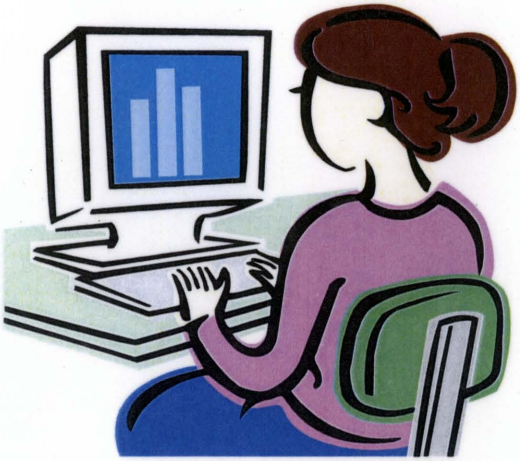
Л. С. Таганов В. Г. Левин



ИНФОРМАТИКА

Учебное пособие

Кемерово 2006

Предисловие

Пособие включает предисловие введение, девять глав и заключение. В первой главе рассмотрены понятия данных и информации, а также система их кодирования. Во второй главе рассмотрены основы и методы защиты информации. В третьей главе изложена информация о технических и программных средствах реализации информационных процессов, а также общие понятия о локальных и глобальных компьютерных сетях. В четвертой главе содержатся сведения об операционных системах. В пятой главе дана характеристика программы MS Excel и структура электронных таблиц. В шестой главе изложены сведения о процессорах MS Word 2000 и 2003, а также некоторые технологии работы с документами. В седьмой главе дано понятие баз данных и изложена технология работы с базами данных Microsoft Access в режиме конструктора. В восьмой главе рассмотрены основные понятия алгоритмизации и базовые структуры алгоритмов. В девятой главе дан обзор языков программирования, рассмотрены технологии программирования и изложена информация об основах среды программирования VBA с множеством примеров, показывающих применение различных составляющих языка VBA.

Настоящее пособие в полном объеме охватывает все разделы дисциплины "Информатика", является существенным дополнением лекционного курса и может быть весьма полезным при самостоятельном изучении дисциплины.

Введение

Предмет **"Информатика"** – это учебная дисциплина, изучающая технологии создания, хранения, воспроизведения и обработки данных (информации) средствами вычислительной техники, а также принципы функционирования этих средств и методы управления ими.

Изучение дисциплины "Информатика" имеет следующие цели:

- формирование общих представлений о возможностях использования средств вычислительной техники;
- ознакомление с основами современных информационных технологий (сбора, обработки, хранения и передачи информации) и тенденциями их развития;

- обучение применению современных информационных технологий в профессиональной деятельности и проведению анализа полученных результатов;
- развитие навыков алгоритмического мышления;
- овладение приемами работы с современными типовыми пакетами прикладных программ (MS Excel, MS Word и MS Access), обеспечивающих широкие возможности обработки информации.

В результате изучения дисциплины студенты должны:

Знать:

- современное состояние уровня и направлений развития аппаратных и программных средств;
- основы современных информационных технологий переработки информации и их влияние на успех в профессиональной деятельности;
- основы и методы защиты информации;
- основы алгоритмизации (свойства алгоритмов, базовые структуры алгоритмов);
- основы системы объектно-ориентированного программирования Visual Basic for Applications (VBA).

Уметь:

- уверенно работать в качестве пользователя персонального компьютера, самостоятельно использовать внешние носители информации, создавать резервные копии и архивы данных и программ;
- работать с программными средствами общего назначения;
- владеть приемами антивирусной защиты;
- грамотно применять технологические средства MS Excel для решения вычислительных задач, MS Word для создания и оформления документов и MS Access для работы с базами данных в режиме конструктора;
- составлять и отлаживать программы в среде VBA.

Глава 1. Данные

1.1. Понятие данных и информации

Все, что нас окружает, и с чем мы ежедневно сталкиваемся, относится либо к физическим телам, либо к физическим полям.

Известно, что все физические объекты находятся в состоянии непрерывного движения 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 физических телах 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.

Итак, данные – это зарегистрированные сигналы.

Обработка данных адекватными им методами создает новый продукт – информацию. Таким образом, информация возникает и существует 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 то, что 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.

Итак, информация – это продукт взаимодействия данных и адекватных им методов.

1.2. 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. В структуре возможных операций с данными можно выделить следующие операции:

- сбор данных – накопление информации с целью обеспечения достаточной полноты для принятия решения;
- форматизация 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 тех данных, в которых нет необходимости для принятия решения; при этом должны возрастать достоверность и адекватность информации;
- сортировка данных – упорядочение данных по заданному признаку с целью удобства их использования; при этом должна повышаться 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;
- преобразование данных – перевод данных из одной формы в другую или из одной структуры в другую;
- защита данных – комплекс мер, направленных на предотвращение утраты, воспроизведения и модификации данных;
- транспортировка данных – прием и передача данных между удаленными участниками информационного процесса.

1.3. Виды и типы данных

Данные могут быть представлены следующими видами:

- целыми и действительными числами;
- текстом;
- мультимедийными (графическими объектами, звуковыми сигналами, цветными изображениями).

В зависимости от вида данных, они могут подразделяться на следующие типы:

- байтовый тип;
- целочисленные типы простой и двойной точности;
- типы действительных чисел простой и двойной точности;
- типы даты и времени;
- строковый тип;
- логический тип;
- тип объектов.

1.4. Кодирование данных двоичным кодом

Для автоматизации работы с данными, относящимися к различным типам, очень важно унифицировать их форму представления – для этого используется прием кодирования, то есть выражение данных одного типа через данные другого типа. В вычислительной технике применяется система кодирования двоичным кодом. Она основана на представлении данных последовательностью всего двух знаков **0** и **1**. Эти знаки называются двоичными цифрами, каждая из которых представляет 1 бит. Одним битом могут быть выражены два понятия: **0** или **1** (да или нет, черное или белое, истина или ложь и т. п.). Двумя битами можно выразить четыре различных понятия: **00, 01, 10, 11**.

Тремя битами можно закодировать 8 различных значений: **000, 001, 010, 011, 100, 101, 110, 111**.

Увеличивая на единицу количество разрядов в системе двоичного кодирования, в два раза увеличивается количество кодируемых значений. Общая формула расчета имеет вид: $\mathbf{N = 2^m}$,

где **N** – количество независимых кодируемых значений;

m – количество разрядов двоичного кодирования.

1.4.1. Кодирование целых и действительных чисел

Двоичный код целого числа можно получить путем деления числа на 2 до тех пор, пока частное не будет равно 1. Совокупность остатков от каждого деления, записанная справа налево вместе с последним частным, и образует двоичный аналог десятичного целого числа.

Примеры:

$47 : 2 = 23 + 1$
$23 : 2 = 11 + 1$
$11 : 2 = 5 + 1$
$5 : 2 = 2 + 1$
$2 : 2 = 1 + 0$

Итак: $47_{10} = 111101_2$

$252 : 2 = 126 + 0$
$126 : 2 = 63 + 0$
$63 : 2 = 31 + 1$
$31 : 2 = 15 + 1$
$15 : 2 = 7 + 1$
$7 : 2 = 3 + 1$
$3 : 2 = 1 + 1$

$252_{10} = 00111111_2$.

Для кодирования целых чисел от 0 до 255 достаточно иметь 8 разрядов двоичного кода (8 бит). Для кодирования чисел от 0 до 65535 потребуется 16 разрядов (16 бит). Используя 24 разряда (24 бита), можно закодировать более 16,5 миллиона разных значений.

Для кодирования действительных чисел используется 80 разрядов (80 бит). При этом действительное число предварительно преобразуется в нормализованную форму: $41{,}2346785 = 0{,}412346785 \cdot 10^2$. Первая часть нормализованного числа называется мантиссой, а вторая – характеристикой. При этом значительная часть из 80 бит задействуется для хранения мантиссы (вместе со знаком числа) и некоторое фиксированное количество бит отводится для хранения характеристики (тоже со знаком степени).

1.4.2. Кодирование текстовых данных

Если каждому символу присвоить порядковый номер (целое число), то с помощью двоичного кода можно кодировать любые текстовые данные. Восьми двоичных разрядов достаточно для кодирования 256 различных символов. Этого хватит, чтобы закодировать комбинациями 8 битов все символы английского и русского алфавитов (строчные и прописные), арабские цифры, знаки препинания, символы арифметических действий и некоторые общепринятые специальные символы.

С этой целью 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 по 255.

Первые 32 кода (от 0 до 31) базовой таблицы выделены производителям аппаратных средств (в первую очередь компьютеров и печатающих устройств). Это управляющие коды, которым не соответствуют никакие символы, ими можно управлять работой технических устройств.

Коды от 32 по 127 предназначены для кодирования символов английского алфавита, знаков препинания, цифр, арифметических действий и некоторых вспомогательных символов. Символы русского

алфавита и другие специальные символы кодируются кодами расширенной таблицы от 128 по 255.

Однако, рассмотренная выше система кодирования ASCII не обеспечивает кодирование алфавитов многих других языков планеты. С целью устранения этого недостатка в настоящее время создана универсальная система – UNICODE, основанная на 16-разрядном кодировании символов. Эта система позволяет обеспечить уникальные коды для 65536 различных символов. Этого количества достаточно для размещения в одной таблице символов большинства различных алфавитов планеты.

1.4.3. Кодирование графических данных

Если графическое изображение рассматривать как комбинацию мельчайших точек, образующих определенный узор, называемый растром, то с помощью линейных координат и индивидуальных свойств каждой точки, выраженных с помощью целых чисел, можно применить систему двоичного кодирования и для графических данных. К индивидуальным свойствам точки относятся яркость и цвет.

Черно-белые иллюстрации представляются в виде комбинации точек с 256 градациями серого цвета. Таким образом, для кодирования яркости любой точки достаточно 8 разрядов двоичного числа.

Кодирование цветных графических изображений осуществляется на принципе декомпозиции произвольного цвета на основные составляющие. В качестве таких составляющих используются три цвета: красный (***Red, R***), зеленый (***Green, G***) и синий (***Blue, B***). Такое кодирование называется системой ***RGB***. При этом если для кодирования яркости каждой из основных составляющих использовать по 256 значений (8 двоичных разрядов), то на кодирование цвета одной точки требуется 24 разряда. Такая система кодирования обеспечивает 16,5 миллиона цветов. Эта система является полноцветной и называется ***True Color***. Если уменьшить количество двоичных разрядов, используемых для кодирования цвета каждой точки, то можно сократить объем данных, но при этом заметно сократится диапазон кодируемых цветов. Кодирование цветной графики двоичными числами, содержащими 16 разрядов, называется ***High Color***.

На практике применяется индексный метод кодирования информации о цвете. При этом код каждой точки растра выражает не

цвет сам по себе, а только его номер (индекс) в справочной таблице, называемой ***палитрой***, которая прилагается к графическим данным.

1.4.4. Кодирование звука

Для кодирования звуковой информации применяется метод таблично-волнового синтеза (***Wave-Table***). Сущность этого метода состоит в том, что используются заранее подготовленные таблицы образцов звуков. В технике такие образцы называют ***сэмплами***. Числовые коды звуковой информации выражают: тип инструмента и номер его модели, высоту тона, продолжительность, интенсивность звука и динамику его изменения. А также некоторые параметры среды, в которой происходит звучание, и прочие параметры, характеризующие особенности звука.

1.5. 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, табличная и иерархическая.** При создании любой структуры данных необходимо обеспечить решение двух задач: как разделять элементы данных между собой и как разыскивать нужные элементы.

Линейные структуры – это хорошо знакомые списки. Список – это простейшая структура данных, отличающаяся тем, что каждый элемент данных однозначно определяется своим уникальным номером в массиве (списке).

Табличные структуры данных подразделяются на двумерные и многомерные.

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

Многомерные таблицы – это упорядоченные структуры данных, в которых адрес элемента определяется тремя и более измерениями. Для отыскания нужного элемента в таких таблицах необходимо знать параметры всех измерений (размерностей).

Линейные 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 таких данных является сортировка. Недостатком простых структур данных является трудность их обновления. При добавлении, например, произвольного элемента в упорядоченную структуру возникает необходимость изменения адресных данных у других элементов.

Иерархические структуры – это структуры, объединяющие нерегулярные данные, которые трудно представить в виде списка или таблицы. В иерархической структуре адрес каждого элемента определяется маршрутом, ведущим от вершины структуры к данному элементу. Эти структуры по форме сложнее, чем линейные и табличные, но они не создают проблем с обновлением данных. Их легко развивать путем создания новых уровней. Недостатком иерархических структур является относительная трудоемкость записи адреса элемента данных и сложность упорядочения. Поэтому для упорядочения в таких структурах применяется метод предварительной индексации. При этом каждому элементу данных присваивается свой уникальный индекс, который используется при поиске, сортировке и тому подобное. В качестве примера иерархической структуры может служить система почтовых адресов.

1.6. Единицы представления, измерения, хранения и передачи данных

Одной из систем представления данных, принятых в информатике и вычислительной технике является система двоичного кодирования. Наименьшей единицей такого представления является **бит** (*двоичный разряд*).

Совокупность двоичных разрядов, выражающих числовые или иные данные, образует некий битовый рисунок. С битовым представлением удобнее работать, если этот рисунок имеет регулярную форму. В качестве таких форм используются группы из 8 битов, каждая из которых называется ***байтом***. Однако во многих случаях целесообразно использовать 16-разрядное, 24-разрядное, 32-разрядное, 64-разрядное кодирование.

Байт является наименьшей единицей измерения количества данных (информации).

Более крупные единицы измерения данных образуются добавлением префиксов кило-, мега-, гига-, тера-.

1 Килобайт (Кбайт) = 1024 байт = 2^{10} байт.

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

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

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

В более крупных единицах пока нет практической надобности.

В качестве единицы хранения данных (информации) принят объект переменной величины, называемый файлом.

Файл – это последовательность произвольного числа байтов, обладающая уникальным собственным именем.

Поскольку в определении файла нет ограничений на его размер, то можно представить себе файл, имеющий 0 байтов (пустой файл), и файл, имеющий любое число байтов. В определении файла особое внимание уделяется имени. Имя файла фактически несет в себе адресные данные, без которых данные, хранящиеся в файле, не станут информацией из-за отсутствия методов доступа к ним. Кроме адресных функций, имя файла может хранить сведения о типе данных, заключенных в нем.

Требование уникальности имени файла в вычислительной технике обеспечивается автоматически – создать файл с именем, тождественным уже существующему, не может ни пользователь, ни автоматика. Уникальность имени файла обеспечивается тем, что полным именем файла считается собственное имя файла вместе с путем доступа к нему.

Хранение файлов организуется в иерархической структуре, которая 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).

Синтаксис записи полного имени файла:

Имя носителя \ *Имя каталога* 1 \ *Имя каталога* N \ *Собственное имя файла.*

Пример: *C\Игры\Стрелялки\Кролики.*

Передача данных в компьютерных системах измеряется ее скоростью. Единицей измерения скорости передачи данных через последовательные порты является: бит в секунду (бит/с, Кбит/с, Мбит/с). Единицей измерения скорости передачи данных через параллельные порты является байт в секунду (байт/с, Кбайт/с, Мбайт/с).

Вопросы для самоконтроля

1. В чем состоит отличие между данными и информацией (понятие данных, понятие информации)?
2. Какие основные операции можно осуществлять с данными?
3. Какие достоинства и недостатки присущи основным структурам данных?
4. Как осуществляется кодирование двоичным кодом?
5. Что собой представляет система кодирования ASCII (базовая и расширенная таблицы кодов)?
6. Какая современная система кодирования обеспечивает кодирование большинства алфавитов планеты?
7. Как называются единицы представления, измерения, хранения и передачи данных? Какова их размерность?

Глава 2. Основы защиты информации

2.1. Информационная безопасность и ее составляющие

С технологической точки зрения информация является продукцией информационных систем. Как и для всякого продукта, для информации большое значение имеет ее качество, то есть способность удовлетворять определенные информационные потребности.

Качество информации является сложным понятием, его основу составляет базовая система показателей, включающая показатели трех классов:

- класс выдачи (своевременность, актуальность, полнота, доступность и другие);
- класс обработки (достоверность, адекватность и другие);
- класс защищенности информации (физическая целостность, логическая целостность, безопасность).

Своевременность информации оценивается временем выдачи (получения), в течение которого информация не потеряла свою актуальность.

Актуальность информации – это степень ее соответствия текущему моменту времени. Нередко с актуальностью связывают коммерческую ценность информации. Устаревшая и потерявшая свою актуальность информация может приводить к ошибочным решениям и тем самым теряет свою практическую ценность.

