрелости» процесса разработки ПО. Согласно результатам 20-летних исследований Министерства обороны США оказалось, что главная причина слишком частых неудач при разработке крупных информационных проектов заключается прежде всего в неумении менеджеров управлять процессом создания качественного ПО.

В отличие от стандарта *ISO* 9000, который просто подтверждает качественную работу компании на основании достаточно общих критериев, методология *CMM* ориентирована именно на качество управления процессом разработки и имеет множество конкретных рекомендаций и указаний по способам организации всех этапов создания ПО. Сегодня в США невозможно получить крупный государственный или военный заказ на создание программного продукта стоимостью более 2 млн. долларов, если компания не сертифицирована как минимум по третьему уровню *CMM*. Сегодня на смену этой модели приходит новая интеграционная концепция *CMMI*, дополняющая процессы разработки не менее важными вопросами организации деятельности персонала, общения с подрядчиками, выбора программного обеспечения и т. д.

Повышение индивидуального мастерства

На основе методологии *CMM* была создана методология *PSP* (*Personal Software Process*), ориентированная на индивидуальных разработчиков. Она позволяет в несколько раз повысить качество создания программ, значительно поднять собственную производительность и научиться предсказывать сроки выполнения работ.

Методология *PSP* состоит из 7 этапов самосовершенствования, и, чтобы хорошо ее освоить, надо закончить специальные курсы. Однако даже простое знакомство с ее идеологией позволит любому программисту значительно улучшить свою работу.

Перед началом проекта составляется подробный календарный план работ и производится попытка оценить его объем в строках кода и рабочих днях. Весь процесс работы детально хронометрируется, а найденные ошибки подробно описываются — накапливается статистика. По окончании проекта весь процесс тщательно анализируется и делаются выводы о том, что можно улучшить в своей работе, каких ошибок надо стараться избегать, какова реальная производительность труда и т. п. Сегодня лучшая характеристика программиста — это не просто знание Си++ или *Delphi*, а способность планировать свой труд, разрабатывать программу точно в срок и без ошибок.

Гибкие методики

В последние годы наряду с ростом интереса к «тяжелым» сертификационным методологиям значительно увеличилась роль и популярность так называемых гибких, проворных (*agile*) методик. Они не требуют значительных усилий по реорганизации компании-разработчика и могут быть внедрены не за годы, а за недели. Наиболее известные среди них — экстремальное программирование, *MDA* (архитектура системы, управляемая моделью), *Scrum* и другие.

Гибкие методики не навязывают исполнителю множество формализованных действий по разработке ПО. Они предоставляют участникам проекта большую свободу

(но и требуют от них большей ответственности) и акцентируют их внимание на важнейших задачах соблюдения требований заказчика, сроков и качества работы.

Методы маркетинга программного обеспечения

Коммерческое ПО. При создании программного продукта *издатель*, выполнив анализ рынка, заказывает у *исполнителя* разработку такого ПО, которое должно пользоваться на рынке спросом, и выделяет на его создание деньги. По окончании работ издатель получает все *имущественные права* на созданный продукт (право на тиражирование, продажу под собственной торговой маркой, право на получение дохода от программы любым способом). При этом может быть оговорено получение исполнителем некоторого процента (*роялти*) с каждой проданной копии (как правило, для программ, издающихся сотнями тысяч или миллионами копий, роялти составляет 1-3%) — тогда он получает меньшую сумму на разработку или вообще создает программу за свой счет. Если же отчисления не предусмотрены, то все расходы по подготовке программы издатель берет на себя. Он также вкладывает средства в упаковку, рекламную кампанию, организацию сетей сбыта и т. д. Издатель обеспечивает расходы, связанные с сопровождением продукта и технической поддержкой пользователей.

За исполнителем навечно остаются *авторские права* на программу — право указывать свое имя или логотип своей фирмы на начальной заставке, в документации, на упаковочной коробке.

Крупные компании имеют и подразделения разработки ПО, и отделы, занимающиеся его распространением, что помогает эффективно организовать весь процесс от производства программ до доставки их потребителю.

Условно-бесплатное ПО (shareware). В связи с активным развитием Интернета огромное число индивидуальных разработчиков получили возможность распространения своих программ по всему миру. Не имея средств на рекламные кампании, они предоставляют возможность получения *ознакомительных версий* их программ (демонстрационных или имеющих искусственные ограничения) через Интернет. Если человеку эта программа нравится, он оплачивает небольшую сумму и получает полную работоспособную версию.

В Интернете есть немало узлов, которые предлагают бесплатные услуги по размещению таких программ. Отечественным *shareware*-разработчикам можно порекомендовать сайт www.swrus.com, на котором можно найти множество материалов и форумов по всем вопросам организации *shareware*.