Полнота информации определяет достаточность данных для принятия решений или для создания новых данных на основе имеющихся. Чем полнее данные, тем проще подобрать метод, вносящий минимум погрешностей в ход информационного процесса.

Достоверность информации – это степень соответствия между получаемой и исходящей информацией.

Адекватность информации – это степень соответствия реальному объективному состоянию дела. Неадекватная информация может образовываться при создании новой информации на основе неполных или недостаточных 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. Отсутствие доступа к данным или отсутствие адекватных методов обработки данных приводят к одинаковому результату: информация оказывается недоступной.

Одним из наиболее существенных показателей качества информации является ее безопасность.

В качестве предмета защиты рассматривается информация, хранящаяся, обрабатываемая и передаваемая в компьютерных системах (КС). Особенностями этой информации являются:

- двоичное ее представление внутри системы, независимо от физической сущности носителей исходной информации;
- высокая степень автоматизации обработки и передачи информации;
- концентрация большого количества информации в КС.

Понятие компьютерные системы (КС) охватывает следующие системы:

- ЭВМ всех классов и назначений;
- вычислительные комплексы и системы;

- вычислительные сети (локальные, глобальные).

Информация доступна человеку, если она содержится на материальном носителе. Поэтому необходимо защищать материальные носители информации, так как с помощью материальных средств можно защищать только материальные объекты.

Информация имеет ценность, которая определяется степенью ее полезности для владельца. В свою очередь степень полезности информации зависит от ее истинности или достоверности. Истинной информацией является та, которая с достаточной точностью отражает объекты и процессы окружающего мира в определенных временных и пространственных рамках. Если информация искажена, то она является дезинформацией. Если к информации ограничен доступ, то такая информация является конфиденциальной. Конфиденциальная информация может содержать государственную или коммерческую тайну.

Государственную тайну могут содержать сведения, принадлежащие государству. Сведениям, представляющим ценность для государства, могут быть присвоены следующие степени секретности (гриф):

- особой важности;
- совершенно секретно;
- секретно;
- для служебного пользования.

Коммерческую тайну могут содержать сведения, принадлежащие частному лицу, фирме, корпорации и тому подобное. Сведениям, представляющим коммерческую тайну, могут быть присвоены следующие категории:

- строго конфиденциально или строго конфиденциально – строгий учет;
- конфиденциально или строго конфиденциально;
- конфиденциально.

Безопасность (защищенность) информации в КС – это состояние всех составляющих компьютерной системы, обеспечивающее на требуемом уровне защиту информации от возможных угроз.

Безопасность информации в КС (информационная безопасность) является одним из основных направлений обеспечения безопасности государства, отрасли, ведомства, государственной организации или частной структуры.

Информационная безопасность достигается проведением руководством соответствующего уровня политики информационной безопасности. Основным документом, на основе которого проводится политика информационной безопасности, является программа информационной безопасности. Этот документ разрабатывается и принимается как официальный руководящий документ высшими органами управления государством, ведомством, организацией. На основе этого документа создается комплексная система защиты информации на уровне соответствующей структуры (государства, отрасли, ведомства, учреждения).

Под системой защиты информации в КС понимается единый комплекс правовых норм, организационных мер, технических, программных и криптографических средств, обеспечивающий защищенность информации в КС в соответствии с принятой политикой безопасности.

2.2. Угрозы безопасности информации в компьютерных системах

Под угрозой безопасности информации понимается потенциально возможное событие, процесс или явление, которое может привести к уничтожению, утрате целостности, конфиденциальности или доступности информации.

Все множество потенциальных угроз безопасности информации в автоматизированных информационных системах (АИС) или в компьютерных системах (КС) может быть разделено на два класса: случайные угрозы и преднамеренные угрозы.

Угрозы, которые не связаны с преднамеренными действиями злоумышленников и реализуются в случайные моменты времени, называются случайными или непреднамеренными.

К случайным угрозам относятся: стихийные бедствия и аварии, сбои и отказы технических средств, ошибки при разработке АИС или КС, алгоритмические и программные ошибки, ошибки пользователей и обслуживающего персонала.

Реализация угроз этого класса приводит к наибольшим потерям информации (по статистическим данным – до 80 % от ущерба, наносимого информационным ресурсам КС любыми угрозами). При этом может происходить уничтожение, нарушение целостности и доступности информации. Реже нарушается конфиденциальность

информации, однако при этом создаются предпосылки для злоумышленного воздействия на информацию. Согласно тем же статистическим данным только в результате ошибок пользователей и обслуживающего персонала происходит до 65 % случаев нарушения безопасности информации.

Следует отметить, что механизм реализации случайных угроз изучен достаточно хорошо и накоплен значительный опыт противодействия этим угрозам. Современная технология разработки технических и программных средств, эффективная система эксплуатации автоматизированных информационных систем, включающая обязательное резервирование информации, позволяют значительно снизить потери от реализации угроз этого класса.

Угрозы, которые связаны со злоумышленными действиями людей, а эти действия носят не просто случайный характер, а, как правило, являются непредсказуемыми, называются преднамеренными.

К преднамеренным угрозам относятся: традиционный или универсальный шпионаж и диверсии, несанкционированный доступ к информации, электромагнитные излучения и наводки, несанкционированная модификация структур, вредительские программы.

В качестве источников нежелательного воздействия на информационные ресурсы по-прежнему актуальны методы и средства шпионажа и диверсий. К методам шпионажа и диверсий относятся: подслушивание, визуальное наблюдение, хищение документов и машинных носителей информации, хищение программ и атрибутов систем защиты, подкуп и шантаж сотрудников, сбор и анализ отходов машинных носителей информации, поджоги, взрывы, вооруженные нападения диверсионных или террористических групп.

Несанкционированный доступ к информации – это нарушение правил разграничения доступа с использованием штатных средств вычислительной техники или автоматизированных систем. Несанкционированный доступ возможен:

- при отсутствии системы разграничения доступа;
- при сбое или отказе в компьютерных системах;
- при ошибочных действиях пользователей или обслуживающего персонала компьютерных систем;
- при ошибках в системе распределения доступа;
- при фальсификации полномочий.

Процесс обработки и передачи информации техническими средствами компьютерных систем сопровождается электромагнитными излучениями в окружающее пространство и наведением электрических сигналов в линиях связи, сигнализации, заземлении и других проводниках. Все это получило название "побочные электромагнитные излучения и наводки" (ПЭМИН). Электромагнитные излучения и наводки могут быть использованы злоумышленниками как для получения информации, так и для ее уничтожения.

Большую угрозу безопасности информации в компьютерных системах представляет несанкционированная модификация алгоритмической, программной и технической структуры системы.

Одним из основных источников угроз безопасности информации в КС является использование специальных программ, получивших название "вредительские программы".

В зависимости от механизма действия вредительские программы делятся на четыре класса:

- "логические бомбы";
- "черви";
- "троянские кони";
- "компьютерные вирусы".

Логические бомбы – это программы или их части, постоянно находящиеся в ЭВМ или в КС и выполняемые только при соблюдении определенных условий. Примерами таких условий могут быть: наступление заданной даты, переход КС в определенный режим работы, наступление некоторых событий заданное число раз и тому подобное.

Черви – это программы, которые выполняются каждый раз при загрузке системы, обладают способностью перемещаться в вычислительных системах (ВС) или в сети и самовоспроизводить копии. Лавинообразное размножение программ приводит к перегрузке каналов связи, памяти и блокировке системы.

Троянские кони – это программы, полученные путем явного изменения или добавления команд в пользовательские программы. При последующем выполнении пользовательских программ наряду с заданными функциями выполняются несанкционированные, измененные или какие-то новые функции.

Компьютерные вирусы – это небольшие программы, которые после внедрения в ЭВМ самостоятельно распространяются путем

создания своих копий, а при выполнении определенных условий оказывают негативное воздействие на КС.

2.3. Методы защиты информации

Защита информации в компьютерных системах обеспечивается созданием комплексной системы защиты. Комплексная система защиты включает:

- правовые методы защиты;
- организационные методы защиты;
- методы защиты от случайных угроз;
- методы защиты от традиционного шпионажа и диверсий;
- методы защиты от электромагнитных излучений и наводок;
- методы защиты от несанкционированного доступа;
- криптографические методы защиты;
- методы защиты от компьютерных вирусов.

Среди методов защиты имеются и универсальные, которые являются базовыми при создании любой системы защиты. Это, прежде всего, правовые методы защиты информации, которые служат основой легитимного построения и использования системы защиты любого назначения. К числу универсальных методов можно отнести и организационные методы, которые используются в любой системе защиты без исключений и, как правило, обеспечивают защиту от нескольких угроз.

Методы защиты от случайных угроз разрабатываются и внедряются на этапах проектирования, создания, внедрения и эксплуатации компьютерных систем. К их числу относятся:

- создание высокой надежности компьютерных систем;
- создание отказоустойчивых компьютерных систем;
- блокировка ошибочных операций;
- оптимизация взаимодействия пользователей и обслуживающего персонала с компьютерной системой;
- минимизация ущерба от аварий и стихийных бедствий;
- дублирование информации.

При защите информации в компьютерных системах от традиционного шпионажа и диверсий используются те же средства и методы защиты, что и для защиты других объектов, на которых не используются компьютерные системы. К их числу относятся:

- создание системы охраны объекта;
- организация работ с конфиденциальными информационными ресурсами;
- противодействие наблюдению и подслушиванию;
- защита от злоумышленных действий персонала.

Все методы защиты от электромагнитных излучений и наводок можно разделить на пассивные и активные. Пассивные методы обеспечивают уменьшение уровня опасного сигнала или снижение информативности сигналов. Активные методы защиты направлены на создание помех в каналах побочных электромагнитных излучений и наводок, затрудняющих прием и выделение полезной информации из перехваченных злоумышленником сигналов. На электронные блоки и магнитные запоминающие устройства могут воздействовать мощные внешние электромагнитные импульсы и высокочастотные излучения. Эти воздействия могут приводить к неисправности электронных блоков и стирать информацию с магнитных носителей информации. Для блокирования угрозы такого воздействия используется экранирование защищаемых средств.

Для защиты информации от несанкционированного доступа создаются:

- система разграничения доступа к информации;
- система защиты от исследования и копирования программных средств.

Исходной информацией для создания системы разграничения доступа является решение администратора компьютерной системы о допуске пользователей к определенным информационным ресурсам. Так как информация в компьютерных системах хранится, обрабатывается и передается файлами (частями файлов), то доступ к информации регламентируется на уровне файлов. В базах данных доступ может регламентироваться к отдельным ее частям по определенным правилам. При определении полномочий доступа администратор устанавливает операции, которые разрешено выполнять пользователю. Различают следующие операции с файлами:

- чтение (R);
- запись;
- выполнение программ (E).

Операции записи имеют две модификации:

- субъекту доступа может быть дано право осуществлять запись с изменением содержимого файла (W);
- разрешение дописывания в файл без изменения старого содержимого (A).

Система защиты от исследования и копирования программных средств включает следующие методы:

- методы, затрудняющие считывание скопированной информации;
- методы, препятствующие использованию информации.

Под криптографической защитой информации понимается такое преобразование исходной информации, в результате которого она становится недоступной для ознакомления и использования лицами, не имеющими на это полномочий. По виду воздействия на исходную информацию методы криптографического преобразования информации разделяются на следующие группы:

- шифрование;
- стенография;
- кодирование;
- сжатие.

Вредительские программы и, прежде всего, вирусы представляют очень серьезную опасность для информации в компьютерных системах. Знание механизмов действия вирусов, методов и средств борьбы с ними позволяет эффективно организовать противодействие вирусам, свести к минимуму вероятность заражения и потерь от их воздействия.

Компьютерные вирусы — это небольшие исполняемые или интерпретируемые программы, обладающие свойством распространения и самовоспроизведения в компьютерных системах. Вирусы могут выполнять изменение или уничтожение программного обеспечения или данных, хранящихся в компьютерных системах. В процессе распространения вирусы могут себя модифицировать.

Все компьютерные вирусы классифицируются по следующим признакам:

- по среде обитания;
- по способу заражения;
- по степени опасности вредительских воздействий;
- по алгоритму функционирования.

По среде обитания компьютерные вирусы подразделяются на:

- сетевые;
- файловые;

- загрузочные;
- комбинированные.

Средой обитания **сетевых** вирусов являются элементы компьютерных сетей. **Файловые** вирусы размещаются в исполняемых файлах. **Загрузочные** вирусы находятся в загрузочных секторах внешних запоминающих устройств. **Комбинированные** вирусы размещаются в нескольких средах обитания. Например, загрузочно-файловые вирусы.

По способу заражения среды обитания компьютерные вирусы делятся на:

- резидентные;
- нерезидентные.

Резидентные вирусы после их активизации полностью или частично перемещаются из среды обитания в оперативную память компьютера. Эти вирусы, используя, как правило, привилегированные режимы работы, разрешенные только операционной системе, заражают среду обитания и при выполнении определенных условий реализуют вредительскую функцию.

Нерезидентные вирусы попадают в оперативную память компьютера только на время их активности, в течение которого выполняют вредительскую функцию и функцию заражения. Затем они полностью покидают оперативную память, оставаясь в среде обитания.

По степени опасности для информационных ресурсов пользователя вирусы разделяются на:

- безвредные;
- опасные;
- очень опасные.

Безвредные вирусы создаются авторами, которые не ставят себе цели нанести какой-либо ущерб ресурсам компьютерной системы. Однако такие вирусы все-таки наносят определенный ущерб:

- расходуют ресурсы компьютерной системы;
- могут содержать ошибки, вызывающие опасные последствия для информационных ресурсов;
- вирусы, созданные ранее, могут приводить к нарушениям штатного алгоритма работы системы при модернизации операционной системы или аппаратных средств.

Опасные вирусы вызывают существенное снижение эффективности компьютерной системы, но не приводят к нарушению

целостности и конфиденциальности информации, хранящейся в запоминающих устройствах.

Очень опасные вирусы имеют следующие вредительские воздействия:

- вызывают нарушение конфиденциальности информации;
- уничтожают информацию;
- вызывают необратимую модификацию (в том числе и шифрование) информации;
- блокируют доступ к информации;
- приводят к отказу аппаратных средств;
- наносят ущерб здоровью пользователей.

По алгоритму функционирования вирусы подразделяются на:

- не изменяющие среду обитания при их распространении;
- изменяющие среду обитания при их распространении.

Для борьбы с компьютерными вирусами используются специальные антивирусные средства и методы их применения. Антивирусные средства выполняют следующие задачи:

- обнаружение вирусов в компьютерных системах;
- блокирование работы программ-вирусов;
- устранение последствий воздействия вирусов.

Обнаружение вирусов и блокирование работы программ-вирусов осуществляется следующими методами:

- сканирование;
- обнаружение изменений;
- эвристический анализ;
- использование резидентных сторожей;
- вакцинирование программ;
- аппаратно-программная защита.

Устранение последствий воздействия вирусов реализуется следующими методами:

- восстановление системы после воздействия известных вирусов;
- восстановление системы после воздействия неизвестных вирусов.

2.3.1. Профилактика заражения вирусами компьютерных систем

Главным условием безопасной работы в компьютерных системах является соблюдение правил, которые апробированы на практике и показали свою высокую эффективность.

Правило первое. Обязательное использование программных продуктов, полученных законным путем, так как в пиратских копиях вероятность наличия вирусов во много раз выше, чем в официально полученном программном обеспечении.

Правило второе. Дублирование информации, то есть создавать копии рабочих файлов на съемных носителях информации (дискеты, компакт-диски и другие) с защитой от записи.

Правило третье. Регулярно использовать антивирусные средства, то есть перед началом работы выполнять программы-сканеры и программы-ревизоры (Aidstest и Adinf). Эти антивирусные средства необходимо регулярно обновлять.

Правило четвертое. Проявлять особую осторожность при использовании новых съемных носителей информации и новых файлов. Новые дискеты и компакт-диски необходимо проверять на отсутствие загрузочных и файловых вирусов, а полученные файлы – на наличие файловых вирусов. Проверка осуществляется программами-сканерами и программами, осуществляющими эвристический анализ (Aidstest, Doctor Web, AntiVirus). При первом выполнении исполняемого файла используются резидентные сторожа. При работе с полученными документами и таблицами нужно запретить выполнение макрокоманд встроенными средствами текстовых и табличных редакторов (MS Word, MS Excel) до завершения полной проверки этих файлов на наличие вирусов.

Правило пятое. При работе в системах коллективного пользования необходимо новые сменные носители информации и вводимые в систему файлы проверять на специально выделенных для этой цели ЭВМ. Это должен выполнять администратор системы или лицо, отвечающее за безопасность информации. Только после всесторонней антивирусной проверки дисков и файлов они могут передаваться пользователям системы.

Правило шестое. Если не предполагается осуществлять запись информации на носитель, то необходимо заблокировать выполнение этой операции.

Постоянное выполнение изложенных правил позволяет значительно уменьшить вероятность заражения программными вирусами и обеспечить защиту пользователя от безвозвратных потерь информации.

В особо ответственных системах для борьбы с вирусами используются аппаратно-программные средства (например, Sheriff).

2.3.2. Порядок действий пользователя при обнаружении заражения вирусами компьютерной системы

Несмотря на строгое выполнение всех правил профилактики заражения вирусами компьютерной системы, нельзя полностью исключить возможность их заражения. Однако если придерживаться определенной последовательности действий при заражении вирусами, то последствия пребывания вирусов в компьютерной системе можно свести к минимуму.

О наличии вирусов можно судить по следующим событиям:

- появление сообщений антивирусных средств о заражении или о предполагаемом заражении;
- явные проявления присутствия вирусов (сообщения, выдаваемые на монитор или принтер, звуковые эффекты, уничтожение файлов и другие);
- неявные проявления заражения, которые могут быть вызваны сбоями или отказами аппаратных и программных средств, “зависаниями” системы, замедлением выполнения определенных действий, нарушением адресации, сбоями устройств и другими проявлениями.

При получении информации о предполагаемом заражении пользователь должен убедиться в этом. Решить такую задачу можно с помощью всего комплекса антивирусных средств. Если заражение действительно произошло, тогда пользователю следует выполнить следующие действия:

- выключить ЭВМ для уничтожения резидентных вирусов;
- осуществить загрузку эталонной операционной системы со сменного носителя информации, в которой отсутствуют вирусы;
- сохранить на сменных носителях информации важные файлы, которые не имеют резидентных копий;

- использовать антивирусные средства для удаления вирусов и восстановления файлов, областей памяти. Если работоспособность компьютерной системы восстановлена, то завершить восстановление информации всесторонней проверкой компьютерной системы с помощью всех имеющихся в распоряжении пользователя антивирусных средств;
- осуществить полное стирание и разметку (форматирование) несъемных внешних запоминающих устройств. В персональных компьютерах для этого могут быть использованы программы MS-DOS FDISK и FORMAT. Программа форматирования FORMAT не удаляет главную загрузочную запись на жестком диске, в которой может находиться загрузочный вирус. Поэтому необходимо выполнить программу FDISK с недокументированным параметром MBR, создать с помощью этой же программы разделы и логические диски на жестком диске. Затем выполняется программа FORMAT для всех логических дисков;
- восстановить операционную систему, другие программные системы и файлы с резервных копий, созданных до заражения;
- тщательно проверить файлы, сохраненные после обнаружения заражения, и, при необходимости, удалить вирусы и восстановить файлы;
- завершить восстановление информации всесторонней проверкой компьютерной системы с помощью всех имеющихся в распоряжении пользователя антивирусных средств.

2.3.3. Особенности защиты информации в базах данных

Базы данных рассматриваются как надежное хранилище структурированных данных, снабженное специальным механизмом для их эффективного использования в интересах пользователей (процессов). Таким механизмом является система управления базами данных (СУБД). Под системой управления базами данных понимаются программные или аппаратно-программные средства, реализующие функции управления данными, такие как: просмотр, сортировка, выборка, модификация, выполнение операций определения статистических характеристик и другие.

Базы данных размещаются:

- на компьютерной системе пользователя;
- на специально выделенной ЭВМ (сервере).

На компьютерной системе пользователя, как правило, размещаются личные или персональные базы данных, которые обслуживают процессы одного пользователя.

На серверах базы данных размещаются в локальных и корпоративных компьютерных сетях, которые используются, как правило, централизованно. Общедоступные глобальные компьютерные сети имеют распределенные базы данных. В таких сетях серверы размещаются на различных объектах сети. Серверы – это специализированные ЭВМ, приспособленные к хранению больших объемов данных и обеспечивающие сохранность и доступность информации, а также оперативность обработки поступающих запросов. В централизованных базах данных решаются проще проблемы защиты информации от преднамеренных угроз, поддержания актуальности и непротиворечивости данных. Достоинством распределенных баз данных является их высокая защищенность от стихийных бедствий, аварий, сбоев технических средств и диверсий, если осуществляется дублирование этих данных.

Особенности защиты информации в базах данных:

- необходимость учета функционирования СУБД при выборе механизмов защиты;
- разграничение доступа к информации реализуется не на уровне файлов, а на уровне частей баз данных.

При создании средств защиты информации в базах данных необходимо учитывать взаимодействие этих средств не только с операционной системой, но с СУБД. При этом возможно встраивание механизмов защиты в СУБД или использование их в виде отдельных составляющих. Для большинства СУБД придание им дополнительных функций возможно только на этапе их разработки. В эксплуатируемые системы управления базами данных дополнительные составляющие могут быть внесены путем расширения или модификации языка управления.

2.4. Законодательные акты РФ, регулирующие правовые отношения в сфере информационной безопасности и защиты государственной тайны

В государстве должна проводиться единая политика в области безопасности информационных технологий. Это требование нашло отражение в "Концепции национальной безопасности Российской Федерации", утвержденной Указом Президента РФ № 1300 от 17 декабря 1997 года. В этом документе отмечается, что в современных условиях всеобщей информатизации и развития информационных технологий резко возрастает значение обеспечения национальной безопасности РФ в информационной сфере. Значимость обеспечения безопасности государства в информационной сфере подчеркнута и в принятой в сентябре 2000 года "Доктрине информационной безопасности Российской Федерации". В этих документах определены важнейшие задачи государства в области информационной безопасности:

- установление необходимого баланса между потребностью в свободном обмене информацией и допустимыми ограничениями ее распространения;
- совершенствование информационной структуры, ускорение развития новых информационных технологий и их широкое внедрение, унификация средств поиска, сбора, хранения и анализа информации с учетом вхождения России в глобальную информационную инфраструктуру;
- разработка соответствующей нормативной правовой базы в интересах обеспечения информационной безопасности;
- координация деятельности органов государственной власти и других органов, решающих задачи обеспечения информационной безопасности;
- развитие отечественной индустрии телекоммуникационных и информационных средств, их приоритетное по сравнению с зарубежными аналогами распространение на внутреннем рынке;
- защита государственного информационного ресурса и, прежде всего, в федеральных органах власти и на предприятиях оборонного комплекса.

25 февраля 1995 года Государственной Думой принят Федеральный закон "Об информации, информатизации и защите

информации". В законе даны определения основных терминов: информация, информатизация, информационные системы, информационные ресурсы, конфиденциальная информация, собственник и владелец информационных ресурсов, пользователь информации. Государство гарантирует право владельца информации, независимо от форм собственности, распоряжаться ею в пределах, установленных законом. Владелец информации имеет право защищать свои информационные ресурсы, устанавливать режим доступа к ним. В этом законе определены цели и режимы защиты информации, а также порядок защиты прав субъектов в сфере информационных процессов и информатизации.

Другим важным правовым документом, регламентирующим вопросы защиты информации в КС, является закон РФ "О государственной тайне", принятый 21 июля 1993 года. Закон определяет уровни секретности государственной информации и соответствующую степень важности информации.

Отношения, связанные с созданием программ и баз данных, регулируются законом РФ от 23 сентября 1992 года "О правовой охране программ для ЭВМ и баз данных" и законом РФ от 09 июля 1993 года "Об авторском праве и смежных правах".

Важной составляющей правового регулирования в области информационных технологий является установление ответственности граждан за противоправные действия при работе с КС. Преступления, совершенные с использованием КС или причинившие ущерб владельцам КС, получили название компьютерных преступлений.

В Уголовном кодексе РФ, принятом 1 января 1997 года, есть глава № 28, в которой определена уголовная ответственность за преступления в области компьютерных технологий.

В статье 272 предусмотрены наказания за неправомерный доступ к компьютерной информации. Это правонарушение может наказываться от штрафа в размере 200 минимальных зарплат до лишения свободы на срок до 5 лет.

Статья 273 устанавливает ответственность за создание, использование и распространение вредоносных программ для ЭВМ. Это правонарушение может наказываться от штрафа до лишения свободы на срок до 7 лет.

В статье 274 определена ответственность за нарушение правил эксплуатации ЭВМ, системы ЭВМ или их сети. Если такое деяние

причинило существенный вред, то виновные наказываются лишением права занимать определенные должности или заниматься определенной деятельностью на срок до 5 лет. Если те же деяния повлекли тяжкие последствия, то предусмотрено лишение свободы на срок до 4 лет.

Вопросы для самоконтроля

1. Базовая система показателей качества информации.
2. Особенности информации, хранящейся, обрабатываемой и передаваемой в компьютерных системах.
3. Степени секретности государственной тайны.
4. Категории секретности коммерческой тайны.
5. Классы угроз безопасности информации.
6. Классы вредительских программ.
7. Основные правовые документы, регулирующие вопросы защиты информации в компьютерных системах.

Глава 3. Технические и программные средства реализации информационных процессов

Состав вычислительной системы называют конфигурацией. Конфигурация вычислительной системы включает аппаратные и программные средства, которые представляют собой отдельно аппаратную конфигурацию и программную конфигурацию.

3.1. Аппаратная конфигурация вычислительной системы

Современные компьютеры и вычислительные комплексы имеют блочно-модульную конструкцию – аппаратную конфигурацию, необходимую для исполнения конкретных видов работ, которую можно собирать из готовых блоков и узлов. По способу расположения устройств различают внутренние и внешние устройства. Внешними, как правило, являются большинство устройств ввода-вывода данных и некоторые устройства, предназначенные для длительного хранения данных.

Согласование между отдельными узлами и блоками выполняют с помощью аппаратных интерфейсов.

Аппаратными интерфейсами называют переходные аппаратно-логические устройства.

Стандарты на аппаратные интерфейсы называют протоколами.

Протокол – это совокупность технических условий, обеспечивающих взаимное согласование различных устройств при их совместной работе.

Многочисленные интерфейсы, присутствующие в любой вычислительной системе, можно условно разделить на последовательные и параллельные. Через последовательные интерфейсы данные передаются последовательно бит за битом, а через параллельные – одновременно группами битов. При этом количество битов, участвующих в одной посылке, определяется разрядностью интерфейса (8-, 16-, 24-, 32-, 64-разрядные).

Поскольку обмен данными через последовательные интерфейсы производится битами, их производительность измеряют битами в секунду (бит/с, Кбит/с, Мбит/с). Последовательные интерфейсы применяют для подключения "медленных" устройств, когда нет существенных ограничений на продолжительность обмена данными.

Так как обмен данными через параллельные интерфейсы производится группами битов (байтами), то их производительность измеряется байтами в секунду (байт/с, Кбайт/с, Мбайт/с). Параллельные интерфейсы применяют для подключения быстродействующих устройств там, где важна скорость передачи данных.

3.2. Базовая аппаратная конфигурация компьютера

Компьютер – это электронный прибор (универсальная техническая система), предназначенный для автоматизации создания, хранения, обработки и транспортировки данных.

Конфигурацию компьютера (состав оборудования) можно гибко изменять по мере необходимости. Однако существует понятие базовой конфигурации, которую считают типовой. Понятие базовой конфигурации по мере развития техники может меняться. **В настоящее время в состав базовой конфигурации включают: системный блок, монитор, клавиатуру и мышь.**

Системный блок является основным узлом, внутри которого установлены наиболее важные компоненты:

1. Материнская плата – основная плата персонального компьютера. На ней размещаются:
   - процессор – основная микросхема, выполняющая большинство математических и логических операций;
   - микропроцессорный комплект (чипсет) – набор микросхем, управляющих работой внутренних устройств компьютера и определяющих функциональные основные возможности материнской платы;
   - шины – наборы проводников, по которым происходит обмен сигналами между внутренними устройствами компьютера;
   - оперативная память (оперативное запоминающее устройство ОЗУ) – набор микросхем, предназначенных для временного хранения данных, когда компьютер включен;
   - постоянное запоминающее устройство (ПЗУ) – микросхема, предназначенная для длительного хранения данных и даже при выключенном компьютере;
   - разъемы для подключения дополнительных устройств (слоты).
2. Жесткий диск – основное устройство для долговременного хранения больших объемов данных и программ. Управление работой жесткого диска выполняет аппаратно-логическое устройство – контроллер жесткого диска. К основным параметрам жестких дисков относятся емкость и производительность. Емкость современных жестких дисков может достигать нескольких десятков Гбайт. Производительность диска оценивается скоростью внутренней передачи данных, которая может достигать 30-80 Мбайт/с. С производительностью диска, кроме скорости внутренней передачи данных, напрямую связан параметр среднего времени доступа. Он определяет интервал времени, необходимый для поиска нужных данных. Этот показатель может составлять 5–10 микросекунд (мкс), в зависимости от скорости вращения диска.
3. Дисковод гибких дисков – специальный накопитель для оперативного переноса небольших объемов информации на гибкие магнитные диски (дискеты) или с дискет на жесткий диск или в ОЗУ.
4. Дисковод компакт-дисков CD-RW (постоянное запоминающее устройство на основе компакт-диска) – устройство для считывания и записи больших объемов числовых данных с

помощью лазерного луча. Основным параметром дисководов CD-RW является скорость чтения данных. Она измеряется в кратных долях. За единицу измерения была принята скорость чтения 150 Кбайт/с. Двукратная скорость чтения 300 Кбайт/с, четырехкратная – 600 Кбайт/с и т. д.

5. Видеокарта (видеоадаптер) – это устройство, образующее совместно с монитором видеоподсистему компьютера. Видеоадаптер выполнен в виде отдельной дочерней платы, которая вставляется в один из слотов материнской платы и называется видеокартой. Видеоадаптер выполняет функции видеоконтроллера, видеопроцессора и видеопамяти. Одним из важнейших параметров видеосистемы является разрешение экрана. Для каждого размера монитора существует свое оптимальное разрешение экрана, которое должен обеспечивать видеоадаптер. Для монитора размером 15 дюймов оптимальное разрешение экрана составляет 880x600, для 17 дюймов – 1024x768, для 19 дюймов – 1280x1024 (1 дюйм равен 2,54 см). Цветовое разрешение (глубина цвета) определяет количество различных оттенков, которые может принимать отдельная точка экрана. Максимально возможное цветовое разрешение зависит от свойств видеоадаптера и, в первую очередь, от количества установленной на нем видеопамяти.
6. Звуковая карта – устройство, выполняющее вычислительные операции, связанные с обработкой звука, речи, музыки. Звуковая карта подключается к одному из слотов материнской платы в виде дочерней платы. Основным параметром звуковой карты является разрядность, определяющая количество битов, используемых при преобразовании сигналов из аналоговой формы в цифровую форму и наоборот.

Монитор – устройство визуального представления данных. Его основными потребительскими параметрами являются: размер и шаг маски экрана, максимальная частота регенерации изображения, класс защиты. Стандартные размеры мониторов 14, 15, 17, 19, 20 и 21 дюймов. Маска – это панель с регулярно расположенными отверстиями или щелями, которая расположена перед люминофором. Шаг маски – это расстояние между отверстиями или щелями. Чем меньше шаг маски, тем четче и точнее изображение. В современных мониторах шаг маски составляет 0,25–0,27 мм. Частота регенерации (обновления)

изображения показывает, сколько раз в течение секунды монитор может полностью сменить изображение. В настоящее время минимальная величина частоты регенерации составляет 75 Гц, нормальная – 85 Гц и комфортная – 100 и более Гц. Класс защиты монитора определяется стандартом, которому соответствует монитор с точки зрения требований техники безопасности. В настоящее время самые жесткие нормы по параметрам, определяющим качество изображения (яркость, контрастность, мерцание, антибликовые свойства покрытия), установлены в стандарте ТСО-99.

Клавиатура – клавишное устройство управления компьютером. Служит для ввода алфавитно-цифровых (знаковых) данных, а также команд управления. Комбинация монитора и клавиатуры обеспечивает простейший интерфейс (взаимодействие) пользователя. С помощью клавиатуры управляют компьютерной системой, а с помощью монитора получают от нее отклик.

Мышь – устройство управления манипуляторного типа. Перемещение мыши по плоской поверхности синхронизировано с перемещением графического объекта (указателя мыши) на экране монитора. Работу мыши обеспечивает специальная системная программа – драйвер мыши. Драйвер мыши предназначен для интерпретации сигналов, поступающих через порт, и обеспечивает механизм передачи информации о положении и состоянии мыши операционной системе и работающим программам. Комбинация монитора и мыши обеспечивает наиболее современный тип интерфейса пользователя.

Кроме базовой конфигурации компьютера в его состав могут входить и периферийные устройства. Периферийные устройства компьютера подключаются к его интерфейсам и предназначены для выполнения вспомогательных операций. Благодаря периферийным устройствам, компьютерная система приобретает гибкость и универсальность.