Бесплатное ПО (freeware, public domain). Такие программы не имеют никаких ограничений, однако автор может *попросить* заплатить ему некоторую сумму, не настаивая, впрочем, на этом (это метод *freeware*). Некоторые программы авторы называют «общественным достоянием» (*public domain*), ничего взамен не требуют и нередко распространяют такое ПО в исходных текстах.

Как правило, стимулом к созданию бесплатных программ служит стремление повысить собственную квалификацию, установить контакты с коллегами, а в случае удачно созданной программы получить известность.

Вопросы для самоконтроля

1. В чем трудности разработки крупных программных проектов?
2. Опишите организацию работы над сложной программной системой.
3. Какой этап разработки проекта является наиболее ответственным?
4. Какова роль собственно программирования в ходе работы над проектом?
5. Опишите известные вам методы контроля качества программного обеспечения.
6. Каковы основные методы распространения программного обеспечения?

20.7. Пример на Бейсике. Разведение кроликов

В данном и последующих разделах рассматриваются три примера, реализованные с помощью разных систем программирования: *QBasic* корпорации *Microsoft* (интерпретирующая версия Бейсика для операционной системы *MS-DOS*), *Borland Delphi 7* (система визуального программирования на Паскале) и *Microsoft Visual Studio .NET* (система программирования на Си++). Эти примеры включают в себя описание основных приемов работы с данными системами.

Постановка задачи

Итальянский математик Леонардо Фибоначчи придумал оригинальную числовую последовательность, названную в его честь, которая описывает рост численности поколений кроликов. Считается, что каждый год каждая пара животных приносит приплод — новую пару (самца и самку), которые в свою очередь начинают давать приплод через два года (смертность не учитывается). То есть каждый следующий член последовательности равен сумме двух предыдущих, а классическая последовательность Фибоначчи выглядит так:

1, 1, 2, 3, 5, 8, 13, 21, ...

Надо определить, через сколько лет будет достигнута популяция в N особей.

Запуск QBasic

Интерпретатор *QBasic* входит в стандартную поставку *MS-DOS* и расположен обычно в каталоге \DOS. Программа-интерпретатор называется qbasic.exe. После ее запуска на экране появится приветствие, которое пропускается нажатием на клавишу ENTER, после чего *QBasic* вызывает встроенную справочную систему на английском языке. Она закрывается нажатием клавиши ESC.

Рабочая область экрана (рис. 20.1) поделена на две части. В нижней части, в окне Immediate (Немедленное выполнение) можно вводить операторы Бейсика и тут же их выполнять.

Вывод на экран

Каждый язык программирования имеет оригинальные средства вывода информации, сильно зависящие от операционной системы. В Бейсике реализован оператор PRINT, который выводит значение следующего за ним выражения на экран, в новую строку.

Для перехода в окно Immediate (Немедленное выполнение) надо нажать клавишу F6. Чтобы сразу получить ответную реакцию от *QBasic*, достаточно набрать оператор

```
PRINT 2+2
```

и нажать клавишу ENTER, чтобы этот оператор выполнился.

На экране вывода появится число 4 (результат вычисления выражения 2+2), а в нижней строке — сообщение Press any key to continue (Нажмите любую клавишу для продолжения). Чтобы вернуться в *QBasic*, надо это сделать.

В операторе вывода можно указывать несколько значений через запятую, тогда они будут выведены в одной строке. Ранее введенный оператор можно изменить, добавив к нему текстовую подсказку:

```
PRINT "Сумма = "; 2+3
```

Если теперь нажать клавишу ENTER, то на экране вывода в новой строке (под ранее напечатанной четверкой) появится фраза

```
Сумма = 5
```

Редактор программы

Таким образом можно познакомиться с работой разных операторов Бейсика, однако выполнять их удастся только поодиночке. Чтобы выполнить группу операторов, их надо объединить в программу.

Набор и редактирование исходного текста программы осуществляется в верхнем окне интерпретатора. Для перехода в него используется клавиша F6.

Решать данную задачу удобнее всего с помощью нисходящего метода. В программе будет бесконечный главный цикл, в котором осуществляется ввод очередного значения количества особей, происходит расчет числа лет, необходимых для размножения, полученный результат печатается, и цикл повторяется снова. Если человек вводит ноль, это будет означать, что программу надо завершить.

Ввод информации от пользователя

В Бейсике ввести в переменную значение с экрана можно с помощью оператора INPUT. Сначала указывается необязательная текстовая подсказка, а потом — имя переменной. Например, оператор

```
INPUT "Введите число: ", x
```

при выполнении напечатает в новой строке подсказку

```
Введите число:
```

и будет ожидать, когда пользователь введет число и нажмет клавишу ENTER. В результате в переменную x запишется новое, введенное с клавиатуры значение.

Главная часть программы

Ввод и редактирование текста программы осуществляется во встроенном редакторе *QBasic*, правила работы с которым аналогичны правилам работы с большинством известных текстовых редакторов.

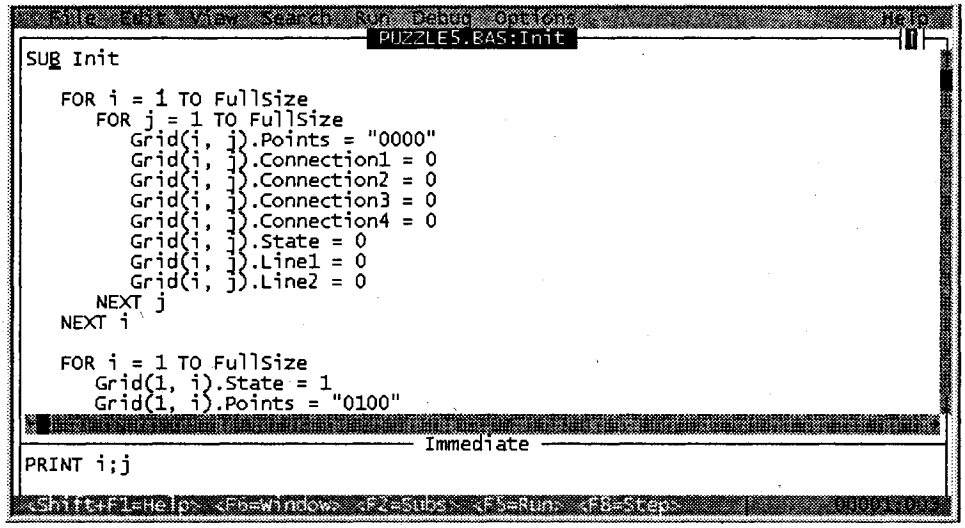


Рис. 20.1. Окно программы QBasic

Главная часть программы набирается в этом редакторе и должна выглядеть так (комментарии вводить не обязательно):

```
' описание переменной N — числа особей
DIM N AS INTEGER
' начало бесконечного цикла
DO
' ввод числа особей в переменную N
  INPUT "Введите количество особей: ", N
' если введен 0, то
   IF N = 0 THEN
' закончить программу
      END
   END IF
' напечатать результат:
   PRINT "Требуемое число лет: ", Years%(N)
' продолжить цикл с начала
LOOP
```

В тексте используется оператор **END**, который предназначен для немедленного завершения работы программы. Операторы, вложенные в цикл и в условные операторы, выделяются отступами, чтобы структура текста была более понятной и наглядной.

Основная, глобальная часть алгоритма реализована. Осталось «спуститься вниз» и запрограммировать функцию Years%(), которая в качестве аргумента получает количество особей и возвращает число лет, требуемое для их разведения.

Типы данных в Бейсике

В конце названия функции Years% указан символ %. Таким образом в Бейсике описывается тип возвращаемого функцией значения. Допустимые символы приведены в таблице.

Тип переменной	Символ в конце имени переменной
INTEGER	%
STRING	$
DOUBLE	#

Добавление новой функции

В *QBasic* имеется удобная возможность добавить в программу новую функцию, избежав при этом дополнительного ручного кодирования. Это делает команда Edit ▸ New Function (Правка ▸ Создать функцию). В появившемся диалоговом окне надо ввести название функции Years% и нажать клавишу ENTER. Основной текст программы временно пропадет, и появится автоматически сгенерированное описание новой функции:

```
FUNCTION Years%
END FUNCTION
```

Для того чтобы вернуться обратно к главному тексту, а из него — к любой введенной подпрограмме, необходимо использовать клавишу F2. При ее нажатии на экран выводится список всех созданных подпрограмм, а в первой строке — имя главного модуля.

Функции Years% надо указать список аргументов. В данном случае он будет состоять из одного параметра:

```
FUNCTION Years%(X AS INTEGER)
```

Расчет популяции

Так как для определения нового члена последовательности Фибоначчи требуется знать значения двух предыдущих членов, прежде всего надо описать три локальные переменные F1, F2 и F3, хранящие три очередных значения последовательности. Исходно первые три значения 1, 1 и 2 запишутся в переменные F1, F2 и F3 явно, а в дальнейшем новые значения будут вычисляться программно.