Классификация периферийных устройств по назначению.

1. Устройства ввода данных:
   - специальные клавиатуры;
   - специальные манипуляторы;
   - планшетные сканеры;
   - ручные сканеры;
   - барабанные сканеры;

- сканеры форм;
- штрих-сканеры;
- графические планшеты (дигитайзеры);
- цифровые фотокамеры.

2. Устройства вывода данных:
   - матричные принтеры;
   - лазерные принтеры;
   - светодиодные принтеры;
   - струйные принтеры.
3. Устройства хранения данных:
   - стримеры;
   - ZIP-накопители;
   - накопители HiFD;
   - накопители JAZ;
   - магнитооптические устройства.
4. Устройства обмена данными (модемы).

3.3. Программная конфигурация вычислительной системы

Программа – это упорядоченная последовательность команд. Конечная цель любой компьютерной программы – управление аппаратными средствами. Программное и аппаратное обеспечение в компьютере работает в неразрывной связи и в непрерывном взаимодействии. Состав программного обеспечения вычислительной системы называют программной конфигурацией. В программной конфигурации между ее программами существует взаимосвязь, то есть имеет место межпрограммный интерфейс. Возможность существования такого интерфейса основана на существовании технических условий и протоколов взаимодействия. На практике межпрограммный интерфейс (взаимодействие) обеспечивается путем распределения программного обеспечения по нескольким взаимодействующим между собой уровням. Эти уровни представляют собой пирамидальную конструкцию. Каждый следующий уровень опирается на программное обеспечение предшествующих уровней. Уровни программного обеспечения подразделяются на: базовый, системный, служебный и прикладной.

Базовый уровень – самый низкий уровень программного обеспечения – представляет базовое программное обеспечение. Оно

отвечает за взаимодействие с базовыми аппаратными средствами и, как правило, программные средства входят непосредственно в состав базового оборудования и хранятся в специальных микросхемах ПЗУ. Программы и данные записываются в микросхемы ПЗУ на этапе производства и не могут быть изменены в процессе эксплуатации.

Системный уровень – переходной. Программы, работающие на этом уровне, обеспечивают взаимодействие прочих программ компьютерной системы с программами базового уровня и непосредственно с аппаратным обеспечением, то есть выполняют "посреднические" функции. Конкретные программы, отвечающие за взаимодействие с конкретными устройствами, называются драйверами устройств. Они входят в состав программного обеспечения системного уровня. Программы, отвечающие за взаимодействие с пользователем, называют средствами обеспечения пользовательского интерфейса. **Совокупность программного обеспечения системного уровня образует ядро операционной системы компьютера.** Если компьютер оснащен программным обеспечением системного уровня, то он уже подготовлен к установке программ более высоких уровней, к взаимодействию программных средств с оборудованием и с пользователем. Наличие ядра операционной системы – непременное условие для возможности практической работы человека с вычислительной системой.

Служебный уровень – это служебные программы, обеспечивающие взаимодействие с программами базового и системного уровней. Служебные программы (утилиты) предназначены для автоматизации работ по проверке, наладке и настройке компьютерной системы.

Классификация служебных программ:

1. Диспетчеры файлов (файловые менеджеры), которые выполняют операции, связанные с обслуживанием файловой структуры: копирование, перемещение и переименование файлов, создание каталогов (папок), удаление файлов и каталогов, поиск файлов и навигация в файловой структуре.
2. Средства сжатия данных (архиваторы), которые предназначены для создания архивов. Архивирование данных упрощает их хранение, повышает эффективность использования носителя (устройства памяти) за счет того, что архивные файлы обычно

имеют повышенную плотность записи информации. Архиваторы часто используют для создания резервных копий ценных данных.

3. Средства просмотра и воспроизведения, предназначенные для просмотра и воспроизведения документов без загрузки их в "родительскую" прикладную систему.
4. Средства диагностики, предназначенные для автоматизации процессов диагностики аппаратного и программного обеспечения.
5. Средства контроля (мониторинга), предназначенные для того, чтобы следить за процессами, происходящими в компьютерной системе.
6. Мониторы установки, предназначенные следить за тем, чтобы не происходило нарушений работоспособности прочих программ при установке и удалении программного обеспечения.
7. Средства коммуникации (коммуникационные программы), предназначенные для установления соединений с удаленными компьютерами. Для обслуживания передачи сообщений электронной почты, обеспечения пересылки факсимильных сообщений и множества других операций в компьютерных сетях.
8. Средства обеспечения компьютерной безопасности – это средства пассивной и активной защиты данных от повреждения, несанкционированного доступа, просмотра и изменения данных.

Прикладной уровень – комплекс прикладных программ, с помощью которых на рабочем месте обеспечивается выполнение конкретных задач.

Классификация прикладных программ:

1. Текстовые редакторы, предназначенные для ввода и редактирования текстовых данных.
2. Текстовые процессоры, обеспечивающие ввод, редактирование текста и форматирование (оформление) документов, предназначенных для печати, а также электронных документов, предназначенных для отображения на экране.
3. Графические редакторы, предназначенные для создания и (или) обработки графических изображений.
4. Системы управления базами данных (СУБД), предназначенные для создания структуры базы данных, предоставления средств для заполнения этой структуры или импорта данных из таблиц других

баз данных, обеспечения возможности доступа к данным, а также предоставления средств поиска и фильтрации данных.

5. Электронные таблицы – это комплексные средства для хранения различных типов данных и их обработки.
6. Системы автоматизированного проектирования (CAD-системы), предназначенные для проектно-конструкторских работ.
7. Настольные издательские системы, предназначенные для автоматизации процесса верстки полиграфических изданий.
8. Экспертные системы, предназначенные для анализа данных, содержащихся в базах значений, и выдачи рекомендаций по запросу пользователя.
9. Редакторы HTML (*Web*-редакторы), предназначенные для создания и редактирования *Web*-документов (*Web*-страниц Интернета).
10. Браузеры – это программные средства, предназначенные для просмотра электронных документов, выполненных в формате HTML.
11. Интегрированные системы делопроизводства, предназначенные для автоматизации рабочего места руководителя (создания, редактирования и форматирования простейших документов, централизации функций электронной почты, факсимильной и телефонной связи, диспетчеризации и мониторинга документооборота предприятия, координации деятельности подразделений, оптимизации административно-хозяйственной деятельности и поставки по запросу оперативной и справочной информации).
12. Бухгалтерские системы – это специализированные системы, сочетающие в себе функции текстовых и табличных редакторов, электронных таблиц и систем управления базами данных.
13. Финансовые аналитические системы, предназначенные для банковских и биржевых структур.
14. Геоинформационные системы (ГИС), предназначенные для автоматизации картографических и геодезических работ на основе информации, полученной топографическими или аэрокосмическими методами.
15. Системы видеомонтажа, предназначенные для цифровой обработки видеоматериалов (монтажа, создания видеоэффектов, устранения дефектов, наложения звука, титров и субтитров).

16. Обучающие, развивающие, справочные и развлекательные системы и программы, представляющие отдельные категории прикладных программных средств и обладающие своими развитыми внутренними системами классификации.

3.4. Локальные и глобальные компьютерные сети

При соединении двух и более компьютеров образуется компьютерная сеть. Для создания компьютерных сетей необходимо специальное аппаратное обеспечение (сетевое оборудование) и программное обеспечение (сетевые программные средства).

Основной задачей, решаемой при создании компьютерных сетей, является обеспечение совместимости оборудования по электрическим и механическим характеристикам и совместимости информационного обеспечения (программ и данных) по системе кодирования и формату данных. Решение этой задачи относится к области стандартизации и основано на модели взаимодействия открытых систем – ***Model of Open System Interconnections (OSI)***. Она создана на основе технических предложений Международного института стандартов – ***International Standards Organization (ISO***). Согласно модели ***ISO/OSI*** архитектура компьютерных сетей включает семь уровней: *прикладной*, *уровень представления*, *сеансовый уровень*, *транспортный*, *сетевой*, *уровень соединения* и *физический*. Самый верхний уровень – ***прикладной***. На этом уровне пользователь взаимодействует с вычислительной системой. Самый нижний уровень – ***физический***. Он обеспечивает обмен сигналами между устройствами. Обмен данными в системах связи происходит путем их перемещения с верхнего уровня на нижний. Затем транспортировка и, наконец, обратное воспроизведение на компьютере клиента в результате перемещения с нижнего уровня на верхний уровень.

Для обеспечения необходимой совместимости на каждом из семи возможных уровней действуют специальные стандарты – ***протоколы***. Они определяют характер аппаратного взаимодействия компонентов сети (***аппаратные протоколы***) и взаимодействия программ данных (***программные протоколы***). Физически функции поддержки протоколов исполняют аппаратные устройства (***интерфейсы***) и программные средства (***программы поддержки протоколов***).

В соответствии с используемыми протоколами компьютерные сети подразделяются на локальные и глобальные.

В локальных компьютерных сетях преимущественно используется единый комплект протоколов для всех участников. По территориальному признаку локальные сети отличаются компактностью. Они могут объединять компьютеры одного помещения, этажа, здания, группы компактно расположенных сооружений. Создание локальных сетей характерно для отдельных предприятий или отдельных подразделений предприятий.

Глобальные сети имеют, как правило, увеличенные географические размеры. Они могут объединять как отдельные компьютеры, так и отдельные локальные сети, в том числе и использующие различные протоколы. Создание глобальных сетей характерно для предприятия или отрасли, занимающих обширную территорию. В этом случае в одну глобальную сеть могут объединяться отдельные локальные сети. Такие локальные сети связывают между собой с помощью традиционных каналов связи (кабельных, радиорелейных, спутниковых). Для связи между собой нескольких локальных сетей, работающих по разным протоколам, служат специальные средства, называемые шлюзами. При подключении локальной сети предприятия к глобальной сети важную роль играет сетевая безопасность. К числу мер безопасности относятся:

- ограничение доступа в локальную сеть извне для посторонних лиц;
- ограничение выхода за пределы локальной сети для сотрудников предприятия, не имеющих соответствующих прав;
- установление между локальной и глобальной сетями ***брандмауэров***. Брандмауэром может быть специальный компьютер или компьютерная программа, препятствующая несанкционированному перемещению данных между сетями.

Предназначение всех видов компьютерных сетей определяется двумя функциями:

- обеспечение совместного использования аппаратных и программных ресурсов;
- обеспечение совместного доступа к ресурсам данных.

Все участники локальной сети могут совместно использовать одно общее устройство печати (*сетевой принтер*), ресурсы жестких дисков одного выделенного компьютера (*файлового сервера*). Это же

относится и к программному, и к информационному обеспечению. Совокупность приемов разделения и ограничения прав участников локальной сети называется *политикой сети.* Управление сетевыми политиками называется *администрированием сети.* Лицо, управляющее организацией работы участников сети, называется системным *администратором.*

В модели ***ISO/OSI*** обмен данными между удаленными клиентами сети происходит:

- на прикладном уровне пользователь с помощью специальных приложений создает документ (сообщение, рисунок и тому подобное);
- на уровне представления операционная система его компьютера фиксирует, где находятся созданные данные (в оперативной памяти, в файле на жестком диске и тому подобное) и обеспечивает взаимодействие со следующим уровнем;
- на сеансовом уровне компьютер пользователя взаимодействует с локальной или глобальной сетью. Протоколы этого уровня проверяют права пользователя на выход в сеть и передают документ к протоколам транспортного уровня;
- на транспортном уровне документ преобразуется в ту форму, в которой положено передавать данные в используемой сети;
- сетевой уровень определяет маршрут движения данных в сети;
- уровень соединения необходим для того, чтобы промодулировать сигналы, циркулирующие на физическом уровне, в соответствии с данными, полученными с сетевого уровня;
- реальная передача данных происходит на физическом уровне. здесь нет ни документов, ни пакетов, ни даже байтов – только биты. Средства физического уровня находятся за пределами компьютера. В локальных сетях это оборудование самой сети. В глобальных сетях это линии телефонной связи, коммутационное оборудование телефонных станций и тому подобное.

На компьютере получателя информации происходит обратный процесс преобразования данных от битовых сигналов до документа.

Вопросы для самоконтроля

1. Какова связь между аппаратным и программным обеспечением?

2. Назовите четыре основных уровня программного обеспечения. Каков порядок их взаимодействия?
3. В чем преимущества и недостатки выполнения работ на компьютере аппаратными и программными средствами?
4. Какие категории программного обеспечения необходимы для автономного использования компьютера и для работы в компьютерной сети?
5. Какие основные категории программного обеспечения относятся к классу графических редакторов? В чем состоит принципиальная разница между этими категориями?
6. Что общего и в чем различие между понятиями ***программное обеспечение*** и ***информационное обеспечение*** средств вычислительной техники?
7. Понятие локальной и глобальной компьютерных сетей.
8. Уровни архитектуры компьютерных сетей.
9. Меры сетевой безопасности в компьютерных сетях.
10. Как происходит процесс обмена данными между удаленными клиентами в компьютерной сети?

Глава 4. Операционные системы персональных компьютеров

4.1. Общие сведения об операционных системах

Операционная система представляет комплекс системных и служебных программных средств. С одной стороны, она опирается на базовое программное обеспечение компьютера, входящее в его систему BIOS (базовая система ввода-вывода), с другой стороны, она сама является опорой для программного обеспечения более высоких уровней – прикладных и большинства служебных приложений. Приложениями операционной системы принято называть программы, предназначенные для работы под управлением данной системы.

Основная функция всех операционных систем – посредническая. Она заключается в обеспечении нескольких видов интерфейса:

- интерфейса между пользователем и программно-аппаратными средствами компьютера (интерфейс пользователя);
- интерфейса между программным и аппаратным обеспечением (аппаратно-программный интерфейс);

- интерфейса между разными видами программного обеспечения (программный интерфейс).

Все операционные системы способны обеспечивать как пакетный, так и диалоговый режим работы с пользователем.

В пакетном режиме операционная система автоматически исполняет заданную последовательность команд.

В диалоговом режиме операционная система находится в состоянии ожидания команды пользователя и, получив ее, приступает к исполнению, а исполнив, возвращает отклик и ждет очередной команды.

По реализации интерфейса пользователя различают неграфические и графические операционные системы.

Неграфические операционные системы реализуют интерфейс командной строки. Основным устройством управления в данном случае является клавиатура. Именно эти системы и обеспечивают диалоговый режим работы.

Графические операционные системы реализуют более сложный тип интерфейса, в котором в качестве органа управления кроме клавиатуры используется мышь или адекватное устройство позиционирования. Работа с графической операционной системой основана на взаимодействии активных и пассивных экранных элементов управления. В качестве активного элемента управления выступает указатель мыши – графический объект, перемещение которого на экране синхронизировано с перемещением мыши. В качестве пассивных элементов управления выступают графические элементы управления приложений (экранные кнопки, значки, переключатели, флажки, раскрывающиеся списки, строки меню и многие другие).

Все операционные системы обеспечивают свой автоматический запуск.

4.2. Файловая структура операционных систем

Все современные дисковые операционные системы обеспечивают создание файловой структуры, предназначенной для хранения данных на дисках и обеспечения доступа к ним. Принцип организации файловой структуры – табличный. Поверхность жесткого диска рассматривается как трехмерная матрица, измерениями которой

являются номера поверхности, цилиндра и сектора. Под цилиндром понимается совокупность всех дорожек, принадлежащих разным поверхностям и находящихся на равном удалении от оси вращения. Данные о том, в каком месте диска записан тот или иной файл, хранятся в системной области диска в специальных таблицах размещения файлов (FAT-таблицах). К FAT-таблице предъявляются очень высокие требования по ее надежности, поскольку нарушение FAT-таблицы приводит к нарушению доступа к данным, записанным на диске. Поэтому FAT-таблица создается в двух экземплярах, идентичность которых регулярно контролируется средствами операционной системы.

Наименьшей физической единицей хранения данных является сектор. Емкость сектора составляет 512 Кбайт. Поскольку емкость FAT-таблицы ограничена, то для дисков, емкость которых превышает 32 Мбайта, обеспечить адресацию к каждому отдельному сектору невозможно. С целью устранения этого недостатка секторы условно объединяются в кластеры. Кластер – это наименьшая единица адресации к данным. Емкость кластера не фиксирована и зависит от емкости диска.

Несмотря на то, что сведения о местоположении файлов хранятся в табличной структуре, пользователю они представляются в виде иерархической структуры, а все необходимые преобразования берет на себя операционная система.