Сам расчет представляет собой условный цикл, который выполняется до тех пор, пока очередное значение не превысит заданное количество особей. Число таких циклов — число лет — будет подсчитываться в локальной переменной-счетчике YearsNum, первоначально имеющей значение 3.

```
FUNCTION Years%(X AS INTEGER)
' описание переменных
```

```
DIM F1 AS INTEGER, F2 AS INTEGER, F3 AS INTEGER
DIM YearsNum AS INTEGER
' задание начальных значений
F1 = 1: F2 = 1: F3 = 2: YearsNum = 3
' цикл, пока число кроликов меньше заданного
DO WHILE F3 < X
' определяем новый член последовательности
  F1 = F2: F2 = F3
  F3 = F1 + F2
' увеличиваем число лет на 1:
  YearsNum = YearsNum + 1
' повторяем цикл
LOOP
' в качестве возвращаемого значения
' используется значение переменной YearsNum
Years% = YearsNum
END FUNCTION
```

Сохранение текста программы в файле

После того как текст программы набран, его желательно сохранить в файле, чтобы потом снова обращаться к нему, улучшать, изменять или просто повторно запускать готовую программу.

Сохранение текста программы в файле осуществляется командой File ▸ Save (Файл ▸ Сохранить), в результате чего на экране показывается диалоговое окно выбора каталога и имени файла. В качестве такого имени можно указать kroliki, выбрать нужный каталог и нажать клавишу ENTER. По умолчанию к названию kroliki припишется расширение .BAS. В дальнейшем эту программу можно снова загрузить в *QBasic* командой File ▸ Open (Файл ▸ Открыть).

Запуск программы

Для запуска программы надо перейти к ее главной части (с помощью клавиши F2) — при этом в самом ее начале автоматически добавится строка с объявлением только что определенной функции:

```
DECLARE FUNCTION Years% (X AS INTEGER)
```

Теперь надо нажать клавишу F5 (Запуск). Программа начинает работать.

Возможный вариант диалога:

```
Введите количество особей: 10
Требуемое число лет: 7
Введите количество особей: 100
```

```
Требуемое число лет: 12
Введите количество особей: 1000
Требуемое число лет: 17
Введите количество особей: 10000
Требуемое число лет: 21
Введите количество особей: 0
```

Первую сотню кроликов надо разводить довольно долго, зато потом их приплод будет увеличиваться стремительными темпами.

20.8. Пример на Паскале. Раскрашивание круга

Постановка задачи

В рабочем окне программы должны находиться: изображение круга, поле ввода с подписью, кнопки Закрасить и Закрыть. В поле ввода шестью символами (шестнадцатеричными цифрами от 0 до F) задается новый цвет круга. Первые два символа определяют интенсивность синего цвета, третий и четвертый — интенсивность зеленого, пятый и шестой — интенсивность красного. Чистый синий опишется строкой ff0000, чистый зеленый — строкой 00ff00, чистый красный — строкой 0000ff, черный — строкой 000000, белый — строкой ffffff и т. д.

В самой программе надо дополнительно проверить, правильно ли введена эта строка. При нажатии на кнопку Закрасить цвет круга должен измениться.

Знакомство с Delphi

Для работы со средой *Delphi* необходимо, чтобы она была предварительно установлена на компьютере.

После ее запуска (например, Пуск ▸ Программы ▸ Borland Delphi 7 ▸ Delphi 7) на экране появятся следующие окна (рис. 20.2).

Главное окно Delphi 7. Здесь расположено основное меню, командные кнопки, а в правой части — *палитра компонентов*. Она состоит из набора панелей, на которых компоненты сгруппированы по решаемым задачам: панель Standard (Стандартная) — стандартные элементы управления, панель Win32 — элементы управления для версии *Windows 9x*, панель Internet (Интернет) — компоненты для организации работы в Интернете и т. д.

Визуальный проектировщик. С его помощью проектируется будущее окно программы. Оно представлено в виде формы, у которой можно менять свойства и размеры. На ней будут располагаться нужные элементы управления.

Редактор исходных тестов. Предназначен для набора и редактирования текстов программы. Ключевые слова и различные идентификаторы выделяются в этом редакторе особыми цветами и разным шрифтом. Принцип его работы аналогичен принципам работы большинства редакторов *Windows*.

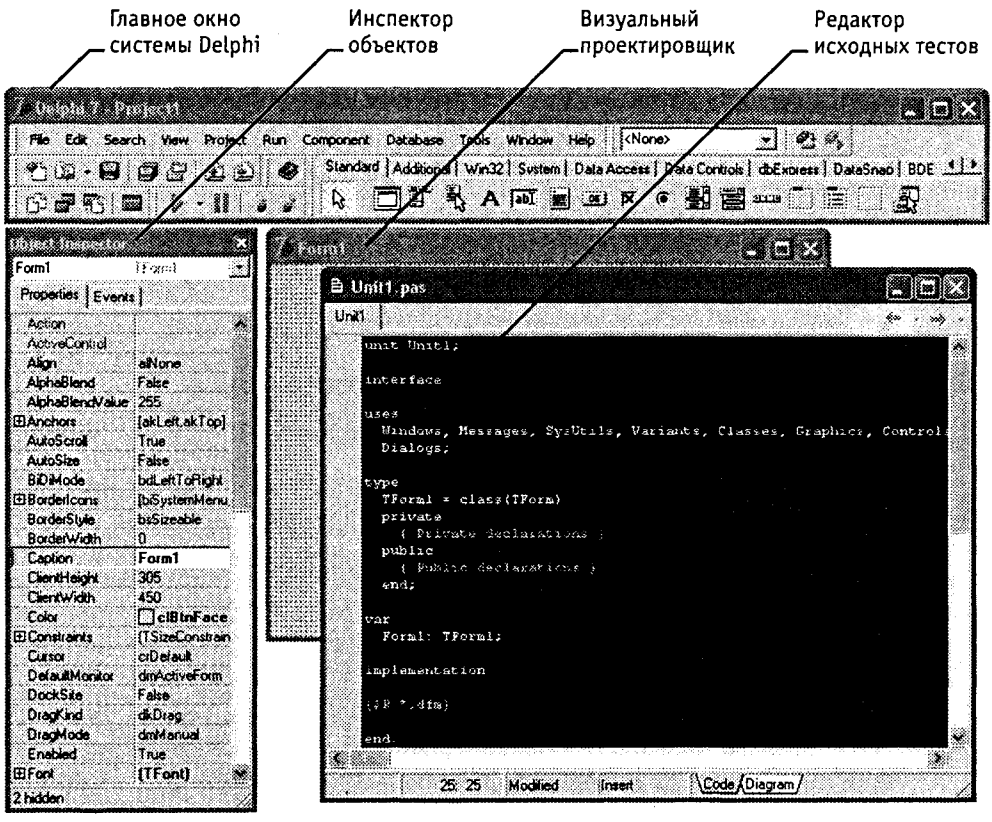


Рис. 20.2. Рабочие окна программы Delphi 7

Инспектор объектов. Используется для визуальной (без программирования) настройки свойств различных объектов на этапе проектирования.

Заголовок окна

Понять, как работает Инспектор объектов, лучше всего на примере. Пока что автоматически создана только одна пустая форма (называющаяся Form1), но она обладает множеством различных свойств. Заголовок формы (будущего окна программы) задается в свойстве Caption (Заголовок). Чтобы его изменить, надо в Инспекторе объектов найти строку, в левой части которой написано Caption, и в правой части этой строки, небольшом поле ввода, указать новое название, например Раскрашивание. Тут же изменится и заголовок формы в визуальном проектировщике.

Аналогично меняются и любые другие свойства. Сначала в выпадающем списке в верхней части Инспектора выбирается нужный объект (или он выделяется на форме щелчком мыши), затем находится название свойства и в правой части его значение меняется на новое. Изменение может происходить как путем простого ввода нового значения с клавиатуры, так и выбором одного из предопределенных значе-

ний из списка. В некоторых случаях для редактирования свойства вызывается специальный редактор.

Свойства могут быть многосоставными, например, свойство Font (Шрифт) — слева от названия такого свойства ставится символ «±». Двойным щелчком мыши на его названии оно раскрывается и показывает все свои вложенные подсвойства.

Размещение компонентов на форме

Прежде всего разместим на форме поле ввода. Для этого на палитре компонентов с помощью закладки выбирается панель Standard (Стандартная) и нажимается кнопка с всплывающей подсказкой Edit (выбран компонент «поле ввода»). Затем надо щелкнуть мышкой на форме, и в месте щелчка появится элемент управления Edit1. Его можно перетаскивать по форме и менять размеры.

В свойстве Text (Содержимое) этого объекта исходно надо задать пустую строку, чтобы при запуске программы в данном поле ничего не показывалось.

Рядом с полем ввода надо разместить свободное поле с комментарием. Для этого на панели компонентов выбирается компонент, называющийся Label (Подпись), и помещается на форме так, как это было сделано с полем ввода. Новый объект автоматически получит название Label1. Чтобы указать в нем текст «Цвет:», его надо ввести в свойство Caption.

Изменить название любого объекта на форме можно, изменив его свойство Name в Инспекторе объектов.

В нижней части формы надо разместить кнопку — компонент Button (Кнопка) на панели Standard (Стандартная). Название этой кнопки (Закрасить) задается в свойстве Caption.