Под управлением операционной системы осуществляются следующие функции обслуживания файловой структуры:

- создание файлов и присвоение им имен;
- создание каталогов (папок) и присвоение им имен;
- копирование и перемещение файлов между дисками и между каталогами (папками) одного диска;
- удаление файлов и каталогов (папок);
- навигация по файловой структуре с целью доступа к заданному файлу, каталогу (папке);
- управление атрибутами файлов.

Файл – это именованная последовательность байтов произвольной длины. Поскольку файл может иметь нулевую длину, то создание файла состоит в присвоении ему имени и регистрации его в файловой структуре – это одна из функций операционной системы. По способам именования файлов различают “короткое” и “длинное” имя.

Короткое имя файла состоит из двух частей: собственно имени (длина имени от 1 до 8 символов) и расширения имени (длина 3 символа). Имя от расширения отделяется точкой. Как имя, так и расширение могут состоять только из алфавитно-цифровых символов латинского (английского) алфавита. Основным недостатком таких имен является их низкая содержательность, так как несколькими символами не всегда удается выразить характеристику файла. С появлением операционных систем серии Windows (начиная с Windows 95) стало возможным создавать длинные имена файлов.

Длинное имя может содержать до 256 символов. Этого вполне достаточно для создания содержательных имен файлов. Длинное имя может содержать любые символы, кроме девяти специальных: \ / : * ? “ < > |. В имени разрешается использовать пробелы и несколько точек. Расширением имени считаются все символы, идущие после последней точки.

В современных операционных системах Windows использование длинных имен файлов имеет ряд особенностей:

- в корневой папке диска (на верхнем уровне иерархической файловой структуры) нежелательно хранить файлы с длинными именами, так как в этой папке ограничено количество единиц хранения, поэтому, чем длиннее имена, тем меньше файлов можно разместить в корневой папке;
- существует жесткое ограничение на длину полного имени файла (в него кроме собственного имени файла входит путь доступа к файлу, начиная от вершины иерархической структуры). Полное имя не может быть длиннее 260 символов;
- разрешается использовать символы любых алфавитов, в том числе и русского;
- прописные и строчные буквы не различаются операционной системой;
- во многих случаях выбор расширения имени файла не является частным делом пользователя. Приложения операционных систем предлагают выбрать только основную часть имени и указать тип файла, а соответствующее расширение имени создается автоматически.

Кроме имени файла операционная система хранит для каждого файла дату его создания (изменения) и его атрибуты – это дополнительные параметры, определяющие свойства файлов.

Операционная система позволяет их контролировать и изменять. Состояние атрибутов учитывается при проведении автоматических операций с файлами. К основным атрибутам относятся следующие четыре:

- только для чтения (Read only), то есть файл не предназначен для внесения изменений;
- скрытый (Hidden), то есть файл не отображается на экране при проведении файловых операций. Это мера защиты против случайного повреждения файла;
- системный (System) – это файлы, обладающие важными функциями в работе самой операционной системы;
- архивный (Archive) в прошлом использовался для работы программ резервного копирования. В современных операционных системах утратил практическое значение, так как используются другие средства для резервного копирования.

4.3. Базовые функции операционных систем

Основная функция операционной системы состоит в обеспечении интерфейса приложений с аппаратными и программными средствами вычислительной системы, а также с пользователем. С точки зрения управления исполнением приложений, различают однозадачные и многозадачные операционные системы.

Однозадачные операционные системы передают все ресурсы вычислительной системы одному исполняемому приложению и не допускают ни параллельного выполнения другого приложения (полная многозадачность), ни его приостановки и запуска другого приложения (вытесняющая многозадачность). В то же время параллельно с однозадачными операционными системами возможна работа специальных программ называемых резидентами. Такие программы не опираются на операционную систему, а непосредственно работают с процессором, используя его систему прерываний.

Большинство современных графических операционных систем являются многозадачными. Они управляют распределением ресурсов вычислительной системы между задачами и обеспечивают:

- возможность одновременной или поочередной работы нескольких приложений;

- возможность обмена данными между несколькими приложениями;
- возможность совместного использования программных, аппаратных, сетевых и прочих ресурсов вычислительной системы несколькими приложениями.

От того, как операционная система управляет работой приложений, во многом зависит надежность всей вычислительной системы. Операционная система должна предоставлять возможность прерывания работы приложений по желанию пользователя и снятия сбойной задачи без ущерба для работы других приложений. При этом требование надежности может входить в противоречие с требованием ее универсальности.

Для правильной работы приложений на компьютере они должны пройти операцию, называемую установкой. При этом осуществляется привязка приложения к существующей программно-аппаратной среде компьютера и его настройка на работу именно в этой среде.

Современные графические операционные системы берут на себя управление установкой приложений. Они управляют распределением ресурсов вычислительной системы между приложениями, обеспечивают доступ устанавливаемых приложений к драйверам устройств вычислительной системы, формируют общие ресурсы, которые могут использоваться разными приложениями, выполняют регистрацию установленных приложений и выделенных им ресурсов.

Средства аппаратного обеспечения вычислительной техники отличаются широким многообразием. Существуют сотни различных моделей видеоадаптеров, звуковых карт, мониторов, принтеров, сканеров и прочего оборудования. При таком многообразии технических устройств ни один разработчик программного обеспечения не в состоянии предусмотреть все варианты взаимодействия своей программы с соответствующим устройством. Гибкость аппаратных и программных конфигураций вычислительных систем поддерживается за счет того, что каждый разработчик оборудования прикладывает к нему специальные программные средства управления – драйверы. Драйверы имеют точки входа для взаимодействия с прикладными программами, а диспетчеризация обращений прикладных программ к драйверам устройств – это одна из функций операционной системы. Современные операционные системы позволяют управлять не только

установкой и регистрацией программных драйверов устройств, но и процессом аппаратно-логического подключения.

4.4. Прочие функции операционных систем

Кроме базовых функций операционные системы могут предоставлять различные дополнительные функции. Прочие функции операционных систем могут включать следующие:

- поддерживать функционирование локальной компьютерной сети без специального программного обеспечения;
- обеспечивать доступ к основным службам Интернета средствами, интегрированными в состав операционной системы;
- создавать системными средствами сервера Интернета, его обслуживание и управление, в том числе дистанционное посредством удаленного соединения;
- средства защиты данных от несанкционированного доступа, просмотра и внесения изменений;
- обеспечение комфортной поочередной работы различных пользователей на одном персональном компьютере с сохранением персональных настроек рабочей среды каждого из них;
- автоматическое исполнение операций обслуживания компьютера и операционной системы по заданному расписанию или под управлением удаленного сервера;
- обеспечивать работу с компьютером лицам, имеющим физические недостатки, связанные с органами зрения, слуха и другими.

Современные операционные системы могут также включать минимальный набор прикладного программного обеспечения, которое можно использовать для исполнения простейших практических задач:

- чтение, редактирование и печать текстовых документов;
- создание и редактирование простейших рисунков;
- выполнение арифметических и математических расчетов;
- ведение дневников и служебных блокнотов;
- создание, передача и прием сообщений электронной почты;
- создание и редактирование факсимильных сообщений;

- воспроизведение и редактирование звукозаписи;
- воспроизведение видеозаписи;
- разработка и воспроизведение комплексных электронных документов, включающих текст, графику, звукозапись и видеозапись.

В дальнейшем по мере развития аппаратных средств неизбежно будут развиваться и функции операционных систем.

Вопросы для самоконтроля

1. Что такое операционная система? Каково ее предназначение?
2. Какие основные функции выполняет операционная система?
3. Какие режимы работы с компьютером способны обеспечивать операционные системы и в чем их сущность?
4. Какие операционные системы различают в зависимости от реализации интерфейса пользователя и что они могут реализовывать?
5. Как организуется хранение файлов на дисках компьютера?
6. Какие функции выполняет операционная система по обслуживанию файловой структуры?
7. Как формируются короткое и длинное имя файла?

Глава 5. Электронные таблицы Microsoft Excel

5.1. Назначение и возможности электронных таблиц

Программа **MS Excel** (электронные таблицы) предназначена для работы с таблицами данных, преимущественно числовых.

Применение электронных таблиц упрощает работу с данными и позволяет получать результаты без проведения расчетов вручную, а также в решении многих задач не применять специального программирования.

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.

Возможности электронных таблиц:

- проведение однотипных расчетов над большими наборами данных;
- автоматизация итоговых вычислений;
- решение задач путем подбора значений параметров;
- табулирование формул (функций);
- обработка результатов экспериментов;
- проведение поиска оптимальных значений параметров;
- подготовка табличных документов;
- построение диаграмм и графиков по имеющимся данным.

Загрузку программы **MS Excel** можно выполнить следующими способами:

- двойным щелчком по ярлыку **Microsoft Excel** на рабочем столе, если ярлык там находится;
- выполнением последовательности команд: **Пуск, Программы, Стандартные,** ярлык **Microsoft Excel**;
- выполнением последовательности команд: **Пуск, Найти, Файлы и папки**. В появившемся диалоговом окне в строке **Имя** ввести **Microsoft Excel** (имя файла ярлыка программы MS Excel) и щелкнуть по кнопке **Найти**. После окончания поиска выполнить двойной щелчок по ярлыку **Microsoft Excel**. По завершению загрузки программы MS Excel закрыть окно поиска.

Загрузка программы MS Excel заканчивается появлением на экране окна программы с открытым рабочим листом с именем "**Лист1**" стандартной рабочей книги с именем "**Книга1**".

При создании своей рабочей книги необходимо выполнить следующие действия:

- щелчком левой кнопки мыши развернуть меню **Сервис**, щелкнуть левой кнопкой мыши по строке **Параметры** и в появившемся диалоговом окне щелкнуть мышью по закладке **Общие**. В окошечке **Листов в новой книге** установить требуемое число листов и щелкнуть по кнопке **Ok;**
- на панели инструментов щелкнуть по кнопке **Создать;**
- щелчком левой кнопки мыши развернуть меню **Файл** и щелкнуть мышью по строке **Сохранить как....** В появившемся окне

щелкнуть по стрелке окна **Мои документы**. В раскрывшемся меню щелкнуть по строке с адресом вашего каталога, а затем в нижнем окне **Имя файла** вместо стандартного имени записать свое название рабочей книги, после чего щелкнуть по кнопке **Сохранить**. В последующем при работе с этим файлом такие действия не выполнять, если не требуется замена имени файла, а достаточно периодически щелкать по кнопке **Сохранить** на панели инструментов.

5.2. Рабочее окно MS Excel

Основными элементами рабочего окна являются:

- строка заголовка (в ней указывается имя программы) с кнопками управления окном программы и окном документа (**Свернуть, Свернуть в окно** или развернуть во весь экран, **Закрыть**);
- строка основного меню (каждый пункт меню объединяет набор команд, объединяющих общую функциональную направленность) плюс окно для запроса справки и кнопка **Закрыть окно**;
- панели инструментов (**Стандартная** и **Форматирование**);
- поле **Имя**, вставка функции (f_x) и **Строка формул**. Строка формул предназначена для ввода и редактирования значений или формул в ячейках. В поле **Имя** отображается адрес текущей ячейки;
- рабочая область (активный рабочий лист);
- полоса прокрутки;
- строка перебора рабочих листов;
- строка состояния.

5.3. Структура электронных таблиц

Документом MS Excel является **рабочая книга**. Рабочих книг создать можно столько, сколько позволит наличие свободной памяти на соответствующем устройстве памяти. Открыть рабочих книг можно столько, сколько их создано. Однако активной рабочей книгой может быть только одна текущая (открытая) книга.

Рабочая книга представляет собой набор рабочих листов, каждый из которых имеет табличную структуру. В окне документа отображается только текущий (активный) рабочий лист, с которым и ведется работа. Каждый рабочий лист имеет название, которое отображается на ярлычке листа в нижней части окна. С помощью ярлычков можно переключаться к другим рабочим листам, входящим в ту же рабочую книгу. Чтобы переименовать рабочий лист, надо дважды щелкнуть мышью на его ярлычке и заменить старое имя на новое или путем выполнения следующих команд: меню **Формат**, строка **Лист** в списке меню, **Переименовать**. А также установить указатель мыши на ярлык активного рабочего листа и щелкнуть правой кнопкой мыши, после чего в появившемся контекстном меню щелкнуть по строке **Переименовать** и выполнить переименование. В рабочую книгу можно добавлять (вставлять) новые листы или удалять ненужные. Вставку листа можно осуществить путем выполнения команд: меню **Вставка**, строка **Лист** в списке меню. Вставка листа произойдет перед активным листом. Выполнение выше изложенных действий можно осуществить с помощью контекстного меню, которое активизируется нажатием правой кнопки мыши, указатель которой должен быть установлен на ярлычке соответствующего листа. Чтобы поменять местами рабочие листы, нужно указатель мыши установить на ярлычок перемещаемого листа, нажать левую кнопку мыши и перетащить в нужное место.

Рабочий лист состоит из строк и столбцов. Столбцы озаглавлены прописными латинскими буквами и, далее, двухбуквенными комбинациями. Всего рабочий лист содержит 256 столбцов, пронумерованных от **A** до **IV**. Строки последовательно нумеруются цифрами, от 1 до 65536.

На пересечении столбцов и строк образуются **ячейки** таблицы. Они являются минимальными элементами для хранения данных. Каждая ячейка имеет свой адрес. **Адрес** ячейки состоит из имени столбца и номера строки, на пересечении которых расположена ячейка, например, **A1**, **B5**, **DE324**. Адреса ячеек используются при записи формул, определяющих взаимосвязь между значениями, расположенными в разных ячейках. В текущий момент времени активной может быть только одна ячейка, которая активизируется щелчком мыши по ней и выделяется рамкой. Эта рамка в программе Excel играет роль курсора. Операции ввода и редактирования данных всегда производятся только в активной ячейке.

На данные, расположенные в соседних ячейках, образующих прямоугольную область, можно ссылаться в формулах как на единое целое. Группу ячеек, ограниченную прямоугольной областью, называют **диапазоном**. Наиболее часто используются прямоугольные диапазоны, образующиеся на пересечении группы последовательно идущих строк и группы последовательно идущих столбцов. Диапазон ячеек обозначают, указывая через двоеточие адрес первой ячейки и адрес последней ячейки диапазона, например, **B5:F15**. Выделение диапазона ячеек можно сделать протягиванием указателя мыши от одной угловой ячейки до противоположной ячейки по диагонали. Рамка текущей (активной) ячейки при этом расширяется, охватывая весь выбранный диапазон.

5.4. Способы адресации ячеек

В MS Excel имеются три способа адресации ячеек: относительная, как показано выше (A7), абсолютная и смешанная. Признаком абсолютной адресации является знак **$**.

Если знак **$** предшествует имени столбца и номеру строки **C12, A2:d24**, то будет абсолютный адрес ячейки или диапазона ячеек. Абсолютная адресация применяется в случаях, когда в формулах необходимо осуществлять ссылку на одну и ту же ячейку (один и тот же диапазон ячеек).

Если знак **$** предшествует имени столбца **$B7**, то будет абсолютный адрес столбца. Если знак **$** предшествует номеру строки **D$23**, то будет абсолютный адрес строки. Это примеры смешанной адресации.

Для изменения способа адресации, при редактировании формулы, нужно выделить ссылку на ячейку и нажать клавишу **F4.** При одном нажатии будет абсолютный адрес ячейки. При двух нажатиях будет абсолютный адрес строки. При трех нажатиях будет абсолютный адрес столбца. При четырех нажатиях будет относительный адрес ячейки.

Отдельная ячейка может содержать данные, относящиеся к одному из следующих типов: число, дата, текст или формула, а также оставаться пустой.

5.5. Ввод и редактирование данных

Ввод данных осуществляется непосредственно в текущую ячейку или в строку формул, располагающуюся в верхней части окна программы непосредственно под панелями инструментов. Вводимые данные в любом случае отображаются как в ячейке, так и в строке формул.

Ввод формулы или функции всегда начинается с символа "=" (знака равенства).

Чтобы завершить ввод, сохранив введенные данные, используется клавиша **Enter**. Чтобы отменить внесенные изменения и восстановить прежнее значение ячейки, используется кнопка **Отмена** на панели инструментов или клавиша **ESC**. Для очистки текущей ячейки или выделенного диапазона проще всего использовать клавишу **DELETE**.

Чтобы изменить формат отображения данных в текущей ячейке или выбранном диапазоне, необходимо раскрыть меню **Формат** и щелкнуть мышью по строке **Ячейки**. Появляется соответствующее диалоговое окно. Вкладки этого диалогового окна позволяют:

- выбирать нужный вид данных;
- выбирать формат записи данных (количество знаков после запятой, 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.

5.6. Конструирование формул. Управление вычислениями

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 ней отображается текущий результат вычисления этой формулы. Если сделать ячейку текущей (активной), то формула отобразится в строке формул.