На панели Additional (Дополнительно) имеется компонент Shape (Фигура). Этот компонент помещается в центр формы. Он получит название Shape1 и исходно примет форму квадрата, закрашенного белым цветом. Чтобы превратить его в круг, надо значение свойства Shape изменить на stCircle.

Теперь осталось только добавить кнопку, закрывающую форму. Это действие стандартное, поэтому в *Delphi 7* имеется специальный компонент BitBtn (Кнопка с картинкой) на панели Additional (Дополнительно), позволяющий автоматизировать такие действия, не прибегая к программированию.

После размещения такой кнопки на форме (она получит название BitBtn1) значение ее свойства Kind (Вид выполняемого действия) надо установить в bkClose (Закрыть окно). При этом на кнопке появится изображение стандартной картинки, символизирующей действие закрытия.

В завершение надо изменить заголовок этой кнопки (свойство Caption) с английского слова Close на русское Закрыть, и на этом процесс проектирования приложения можно считать законченным.

Сохранение проекта

Перед тем как приступить к программированию, проект надо сохранить. Это действие выполняется командой File ▸ Save All (Файл ▸ Сохранить все), после чего снача-

ла выбирается каталог и указывается имя файла, в котором хранится программное описание (на Паскале) структуры и работы спроектированной формы. Имя файла будет иметь расширение .PAS по умолчанию. Далее система *Delphi 7* спросит, куда и под каким именем сохранить файл проекта, содержащий всю информацию об используемых формах и модулях (их может быть в одном проекте сколько угодно, но одна форма всегда будет главной) и всевозможные настройки. Название файла проекта не должно совпадать с названием файла с исходным текстом программы.

Обработка нажатия кнопки

В создаваемой программе вручную придется запрограммировать фактически только одно событие — нажатие на кнопку Закрасить. Чтобы создать первоначально пустую подпрограмму, вызываемую при нажатии на эту кнопку, надо просто дважды щелкнуть на ней мышкой. При этом система *Delphi 7* вызовет редактор, автоматически сгенерирует нужный текст и разместит курсор именно в том месте, где можно начать описание нужного алгоритма.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
end;
```

Обработчик события Нажатие на кнопку Button1 — это обычная подпрограмма, метод класса TForm1 (этот класс описывает главную форму Form1). Единственный параметр Sender характеризует источник сообщения о случившемся событии. Его практически всегда можно игнорировать.

Алгоритм работы данного метода будет следующим. Первоначально надо убедиться, что длина введенной в поле Edit1 строки равна 6 символам и каждый из этих символов — шестнадцатеричная цифра. Если это не так, то выполнение обработчика надо сразу завершить (для этого предназначена стандартная процедура Паскаля Exit, мгновенно завершающая работу текущей подпрограммы).

Если же введенные данные корректны, их надо:

1. Преобразовать в промежуточную строку в формате $00xxxxxx, где xxxxxx — шесть введенных цифр.
2. Эту строку преобразовать в число, которое будет рассматриваться как цвет.
3. Установить новый цвет круга на основании полученного значения.

Содержимое поля ввода Edit1 хранится в виде строки в его свойстве Text. Доступ к этому свойству осуществляется с помощью конструкции Edit1.Text.

Длина строки определяется стандартной функцией length() со строкой в качестве параметра.

Стандартная функция Pos(), получая две строки как аргументы, проверяет, не содержится ли первая строка во второй, и если содержится, то возвращает номер начальной позиции. В противном случае Pos() возвращает ноль. Эта функция потребуется для определения, все ли символы во введенной строке допустимы.

Стандартная функция UpperCase() преобразует строку к верхнему регистру. Такое преобразование требуется, чтобы разрешить ввод значений цветов на любых регистрах.

Преобразование строки в число выполняет стандартная функция StrToInt().

Объект Shape1 имеет свойство Brush (Кисть для фона), которое, в свою очередь, имеет вложенное свойство Color (Цвет заливки). Его и надо в конечном счете изменить. Как только это произойдет, цвет круга в окне автоматически изменится на новый.