Для редактирования формулы следует дважды щелкнуть на соответствующей ячейке. При этом ячейки (диапазоны ячеек), от

которых зависит значение формулы, выделяются цветными рамками, а сами ссылки отображаются в ячейке и в строке формул тем же цветом. Редактирование формулы (функции) можно осуществлять и в строке формул. Для этого нужно сделать активной ячейку с формулой и указатель мыши установить в нужном месте формулы.

Все диалоговые окна программы **Excel**, которые требуют указания адресов ячеек, содержат кнопки, присоединенные к соответствующим полям. При щелчке по такой кнопке диалоговое окно сворачивается до минимально возможного размера, что облегчает выбор нужной ячейки (диапазона ячеек) с помощью щелчка или протягивания.

5.7. Функции рабочего листа

Для ускорения и облегчения вычислительной работы Excel предоставляет в распоряжение пользователя мощный аппарат функций рабочего листа, позволяющих осуществлять все возможные расчеты.

В целом Microsoft Excel содержит более 400 функций рабочего листа (встроенных функций). Все они в соответствии с предназначением делятся на 9 групп:

- финансовые функции;
- функции даты и времени;
- математические функции;
- статистические функции;
- функции ссылки;
- функции работы с базой данных;
- текстовые функции;
- логические функции;
- функции проверки свойств и значений.

Запись любой функции в ячейку рабочего листа обязательно начинается с символа равно (=). Если функция используется в составе какой-либо сложной функции или в формуле, то символ равно (=) пишется перед этой функцией (формулой). Обращение к любой функции производится указанием ее имени и следующего за ним в круглых скобках аргумента (параметра) или списка параметров. Наличие круглых скобок обязательно, именно они служат признаком того, что используемое имя является именем функции. Параметры списка разделяются точкой с запятой (;). Их количество не должно превышать 30, а длина формулы, содержащей сколько угодно

обращений к функциям, не должна превышать 1024 символов. Все имена при записи (вводе) формулы рекомендуется набирать строчными буквами, тогда правильно введенные имена будут отображены прописными буквами.

MS Excel обладает обширной справочной системой, поэтому нет необходимости приводить полное описание функций.

Все функции или почти все могут быть применены двумя способами:

- запись функции непосредственно в ячейку рабочего листа. При этом значения аргументов (параметров) функции могут вводиться в виде конкретных чисел, если параметр имеет одно значение, или в виде адресов ячеек, в которых предварительно записаны значения этих параметров. Если параметр имеет несколько значений, то он записывается в виде диапазона ячеек;
- использование мастера функций (f_x на строке формул). Для этого надо щелкнуть мышью по кнопке на строке формул. В появившемся диалоговом окне **Мастер функций – шаг 1 из 2** выбрать нужную категорию функций в окне **Категория:**, а затем выбрать нужную функцию в окне **Выберите функцию:** и щелкнуть мышью по кнопке **Ok**. Дальше действовать согласно предписанию. Данный пункт изложен применительно к MS Excel 2003. Аналогичные действия применительно к MS Excel 2000 имеют следующую редакцию:
- использование готовой формы для вычисления функции. Для этого надо щелкнуть мышью по кнопке **Изменить формулу** на строке формул. На кнопке имеется символ =. Затем щелкнуть мышью по стрелке (маленький черный треугольник вершиной вниз) справа окна **Имя**. В результате этого действия раскроется список с именами 10 функций, использовавшихся ранее, и строка **Другие функции...** Если нужная функция есть в списке, то надо встать на строку с именем этой функции и щелкнуть мышью. Если нужной функции нет в списке, то надо встать на строку **Другие функции ...** и щелкнуть мышью. В появившейся форме **Мастер функций – шаг 1 из 2** выбрать нужную категорию функций в окне **Категория:**, а затем выбрать нужную функцию в окне **Функция:** и щелкнуть мышью по кнопке **Ok**. Дальше действовать согласно предписанию.

Возможные ошибки при применении функций (формул):

- **#ИМЯ?** – неправильно введено имя функции или адрес ячейки;
- **#ДЕЛ/0!** – значение знаменателя в формуле равно нулю;
- **#ЧИСЛО!** – значение аргумента функции не соответствует допустимой величине, например, Ln(0), Ln(-x), $\sqrt{-x}$;
- **#ЗНАЧ!** – параметры функции введены неправильно, например, вместо диапазона ячеек введено их последовательное перечисление;
- **#ССЫЛКА!** – неверная ссылка на адрес ячейки (диапазон ячеек);
- ########## – ширина ячейки недостаточна для изображения полученного числа.

Вопросы для самоконтроля

1. Какие основные возможности можно реализовать с помощь электронных таблиц MS Excel?
2. Как загрузить программу MS Excel?
3. Как создать свою рабочую книгу?
4. Структура рабочего листа рабочей книги MS Excel.
5. Как можно записать абсолютный и смешанный адрес ячейки рабочего листа MS Excel?
6. Как можно осуществить выбор и форматирование данных требуемого типа в ячейке или интервале ячеек?
7. Как можно записать и отредактировать формулу?
8. Способы применения встроенных (стандартных) функций MS Excel.

Глава 6. Текстовый процессор Microsoft Word

Текстовые документы, создаваемые с помощь персонального компьютера, условно можно разделить на две группы – простые и комплексные. Простые документы представляют собой форматированный текст. Комплексные документы кроме текста содержат объекты иной природы (чертежи, рисунки, формулы, таблицы, объекты мультимедиа и прочие). Эти документы могут создаваться с помощью специальных программных средств, которые называют текстовыми процессорами. В настоящее время в нашей стране широко применяются текстовые процессоры MS Word (Word 9.0) из пакета Microsoft Office 2000 и MS Word из пакета Microsoft

Office 2003. В перспективе по мере развития и совершенствования операционной системы неизбежно появятся и новые версии текстовых процессоров.

6.1. Рабочее окно процессора MS Word

Основными элементами рабочего окна являются:

- строка меню;
- панели инструментов;
- рабочее поле;
- линейка;
- кнопки управления режимами представления документа;
- строка состояния, включающая индикаторы.

Строка меню как элемент управления обеспечивает доступ ко всем функциональным возможностям программы MS Word и удовлетворяет принципу функциональной полноты. Меню, открывающееся из соответствующей строки меню, обладает свойством функциональной автонастройки. Пункты строки меню открываются в два приема. На первом этапе открывается сокращенное меню, а затем открывается расширенное меню наведением указателя мыши на пункт раскрытия или через некоторое время автоматически.

Панели инструментов являются настраиваемыми и обладают контекстной чувствительностью, то есть при выделении в поле документа какого-либо объекта, автоматически открывается панель инструментов, предназначенная для его редактирования. Настройку выполняет пользователь путем подключения необходимых функциональных панелей через меню **Вид**. В раскрывшемся меню выбирается команда **Панели инструментов**. В результате выполнения этой команды раскрывается меню всех имеющихся функциональных панелей инструментов. В состав функциональных панелей инструментов входят следующие панели:

- **стандартная панель**, содержащая элементы управления файловыми операциями, редактирования документа, экранного отображения документа. Устанавливается по умолчанию;
- **панель форматирования**, содержащая инструменты для форматирования документа. Устанавливается по умолчанию;

- **панель *Word-Art***, содержащая элементы управления для создания художественных заголовков;
- **панель *Visual Basic***, обеспечивающая 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;
- **панель Автотекст**, содержащая средства быстрого доступа к настройке функции автотекста. Одновременно предоставляет быстрый доступ к средствам настройки функций автозамены и автоформата;
- **панель Базы данных**, содержащая элементы управления для работы с базами данных (сортировка, поиск, управление структурой таблиц и прочее). В качестве базы данных могут выступать как собственные таблицы Word, так и таблицы Access;
- **панель Веб-компоненты**, содержащая комплект готовых компонентов для создания элементов управления *Web*-страницы или электронной формы. Применяется для создания обратной связи с потребителем документа (опросные листы, анкеты, бланки заказов, заявок и прочее);
- **панель Веб-узел**, содержащая элементы управления для навигации в *Web*-структурах данных;
- **панель Настройка изображения**, содержащая элементы управления для основных функций настройки растровых изображений. Позволяет настраивать яркость, контрастность, размер, рамку, режимы обтекания текстом и прочие параметры выделенного растрового объекта;
- **панель Рамки**, содержащая элементы управления для создания фреймов. Процессор поддерживает два типа фреймов (фреймы печатных документов и фреймы электронных документов). Фреймы в печатных документах представляют особые области печатной страницы для вывода специальной информации, например, колонтитулов. Фреймы в электронных документах представляют собой особые прямоугольные области, предназначенные для вывода нескольких *Web*-документов в рамках одной *Web*-страницы;

- **панель Рецензирование**, содержащая 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 и принять их или отвергнуть;
- **панель Рисование**, содержащая элементы управления и инструменты для выполнения простейших чертежно-графических работ. Графические объекты, создаваемые инструментами данной панели, имеют характер векторных объектов;
- **панель Таблицы и границы**, содержащая элементы управления для создания таблиц и оформления текстовых блоков рамками. Дополнительно предоставляет средства для сортировки данных и проведения итоговых расчетов в таблицах (функция Автосумма);
- **панель Формы**, содержащая элементы управления для разработки стандартных форм. Процессор позволяет создавать три типа форм: *Web*-формы (объекты *Web*-страниц), формы Word (электронный документ) и печатные формы;
- **панель Элементы управления**, содержащая набор готовых компонентов ActiveX для создания элементов управления *Web*-страниц и *Web*-форм.

Набор панелей инструментов, их состав и возможности дают полную характеристику функций процессора Word в целом.

Остальные элементы рабочего окна не требуют, на наш взгляд, особых пояснений.

6.2. Принципы работы с процессором MS Word

Современные текстовые процессоры позволяют создавать документы трех типов.

Первый тип – это печатные документы, которые создаются и распечатываются на одном рабочем месте или в одной рабочей группе. Состав допустимых средств оформления в данном случае определяется только техническими возможностями печатающего устройства.

Второй тип – это электронные документы в формате текстового процессора, которые передаются заказчику в виде файлов. Электронный документ не всегда может быть окончательным. В ряде случаев заказчик может его дорабатывать, редактировать,

форматировать, распечатывать или использовать для подготовки своих документов. Состав допустимых средств оформления и форматирования в данном случае определяется заказчиком и, как правило, минимален.

Третий тип – это *Web*-документы. В таких документах большую роль играет управление цветом и для них наиболее широк выбор средств форматирования и оформления.

Типы документов в полной мере определяют принципы практической работы с текстовым процессором.

Базовый принцип состоит в том, что чем больше возможностей имеет программа MS Word, тем строже надо подходить к выбору тех функций (средств оформления и форматирования), которыми можно пользоваться в каждом конкретном случае при создании документа.

Если заранее четко не определены требования к документу по его оформлению и форматированию, то следует руководствоваться следующими принципами:

- 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 прочее) – 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.

Если игнорировать выше изложенные принципы при создании подобных документов, то при обработке данных, содержащихся в документе, другими программными средствами полученные преимущества форматирования и оформления могут оборотиться тяжкими последствиями.

6.3. Основные режимы представления документов

Указанные выше версии процессоров поддерживают следующие основные режимы представления документов.

1. **Обычный режим**. В этом режиме представляется только содержательная часть документа без реквизитных элементов оформления, относящихся не к тексту, а к печатным страницам (колонтитулы, колонцифры, подстраничные сноски и другие). Этот режим удобен на ранних этапах разработки документа (ввод текста, редактирование, рецензирование).
2. **Режим разметки**. В этом режиме экранное представление документа полностью соответствует печатному документу, вплоть до назначенных параметров печатной страницы. Этот режим удобен для большинства работ, связанных с форматированием текста, предназначенного для печати.
3. **Режим структуры**. В этом режиме документ отображается с утрированным выделением его структуры. Режим полезен в тех случаях, когда разработку документа начинают с создания плана содержания. Он отличается тем, что при его включении на панели инструментов автоматически открывается вспомогательная панель **Структура**, элементы управления которой позволяют править структуру документа.
4. **Режим *Web*-документа**. В этом режиме экранное представление не совпадает с печатным документом. 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.
5. **Режим Схема документа (режим чтения)**. В этом режиме 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 при просмотре документов сложной структуры.

6. **Режим предварительного просмотра *Web*-страницы**. Этот режим используется для предварительного просмотра электронных документов. При этом созданный документ отображается как *Web*-страница.
7. **Режим предварительного просмотра**. Этот режим используется для просмотра печатных документов. При этом документ представляется в специальном окне.

Выбор первых пяти режимов выполняется соответствующими командами меню **Вид**. А выбор первых четырех режимов, кроме того, можно выполнить с помощью соответствующих кнопок, расположенных в левом нижнем углу рабочего окна Word.

Выбор 6 и 7 режимов выполняется соответствующими командами меню **Файл**.

6.4. Приемы работы с текстами в процессоре MS Word

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

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

6.4.1. Создание документа

В текстовом процессоре Word 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 готового шаблона выполняется следующими действиями.

1. Применительно к текстовому процессору Word 2000:

- открыть меню **Файл** и выполнить команду **Создать.** В появившемся диалоговом окне **Создание файла** включить переключатель **Создать документ** и выбрать шаблон **Обычный** на вкладке **Общие** или другой нужный шаблон. Созданный документ приобретает по умолчанию имя **Документ 1**;
- открыть меню **Файл** и выполнить команду **Сохранить как...**;
- в диалоговом окне **Сохранение документа** выбрать папку, в которой будет храниться новый документ, или создать новую папку, используя кнопку **Создание новой папки**;
- в диалоговом окне **Сохранение документа** в окне **Имя файла** удалить имя **Документ 1** и ввести новое нужное имя, а затем щелкнуть мышью по кнопке **Сохранить.**

2. Применительно к текстовому процессору Word 2003:
   - открыть меню **Файл** и выполнить команду **Создать.** В появившемся окне **Создание документа** (в правой части рабочего окна) в разделе **Шаблоны** выбрать и выполнить команду **На моем компьютере.** В появившемся диалоговом окне **Шаблоны** на вкладке **Общие** выбрать шаблон **Новый документ** или другой нужный шаблон. Созданный документ приобретает по умолчанию имя **Документ 1**;
   - открыть меню **Файл** и выполнить команду **Сохранить как...**;
   - в диалоговом окне **Сохранение документа** выбрать папку, в которой будет храниться новый документ, или создать новую папку, используя кнопку **Создание новой папки**;
   - в диалоговом окне **Сохранение документа** в окне **Имя файла** удалить имя **Документ 1** и ввести новое нужное имя, а затем щелкнуть мышью по кнопке **Сохранить.**
3. Простой способ создания документа для обеих версий текстового процессора Word:
   - щелкнуть мышью по кнопке **Создать** стандартной панели инструментов (самая левая кнопка);
   - открыть меню **Файл** и выполнить команду **Сохранить как...**;

- в диалоговом окне **Сохранение документа** выбрать папку, в которой будет храниться новый документ, или создать новую папку, используя кнопку **Создание новой папки**;
- в диалоговом окне **Сохранение документа** в окне **Имя файла** удалить имя **Документ 1** и ввести новое нужное имя, а затем щелкнуть мышью по кнопке **Сохранить**.

6.4.2. Ввод текста

Технология ввода текста и переключения языковых регистров клавиатуры, применение регистровых клавиш и буфера обмена не требуют особых пояснений. Поэтому в данном разделе рассматриваются специальные средства при работе с текстом.

6.4.2.1. Средства отмены и возврата действий

Все операции ввода, редактирования и форматирования текста протоколируются текстовым процессором. Поэтому необходимое количество последних действий всегда можно при необходимости отменить.

Последнее действие отменяется комбинацией клавиш **Ctrl+Z**. Серией этой команды можно отменить серию последних действий.

Отменить последнее действие (серию последних действий) можно также при помощи кнопки **Отменить** на панели инструментов **Стандартная** или последовательностью команд: меню **Правка**, строка **Отменить действие** (в этой строке сообщается конкретное действие).

После отмены ряда действий можно вернуться к исходному состоянию, которое было до отмены. Для этого нужно воспользоваться кнопкой **Вернуть** на панели инструментов **Стандартная** или последовательностью команд: меню **Правка**, строка **Вернуть действие** (в этой строке сообщается конкретное действие).

6.4.2.2. Ввод специальных и произвольных символов

При вводе текста может возникать необходимость ввода специальных символов, отсутствующих на клавиатуре или произвольных символов. Основным средством для ввода специальных и произвольных символов является диалоговое окно **Символ** (меню

Вставка, строка **Символ**). Данное окно имеет две вкладки: **Символы Специальные символы.**

На вкладке **Специальные символы** присутствует список специальных символов и клавиатурные комбинации, которые можно использовать при вставке таких символов. Без применения клавиатурных комбинаций специальные символы можно вставлять следующими действиями:

- открыть меню **Вставка**;
- щелкнуть мышью по строке **Символ**;
- в диалоговом окне выбрать нужный символ и щелкнуть по кнопке **Вставить**.

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

Если вставка символа производится только один раз, то его достаточно выбрать в таблице символов соответствующего набора и щелкнуть мышью по кнопке **Вставить**, а затем по кнопке **Закрыть**.

Если предполагается многократная вставка символа, то за ним лучше закрепить постоянную комбинацию клавиш.

6.4.2.3. Специальные средства редактирования текста

Текстовый процессор имеет два режима редактирования: режим замены и режим вставки.

В режиме вставки вводимый текст раздвигает существующий текст. Он применяется при разработке содержательных основных блоков текстового документа.

В режиме замены новые символы замещают символы предшествующего текста, находящиеся в точке ввода. Режим замены применяется при редактировании стандартных форм и стандартных элементов.

Текущая правка текста индицируется в строке состояния рабочего окна индикатором **ЗАМ** (Замена), если установлен режим замены. В

противном случае индикатор **ЗАМ** выключен. Переключение режима можно выполнить двойным щелчком мыши по этому индикатору.

Настройка режимов редактирования выполняется следующими действиями:

- открыть меню **Сервис**;
- в раскрывшемся меню щелкнуть мышью по строке **Параметры**;
- в диалоговом окне **Параметры** щелкнуть мышью по вкладке **Правка**;
- в разделе **Параметры правки** установить или убрать флажки для соответствующих режимов.

Если установлены флажки **Режим замены** и **Использовать клавишу INS для вставки**, то редактирование текста возможно в режиме замены символов.

Если указанные выше флажки сброшены (убраны), то режим можно выбирать с помощью клавиши **Insert** (**Ins**).

Если флажок **Режим замены** сброшен (убран), а флажок **Использовать клавишу INS для вставки** установлен, то редактирование возможно в режиме вставки.

В процессе ввода текста очень важно вовремя увидеть орфографические и синтаксические ошибки, допущенные при вводе текста. Эта потребность учтена и реализована разработчиками текстового процессора Word, который располагает средствами автоматизации проверки правописания. Эти средства включают возможность проверки орфографии и грамматики (синтаксиса).

Средства автоматизации проверки правописания включают два режима – автоматический и командный.

Для установки автоматического режима необходимо выполнить следующие действия:

- открыть меню **Сервис**;
- в раскрывшемся меню щелкнуть мышью по строке **Параметры**;
- в диалоговом окне **Параметры** щелкнуть мышью по вкладке **Правописание**;
- в окне вкладки **Правописание** установить флажки для включения необходимых элементов управления.

В этом режиме слова, содержащие орфографические ошибки, подчеркиваются красным цветом, а выражения, содержащие грамматические (синтаксические) ошибки, – зеленым. Для того чтобы узнать характер ошибки, нужно щелкнуть правой кнопкой мыши на

помеченном слове (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 предлагаются варианты исправления предполагаемой ошибки.

В командном режиме проверка правописания выполняется независимо от установки элементов управления на вкладке **Правописание**. Запуск командного режима выполняется следующими действиями: открыть меню **Сервис** и щелкнуть мышью по строке **Правописание**. Проверка начинается от начала документа и продолжается до появления первой ошибки. В тех случаях, когда происходит отказ от предлагаемых исправлений командой **Пропустить**, в документе накапливается список пропускаемых выражений. Для того чтобы очистить этот список и начать проверку заново, необходимо выполнить следующие действия:

- открыть меню **Сервис**;
- в раскрывшемся меню щелкнуть мышкой по строке **Параметры**;
- в диалоговом окне Параметры щелкнуть мышью по вкладке **Правописание**;
- в открывшейся вкладке **Правописание** щелкнуть мышью по кнопке **Повторная проверка**.

6.4.2.4. Форматирование текста

Форматирование текста осуществляется средствами меню **Формат** или панели инструментов **Форматирование**.

Панель инструментов **Форматирование** включает следующие средства:

- **Стиль** (меню выбора стиля документа);
- **Шрифт** (меню выбора гарнитуры шрифта);
- **Выбрать размер шрифта** (меню размеров шрифта);
- кнопки для установки начертания текста (**Полужирный**, **Курсив**, **Подчеркнутый**);
- кнопки для выравнивания текста (**По левому краю**, **По центру**, **По правому краю**, **По ширине**);
- **Междустрочный интервал** (меню междустрочных интервалов);

- кнопки (**Нумерованный список по умолчанию, Маркированный список по умолчанию**);
- кнопки (**Уменьшить отступ**, **Увеличить отступ**);
- **Внешние границы** (панель для выбора границ рамок);
- **Выделение цветом** (палитра для выбора цвета фона);
- **Цвет шрифта** (палитра для выбора цвета шрифта).

Перечисленные выше средства применяются для текущего форматирования в процессе ввода текста.

Меню **Формат** предназначается преимущественно для предварительных настроек средств форматирования. Основные настройки форматирования.

1. Настройка шрифта выполняется в диалоговом окне **Шрифт** (открыть меню **Формат** и в раскрывшемся меню щелкнуть мышью по строке **Шрифт**). Это окно имеет три вкладки: **Шрифт**, **Интервал**, **Анимация**. На вкладке **Шрифт** выбирается гарнитура шрифта, его размер, начертание, цвет символов, подчеркивание и видоизменение символов. На вкладке **Интервал** выбирается масштаб символов; обычный, разреженный или уплотненный интервал между символами; смещение символов и кернинг для знаков. Вкладка **Анимация** применяется только при подготовке электронных документов и не применяется для текстовых документов.
2. Настройка параметров абзаца и выравнивания текста выполняется в диалоговом окне **Абзац** (открыть меню **Формат** и в раскрывшемся меню щелкнуть мышью по строке **Абзац**). Это окно имеет две вкладки: **Отступы и интервалы**, **Положение на странице**. На вкладке **Отступы и интервалы** выбираются: выравнивание текста (По левому краю, По центру, По правому краю, По ширине), уровень текста, отступ (величина отступа от левого края, величина отступа от правого края, величина отступа первой (красной) строки абзаца), величина интервала перед абзацем и после него.
3. Настройка нумерованных и маркированных списков (в том числе многоуровневых) выполняется в диалоговом окне Список (открыть меню **Формат** и в раскрывшемся меню щелкнуть мышью по строке **Список**). Это окно имеет четыре вкладки: **Маркированный список**, **Нумерованный список**, **Многоуровневый список**, **Список стилей**. В качестве элементов

управления этих списков представлены их образцы. Для выбора нужного достаточно щелкнуть мышью на выбранном образце. Вход в список может осуществляться автоматически или по команде. Чтобы автоматически создать маркированный список, достаточно начать запись строки с символа *. После завершения строки и нажатия 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 вводится точка и пробел. Этот метод позволяет начать нумерацию с любого пункта (не обязательно с единицы). Для создания списка по команде используются соответствующие кнопки панели **Форматирование**.

Глава 7. Базы данных Microsoft Access

Базы данных — это организованная структура для хранения информации.

Такое понятие базы данных обусловлено тем, что современные системы управления базами данных (СУБД) позволяют размещать в своих структурах не только данные, но и методы (то есть программный код), с помощью которых происходит взаимодействие с потребителем или с другими программно-аппаратными комплексами. Таким образом, в современных базах данных хранятся не только данные, но и информация. Если в базе данных нет никаких данных, то это все равно полноценная база данных. В такой базе данных все-таки есть информация — это структура базы. Она определяет методы занесения и хранения данных в базе.

С понятием базы данных тесно связано понятие системы управления базой данных.

Система управления базами данных (СУБД) — ***это комплекс программных средств, предназначенных для создания структуры новой базы, наполнения ее содержимым, редактирования содержимого и визуализации информации.*** Под визуализацией информации базы понимается отбор отображаемых данных в соответствии с заданным критерием, их упорядочение, оформление и последующая выдача на устройство вывода или передача по каналам связи.

В мире существует множество СУБД. Однако большинство из них опирается на единый устоявшийся комплекс основных понятий. Поэтому рассмотрим и обобщим понятия только одной системы управления базами данных **Microsoft Access**, входящей в пакет Microsoft Office. Базы данных могут содержать различные объекты, но основными объектами являются ее таблицы. Структуру двумерной таблицы образуют столбцы и строки. Их аналогами в структуре базы данных являются соответственно ***поля*** и ***записи***. Поля базы данных определяют групповые свойства данных, записываемых в ячейки, принадлежащие каждому из полей. Изменение состава полей исходной таблицы (или их свойств) приводит к изменению структуры базы данных, то есть создается новая база данных.

7.1. Типы данных Microsoft Access

Система управления базами данных Microsoft Access поддерживает следующие типы данных:

- ***Текстовый*** — используется для хранения обычного неформатированного текста ограниченного размера (до 255 символов);
- ***Поле Мемо*** — специальный тип данных для хранения больших объемов текста (до 65535 символов);
- ***Числовой*** — тип данных для хранения действительных чисел;
- ***Дата/время*** — тип данных для хранения дат и текущего времени;
- ***Денежный*** — тип данных для хранения денежных сумм;
- ***Счетчик*** — специальный тип данных для натуральных уникальных чисел с автоматическим наращиванием (для порядковой нумерации записей);
- ***Логический*** — тип для хранения логических данных (только два значения **Да** или **Нет**);
- ***Поле объекта OLE*** — специальный тип данных для хранения объектов ***OLE***, например мультимедийных;
- ***Гиперссылка*** — специальное поле для хранения адресов URL *Web*-объектов Интернета;
- ***Мастер подстановок*** — это не тип, а объект, настройкой которого можно автоматизировать ввод данных в поле так, чтобы не вводить их вручную, а выбирать из раскрывающегося списка.

7.2. Основные свойства полей таблиц базы данных СУБД Microsoft Access

В СУБД Microsoft Access реализуются следующие основные свойства полей в структурах таблиц базы данных:

- ***Имя поля*** (заголовок столбца таблицы) — определяет порядок обращения с данными этого поля при автоматических операциях с базой;
- ***Тип поля*** — определяет тип данных, которые могут содержаться в данном поле;
- ***Размер поля*** — определяет предельную длину (в символах) данных, которые могут размещаться в данном поле;
- ***Формат поля*** — определяет способ форматирования данных в ячейках, принадлежащих полю;
- ***Маска ввода*** (средство автоматизации ввода) — определяет форму, в которой вводятся данные в поле;
- ***Подпись*** — определяет заголовок столбца таблицы для данного поля (если подпись не указана, то в качестве заголовка столбца используется свойство ***Имя поля***);
- ***Значение по умолчанию*** — то значение, которое вводится в ячейки поля автоматически (средство автоматизации ввода данных);
- ***Условие на значение*** — ограничение, используемое для проверки правильности ввода данных (средство автоматизации ввода, которое используется, как правило, для данных, имеющих числовой тип, денежный или тип даты);
- ***Сообщение об ошибке*** — текстовое сообщение, которое выдается автоматически при попытке ввода в поле ошибочных данных (проверка ошибочности выполняется автоматически, если задано свойство ***Условие на значение***);
- ***Обязательное поле*** — свойство, определяющее обязательность заполнения данного поля при наполнении базы;
- ***Пустые строки*** — свойство, разрешающее ввод пустых строковых данных (от свойства ***Обязательное поле*** отличается тем, что относится не ко всем типам данных, а лишь к некоторым (например, текстовым));

- ***Индексированное поле*** — если поле обладает этим свойством, все операции, связанные с поиском или сортировкой записей по значению, хранящемуся в данном поле, существенно ускоряются.

Примечание: перечень приведенных свойств полей относится в основном к полям текстового типа.

Поскольку в разных полях могут содержаться данные разного типа, то и свойства у полей могут различаться в зависимости от типа данных.

7.3. Основные типы объектов базы данных СУБД Microsoft Access

СУБД Microsoft Access позволяет создавать шесть основных типов объектов.

1. ***Таблицы*** — это определяющие объекты любой базы данных. В таблицах хранятся все данные, имеющиеся в базе, и структура базы (поля, их типы и свойства).
2. ***Запросы*** — это объекты, предназначенные для извлечения данных из таблиц и предоставления их пользователям в удобном виде. С помощью запросов можно: выполнять отбор данных, их сортировку и фильтрацию; преобразовывать данные по заданному алгоритму; создавать новые таблицы; выполнять автоматическое заполнение таблиц данными, импортированными из других источников; выполнять простейшие вычисления в таблицах и другое.
3. ***Формы*** — это средства для ввода данных.
4. ***Отчеты*** — это средства для вывода данных на печатающее устройство (принтер).
5. ***Страницы*** — это специальные объекты баз данных (страницы доступа к данным). Это объект, выполненный в коде HTML, размещаемый на *Web*-странице и передаваемый клиенту вместе с ней.
6. ***Макросы и модули*** — это категории объектов, предназначенные для автоматизации повторяющихся операций при работе с СУБД и для создания новых функций путем программирования. Макросы состоят из последовательности внутренних команд СУБД и являются средством автоматизации работы с базой. Модули

создаются средствами внешнего языка программирования Visual Basic for Applications (VBA).

СУБД Microsoft Access предоставляет следующие средства создания основных объектов базы:

- ручные средства разработки объектов в режиме ***Конструктора***;
- автоматизированные средства разработки объектов с помощью программ-мастеров;
- автоматические средства ускоренной разработки простейших объектов.

Примечание: при изучении и овладении СУБД Microsoft Access рекомендуется начинать работу в режиме ***Конструктора*** для разработки учебных таблиц и запросов. При разработке учебных форм, отчетов и страниц доступа рекомендуется использовать автоматизированные средства, предоставляемые мастерами.

7.4. Работа с СУБД Microsoft Access

Технологии по созданию базовых таблиц, межтабличных связей и запросов рассмотрим на конкретном примере.

Постановка задачи. Руководитель малого предприятия, выполняющего сборку персональных компьютеров из готовых компонентов, заказал разработку базы данных, основанной на двух таблицах комплектующих. Одна таблица содержит данные, которые могут отображаться для клиентов при согласовании спецификации изделия. В ней указаны розничные цены и компоненты. Вторая таблица предназначена для анализа результатов деятельности предприятия. В ней содержатся оптовые цены на компоненты и краткая информация о поставщиках (клиенты предприятия не имеют права доступа к данным этой таблицы).

7.4.1. Технология создания базовых таблиц

1. Запустить программу Microsoft Access (двойной щелчок мышью на рабочем столе по ярлычку ***Microsoft Access*** или меню ***Пуск*** в нем ***Программы (Все программы)***, затем строка ***Microsoft*** **Office,** и в появившемся меню щелкнуть по строке ***Microsoft Access***).

2. В окне ***Microsoft Access*** включить переключатель ***Новая база данных*** и щелкнуть по кнопке ***Ok***.
3. В окне ***Файл новой базы данных*** выбрать папку: ***\Мои документы*** и ввести имя файла: ***Комплектующие***. Убедиться, что в качестве типа файла выбрано ***Базы данных Microsoft Access,*** и щелкнуть по кнопке ***Создать***. Откроется окно новой базы — ***Комплектующие: база данных***.
4. Открыть панель ***Таблицы***.
5. Открыть бланк создания структуры таблицы двойным щелчком мыши по ярлычку ***Создание таблицы в режиме конструктора***; для первой таблицы ввести следующие поля:

Имя поля	***Тип данных***
Компонент	*Текстовый*
Модель	*Текстовый*
Основной параметр	*Числовой*
Цена	*Числовой*

6. Щелкнуть мышью на поле ***Цена*** и в нижней части бланка задать свойство ***Число десятичных знаков*** равным 2.
7. Для связи с будущей таблицей поставщиков нужно задать ключевое поле. Однако в данном примере ни одно поле явно не является ключевым, поэтому нужно использовать комбинацию полей ***Компонент*** и ***Модель***. Для этого выделить оба поля в верхней части бланка при нажатой клавише ***Shift (Alt)***. Щелчком правой кнопки мыши открыть контекстное меню и выбрать в нем пункт ***Ключевое поле***.
8. Закрыть окно конструктора. При закрытии ввести имя таблицы ***Комплектующие***.
9. Создать таблицу ***Поставщики***, повторив действия пунктов 5, 6, 7 и 8, в которую ввести следующие поля:

Имя поля	***Тип данных***
Компонент	*Текстовый*
Модель	*Текстовый*
Цена оптовая	*Числовой*
Поставщик	*Текстовый*
Адрес	*Текстовый*

Примечание: ключевое поле для этой таблицы не создавать, так как в этой задаче оно не требуется.