```
procedure TForm1.Button1Click(Sender: TObject);
var i: integer;
    s: string;
begin
// если длина введенной строки не равна 6,
// то закончить работу
if length(Edit1.Text) <> 6 then exit;
// в локальную переменную s заносится строка,
// содержащая допустимые символы
s := "0123456789ABCDEF";
// проверяется каждый символ во введенной строке
for i := 1 to 6 do
// если очередной символ не найден в строке s, значит,
// он недопустим, и работу требуется прекратить
  if pos(UpperCase(Edit1.Text[i]), s) = 0 then exit;
// все нормально — в переменной s
// готовим промежуточную строку
s := "$00"+Edit1.Text;
// Устанавливаем значение цвета заливки круга равным
// числу, преобразованному из строки в переменной s
Shape1.Brush.Color := StrToInt(s);
end;
```

Запуск программы

Программа запускается нажатием на клавишу F9. Так как *Delphi 7* — это компилирующая система, сначала автоматически выполнится компиляция и только потом программа запустится. Задавая различные строки (FF0FFF, abcdef, 987654 и т. п.), можно наглядно увидеть соответствующие им цвета.

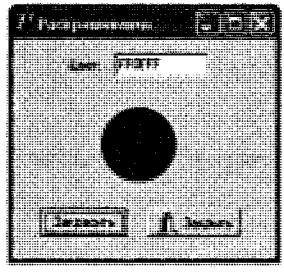


Рис 20.3. Программа закраски в работе

20.9. Пример на Си++. Рисование графиков

Система *Microsoft Visual Studio .NET* во многом схожа с рассмотренной средой *Delphi*, но визуальные возможности построения пользовательского интерфейса реализованы в ней только для Бейсика и C#. Поддержка Си++ в этой системе унаследована от старых версий и не предоставляет разработчику столь удобного проектирования. Тем не менее мы рассмотрим простой пример ее использования для создания программы, рисующей график функции.

Постановка задачи

Некоторая подпрограмма задает зависимость значения функции от аргумента. Надо нарисовать в окне график, показывающий эту зависимость.

Принципы рисования в Visual Studio

Перерисовывать экран в *Windows* приходится по самым разным причинам. Например, окно было закрыто другими приложениями, свернуто или оказалось временно заслоненным своими вспомогательными окнами. При этом перерисовывать приходится или все содержимое, или только часть. Программа, созданная с помощью *Visual Studio*, сама определяет, что и когда ей надо перерисовать, и все элементы управления тоже это «понимают». Особое требование к организации перерисовки возникает, только когда программист напрямую использует функции рисования. Все эти функции в таком случае надо размещать в обработчике события OnDraw, которое вызывается автоматически.

Технология рисования

Система *Microsoft Visual Studio .NET* не содержит визуальных средств создания программ на Си++, аналогичных возможностям *Delphi*. Она предлагает дизайнеры форм только для Бейсика и C#. Поэтому мы изучим несложный вариант использования графических примитивов *Windows* для демонстрации техники рисования графиков в пределах клиентского окна шаблонного приложения. Любое окно *Windows* характеризуется так называемым контекстом устройства, своеобразным объектом, содержащим различные методы графического вывода в пределах этого окна. Доступ к контексту нужного нам окна будет автоматически предоставлен *Microsoft Visual Studio .NET* при формировании обработчика OnDraw.

Для создания графика потребуются два метода этого объекта: метод MoveTo(x,y), устанавливающий новое начальное положение — точку (x, y) для следующих операций рисования, и метод LineTo(x,y), проводящий линию из предыдущей точки в новую.

Метод отрисовки

Так как система *Microsoft Visual Studio .NET* не дает возможности работать с формой напрямую, подготовка «пустого» приложения будет немного сложнее, чем в предыдущих примерах. После запуска системы надо дать команду File ▸ New ▸ Project (Файл ▸ Создать ▸ Проект), на панели Project Types (Типы проектов) выбрать раздел

Visual C++ Projects (Проекты Visual C++), а на панели Templates (Шаблоны) — значок MFC Application (Оконное приложение).

Название проекта (например, Grafiki) и его местонахождение задается в полях Name (Имя) и Location (Расположение). После нажатия на кнопку OK запускается Мастер настройки вида будущего приложения. Изменять в нем ничего не надо, достаточно нажать кнопку Finish (Готово). Система сгенерирует заготовку пустого, но работоспособного проекта. Посмотреть его структуру можно с помощью средства Просмотра решения (*Solution Explorer*), вызываемого командой View ▸ Solution Explorer (Вид ▸ Просмотр решения).

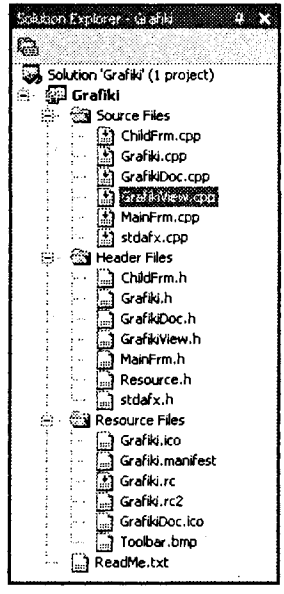


Рис. 20.4. Структура проекта Visual Studio .NET, сформированная автоматически

В этом множестве автоматически сгенерированных файлов нас интересует файл GrafikiView.cpp, непосредственно ответственный за отображение содержимого клиентских окон на экране. Чтобы открыть его в редакторе, надо дважды щелкнуть мышкой на соответствующей строке.

Алгоритм отображения графика несложен. Он умещается в нескольких операторах.

Координату по оси Y нельзя взять непосредственно из переменной y, а надо вычислять по формуле Height–y, потому что в системе *Windows* считается, что точка с координатами (0,0) расположена в верхнем левом углу окна, а ось Y направлена вниз. Для удобства восприятия эту ось надо перевернуть.

Необходимо сформировать обработчик события OnDraw. Для этого перейдем в раскрывающийся список в верхнем правом углу окна редактора исходных текстов, и найдем в нем строку OnDraw. После ее выбора в редакторе появится следующий новый текст:

```
// CGrafikiView drawing

void CGrafikiView::OnDraw(CDC* /*pDC*/)
{
  CGrafikiDoc* pDoc = GetDocument();
  ASSERT_VALID(pDoc);

  // TODO: add draw code for native data here
}
```

Параметр pDC — это нужный нам указатель на контекст устройства. Его мы и будем использовать для вывода графиков. Обратите внимание, что по умолчанию он взят в скобки-комментарии и недоступен внутри функции. Поэтому его надо раскомментировать.

```
void CGrafikiView::OnDraw(CDC* pDC)
{
  CGrafikiDoc* pDoc = GetDocument();
  ASSERT_VALID(pDoc);

  // TODO: add draw code for native data here

// начальные координаты
  int x,y;
  x = 0;
  y = 0;

// переменная-прямоугольник
  CRect rect;

// определяем размеры клиентского окна
// вызовом стандартной функции Windows
    GetClientRect(&rect);

// фиксируем начальную точку
  pDC->MoveTo(0,rect.Height());

// цикл, пока каждая координата очередной точки
// укладывается в клиентскую область
  while( x < rect.Width() && y < rect.Height() )
  {
  x = x + 1;
// соответствующее значение по оси Y
  y = f(x);

// в новую точку (x,rect.Height()-y)
// рисуется линия
  pDC->LineTo(x,rect.Height()-y);
  }
```

Чуть выше метода OnDraw надо определить функцию f(), не привязанную ни к какому классу. В ней происходит вычисление значения анализируемой математической функции по заданному аргументу. Для примера, она может выглядеть так:

```
int f(int x)
{
int y;
y = 50*log(x);
return y;
}
```

Стандартная функция log() вычисляет значение логарифма. Коэффициент 50 нужен, чтобы кривая пропорционально размещалась в окне. Хотя функция log() рассчитывает действительное значение, компилятор автоматически настроит программный код так, чтобы оно было преобразовано в целый тип, соответствующий типу переменной y.

Исходно функция log() и ряд других не подключены к текущему проекту. Чтобы они стали доступными, библиотеку, в которой они хранятся, необходимо явно указать компилятору. Делается это с помощью командной строки

```
#include "Math.h"
```

которую можно поместить в самое начало текущего файла.

Далее проект надо сохранить, выполнить компиляцию и запустить, нажав клавишу F5. В дальнейшем, изменив один оператор присваивания в функции f() и подобрав подходящие коэффициенты, с помощью этой программы можно строить самые разные графики.

Практические задания по программированию

Задание 1

Дано натуральное число. Составить программу, которая представляет данное число в виде суммы квадратов натуральных чисел, содержащей минимальное число слагаемых. Например:

$9=3^2$

$12=2^2+2^2+2^2$

$23=3^2+3^2+2^2+1^2$

Задание 2

Дан массив, содержащий N элементов.

Написать подпрограммы, выполняющие следующие действия:

- перестановку элементов массива в обратном порядке;
- вычисление суммы A[1] + A[2]*A[2] + A[3]*A[3]*A[3] ...;

- определение элементов массива, разность модулей которых имеет наибольшее значение;
- определение значения, которое встречается среди элементов массива максимальное число раз, и вычисление количества таких вхождений;
- упорядочение элементов массива по возрастанию.

Задание 3

Дан двумерный массив, содержащий $N \times N$ элементов.

Написать подпрограммы, выполняющие следующие действия:

- вычисление среднего арифметического для элементов каждой строки массива;
- замену нулями всех элементов, расположенных на главной диагонали матрицы;
- определение наибольшего элемента и его положения в массиве.

Задание 4

Дана текстовая строка.

Написать подпрограммы, выполняющие следующие действия:

- подсчет количества слов в строке (в качестве границ слов рассматриваются пробелы);
- подсчет количества цифр в строке;
- определение десятичного числа, которому соответствует строка, если она представляет запись этого числа в шестнадцатеричной системе;
- проверку соответствия содержимого строки правилам записи идентификаторов языков программирования.