10. В окне ***Комплектующие: база данных*** открыть по очереди обе таблицы и ввести в них 3-4 записи по каждому полю. После заполнения таблиц их закрыть и завершить работу с программой.

7.4.2. Технология создания межтабличных связей

1. Запустить программу ***Microsoft Access***.
2. В окне ***Microsoft Access*** включить переключатель ***Открыть базу данных,*** выбрать ранее созданную базу ***Комплектующие*** и щелкнуть по кнопке ***Ok***.
3. В окне ***Комплектующие: базы данных*** открыть панель ***Таблицы*** и убедиться, что на панели присутствуют ярлычки ранее созданных таблиц ***Комплектующие*** и ***Поставщики***.
4. Открыть окно ***Схема данных*** щелчком мыши по кнопке ***Схема данных*** на панели инструментов (если такой кнопки нет, то раскрыть меню ***Сервис*** и выбрать строку ***Схема данных***). Одновременно с открытием окна ***Схема данных*** откроется диалоговое окно ***Добавление таблицы***. На вкладке ***Таблицы*** этого окна выбрать таблицы, между которыми создаются связи.
5. Выбрать таблицы ***Комплектующие*** и ***Поставщики*** щелчком мыши по кнопке ***Добавить***. В окне ***Схема данных*** откроются списки полей этих таблиц.
6. В таблице ***Комплектующие*** при нажатой клавише ***Shift (Alt)*** выделить поля ***Компонент*** и ***Модель*** и перетащить эти поля на список полей таблицы ***Поставщики***. После перетаскивания автоматически открывается диалоговое окно ***Изменение связей***.
7. Выбрать на правой панели окна ***Изменение связей*** поля ***Компонент*** и ***Модель*** таблицы ***Поставщики,*** включаемые в связь.

   Примечание: в данной работе не нужно устанавливать флажок ***Обеспечение целостности данных***.
8. Закрыть окно ***Изменение связей*** и в окне ***Схема данных*** проанализировать образовавшуюся связь. Щелчком левой кнопкой мыши проверить возможность выделения линии связи, а щелчком правой кнопки мыши проверить открытие контекстного меню, позволяющего разорвать связь или отредактировать ее.

9. Закрыть окно ***Схема данных*** и закрыть программу ***Microsoft Access***.

7.4.3. Технология создания запросов на выборку

Задание. Создать запрос на выборку жестких дисков емкостью не менее 20 Гбайт по цене не более 2500 рублей. Результирующая таблица должна содержать также адрес поставщика и номер его телефона.

1. Запустить программу ***Microsoft Access***.
2. В окне ***Microsoft Access*** включить переключатель ***Открыть базу данных***, выбрать ранее созданную базу ***Комплектующие*** и щелкнуть по кнопке ***Ok***.
3. В окне ***Комплектующие: базы данных*** открыть панель ***Запросы***. Двойным щелчком мыши по ярлычку ***Создание запроса в режиме Конструктора***. Откроется бланк запроса по образцу и диалоговое окно ***Добавление таблицы***.
4. В окне ***Добавление таблицы*** выбрать таблицу Поставщики, щелкнуть мышью по кнопке ***Добавить*** и закрыть окно ***Добавление таблицы***.
5. Двойными щелчками мышью на именах полей в списке полей таблицы ***Поставщики*** выбрать поля, включаемые в результирующую таблицу: ***Компонент***, ***Модель***, ***Цена оптовая***, ***Поставщик***, ***Адрес***.
6. Задать условие отбора для поля ***Компонент***. Для этого в соответствующую строку ввести: ***Жесткий диск***.
7. Задать условие отбора для поля ***Цена оптовая***. Для этого в соответствующую строку ввести: **<=2500**.
8. Далее необходимо задать условие отбора по основному потребительскому параметру — емкость жесткого диска. Однако в таблице ***Поставщики*** такого поля нет, но есть поле ***Модель***, которое однозначно определяет параметры изделия. Так как по полю ***Модель*** установлена связь с таблицей ***Комплектующие***, то имеется возможность ввести в запрос поле ***Основной параметр***, взяв его из другой таблицы. Для этого нужно добавить список полей таблицы ***Комплектующие*** в верхнюю часть бланка запроса по образцу, щелкнув правой кнопкой мыши в верхней области бланка. В открывшемся контекстном меню выбрать пункт ***Добавить таблицу***. Откроется окно ***Добавление таблицы***. В

этом окне выбрать таблицу ***Комплектующие***. Двойным щелчком мыши на поле ***Основной параметр*** в списке полей таблицы ***Комплектующие*** ввести это поле в бланк запроса по образцу.

9. В строке ***Условие отбора*** столбца ***Основной параметр*** ввести условие **>= 20** (емкость жесткого диска).
10. Закрыть бланк запроса по образцу. При закрытии бланка ввести его имя — ***Выбор комплектующих***.
11. В окне ***Комплектующие: база данных*** открыть только что созданный запрос и проанализировать результирующую таблицу. Если ни одно изделие не соответствует условию отбора и получившаяся результирующая таблица не имеет данных, то необходимо открыть базовые таблицы и откорректировать их содержание так, чтобы можно было проверить работу запроса.
12. По окончании исследований закрыть все открытые объекты и завершить работу с программой ***Microsoft Access***.

Следует отметить, что этот способ запросов имеет существенный недостаток, а именно пользователь базы данных работает с запросами, которые подготовил ему разработчик. Поэтому пользователь не имеет возможности изменить критерий (условие) отбора.

7.4.4. Технология создания запросов “с параметром”

Специальный тип запросов “с параметром” позволяет пользователю самому ввести критерий отбора данных на этапе запуска запроса. Этим критерием обеспечивается гибкость работы с базой.

Задание. создать простой запрос, позволяющий отбирать процессоры, предельную цену которых пользователь может задавать сам при запуске запроса.

1. Запустить программу ***Microsoft Access***.
2. В окне ***Microsoft Access*** включить переключатель ***Открыть базу данных***, выбрать ранее созданную базу ***Комплектующие*** и щелкнуть по кнопке ***Ok***.
3. В окне ***Комплектующие: базы данных*** открыть панель ***Запросы***. Двойным щелчком мыши по ярлычку ***Создание запроса в режиме Конструктора***. Откроется бланк запроса по образцу.
4. Создать запрос на выборку, основанный на таблице ***Поставщики***. В этот запрос должны войти следующие поля:
   - ***Компонент;***

- ***Модель;***
- ***Цена оптовая;***
- ***Поставщик;***
- ***Адрес.***

5. В строке ***Условие отбора*** поля ***Компонент*** ввести ***Процессор***.
6. В строку ***Условие отбора*** ввести текст: ***Введите максимальную цену***. Это дает пользователю возможность выбора критерия цены (в другой задаче — другого критерия).
7. Закрыть запрос. При закрытии сохранить его под именем ***Выбор комплектующих***.
8. В окне ***Комплектующие: база данных*** открыть панель ***Запросы*** и запустить запрос ***Выбор комплектующих*** — на экране появится диалоговое окно ***Введите значение параметра***. В этом окне ввести критерий цены процессора, например, **6500** и щелкнуть по кнопке ***Ok***. По результатам запроса будет сформирована результирующая таблица.
9. Закрыть все объекты базы данных и программу ***Microsoft Access***.

7.4.5. Технология создания итогового запроса

Если заполнить данными таблицу ***Комплектующие***, введя параметры всех компонентов, входящих в сборочную спецификацию компьютера, то можно узнать себестоимость комплектующих узлов. Запросы, выполняющие вычисления по всем записям для какого-либо числового поля, называются ***итоговыми запросами***. В итоговом запросе может рассчитываться сумма значений или величина среднего значения по всем ячейкам поля; может выбираться максимальное или минимальное значение данных в поле; может также исполняться иная итоговая функция. Итоговые запросы, как и запросы по образцу, готовятся с помощью бланка запроса по образцу.

Предположим, что малое предприятие собирает компьютеры трех классов: “Элитный”, “Офисный” и “Домашний”. Несмотря на то, что архитектура у всех классов близка, их компоненты заметно отличаются по цене и техническим параметрам. Следовательно, имеются заметные отличия в цене этих трех моделей.

Задание. Подготовить итоговый отчет, с помощью которого можно определять цену каждой модели компьютеров.

1. Запустить программу ***Microsoft Access***.

2. В окне ***Microsoft Access*** включить переключатель ***Открыть базу данных***, выбрать ранее созданную базу ***Комплектующие*** и щелкнуть по кнопке ***Ok***.
3. В окне ***Комплектующие: базы данных*** открыть панель ***Таблицы*** и выбрать таблицу ***Комплектующие***.
4. Щелчком мыши на ярлычке ***Конструктор*** открыть таблицу в режиме проектирования. Это необходимо для создания дополнительного поля ***Класс***, в котором будут храниться данные о том, для какого класса изделий предназначены компоненты.
5. В начало структуры таблицы вставить новое поле. Для этого выделить первое поле (***Компонент***) и нажать клавишу ***Insert***.
6. Ввести имя нового поля — ***Класс*** и его тип — ***Текстовый***.
7. Закрыть окно ***Конструктор***. При закрытии подтвердить необходимость изменить структуру таблицы.
8. Открыть таблицу **Комплектующие** и заполнить ее содержанием, введя для каждого класса данные по следующим изделиям:
   - Материнская плата;
   - Процессор;
   - Оперативная память;
   - Жесткий диск;
   - Корпус;
   - Дисковод CD-RW;
   - Дисковод гибких дисков;
   - Видеоадаптер;
   - Звуковая карта;
   - Клавиатура;
   - Мышь.

   Цены на эти изделия для каждого класса проставить произвольные, но разные. Прочие поля таблицы можно не заполнять, так как в формировании итогового запроса они не участвуют.
9. Закрыть таблицу ***Комплектующие***.
10. Открыть панель ***Запросы*** щелчком мыши по одноименной кнопке окна ***Комплектующие: базы данных***.
11. Двойным щелчком мыши по ярлычку ***Создание запроса в режиме конструктора*** открыть диалоговое окно ***Добавление***

таблицы и выбрать таблицу ***Комплектующие***. Закрыть окно ***Добавление таблицы***.

12. В бланк запроса по образцу ввести следующие поля таблицы ***Комплектующие***: ***Класс***, ***Компонент***, ***Цена***.
13. Для поля ***Класс*** включить сортировку по возрастанию. Для поля ***Цена*** включить сортировку по убыванию.
14. На панели инструментов Microsoft Access щелкнуть на кнопке ***Групповые операции*** или меню ***Вид***, затем строка ***Групповые операции***. Эта строка необходима для создания в нижней части бланка строки ***Групповые операции***. На ее базе создаются итоговые вычисления. Все поля, отобранные для запроса, получают в этой строке значение ***Группировка***.
15. Для поля ***Класс*** оставить в строке ***Групповые операции*** значение ***Группировка***. Для остальных полей щелкнуть в этой строке — появится кнопка раскрывающегося списка, из которого можно выбрать итоговую функцию для расчета значений в данном поле.
16. Для поля ***Цена*** выбрать итоговую функцию ***Sum*** для определения стоимости изделия как суммы стоимостей комплектующих.
17. Для поля ***Компонент*** выбрать итоговую функцию ***Count***, определяющую общее количество записей, вошедших в группу.
18. Закрыть бланк запроса и дать ему имя: ***Расчет стоимости изделия***. Запустить запрос и убедиться в правильности его работы.
19. Закрыть все объекты базы данных. Завершить работу программы Microsoft Access.

Вопросы для самоконтроля

1. Понятие базы данных.
2. Понятие системы управления базами данных.
3. Что такое поле базовой таблицы?
4. Что такое запись базовой таблицы?
5. Что содержит база данных, если в ней нет ни одной записи?
6. Основные типы данных в среде Microsoft Access.
7. Основные типы объектов в среде Microsoft Access.
8. Основные свойства полей таблицы данных.

9. Технология создания базовых таблиц.
10. Технология создания запросов “с параметром”.

Глава 8. Основы алгоритмизации

8.1. Понятие алгоритма. Свойства и способы описания алгоритмов

Алгоритм – это конечная последовательность точно определенных действий, приводящих к решению поставленной задачи. При составлении алгоритмов следует учитывать ряд требований, выполнение которых приводит к формированию необходимых свойств:

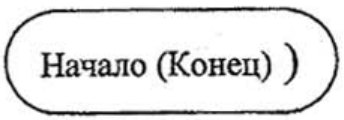
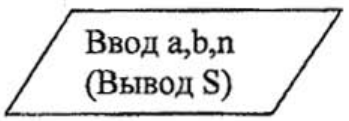
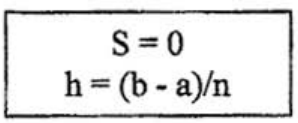
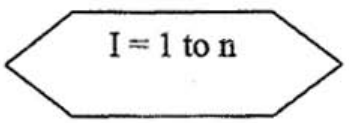
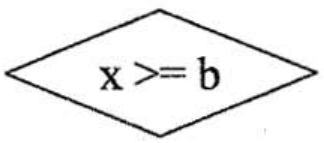
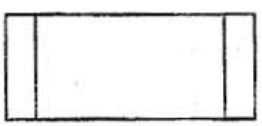
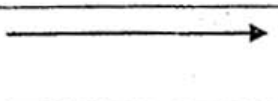
- алгоритм должен быть однозначным, исключающим произвольность толкования любого из предписаний и заданного полрядка исполнения. Это свойство алгоритма называется **определенностью**;
- любой алгоритм должен иметь только одно начало (один вход) и одно окончание (один выход);
- реализация процесса, предусмотренного алгоритмом, должна выдать результаты или сообщение о невозможности решения задачи. Это свойство алгоритма называется **результативностью**;
- способность алгоритма обеспечить решение однотипных задач с различными исходными данными. Это свойство называется **массовостью**;
- расчленение процесса, предусмотренного алгоритмом, на отдельные этапы, элементарные операции. Это свойство называется **дискретностью**.

Для строгого задания различных структур данных и алгоритмов, их обработки требуется иметь такую систему формальных обозначений и правил, чтобы смысл всякого используемого предписания трактовался точно и однозначно. Для выполнения этого условия или требования существуют следующие способы описания алгоритмов:

- словесное описание (запись на естественном языке);
- графическое описание;
- программное описание (тексты программ на алгоритмическом языке).

8.2. Графический способ описания алгоритма (блок–схема)

Для составления алгоритма в виде блок-схемы применяются следующие основные графические изображения.

Графический объект	Предназначение объекта
	Начало или конец алгоритма
	Ввод исходных данных или вывод результатов
	Выполнение операции или группы операций (Процесс)
	Безусловный цикл (Модификация)
	Выбор направления в зависимости от условия (Решение)
	Подпрограмма — процедура или Подпрограмма — функция (Типовой процесс)
	Линия потока, соединяющая фигуры блок-схемы. Направление линии указывать при ее ходе слева направо. Изменение направления линии под прямым углом
	Разрыв линии потока (Узел)
	Ссылка на другую страницу

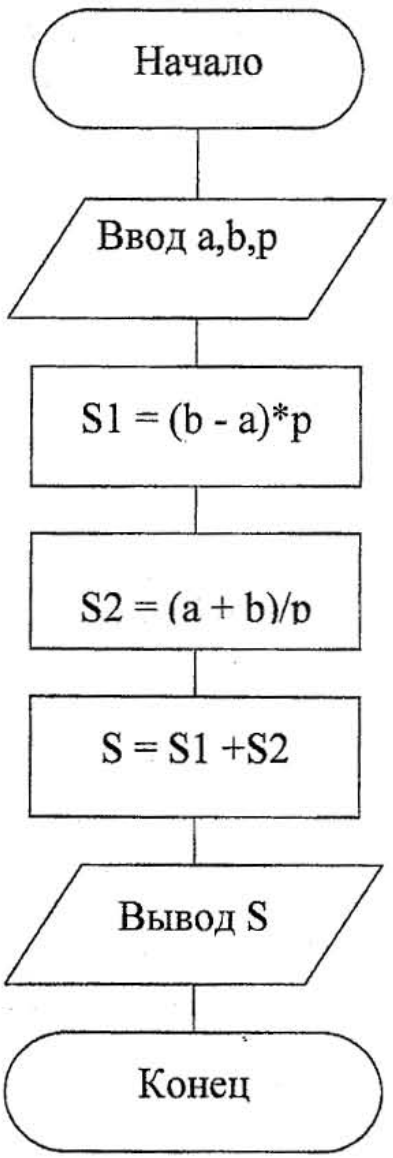
8.3. Базовые конструкции алгоритмов

При разработке и составлении блок-схемы того или иного алгоритма применяются следующие базовые конструкции: линейная, циклическая и ветвящаяся.

8.3.1. Линейная конструкция

Линейная конструкция – это последовательное выполнение операций без повторов и разветвлений.

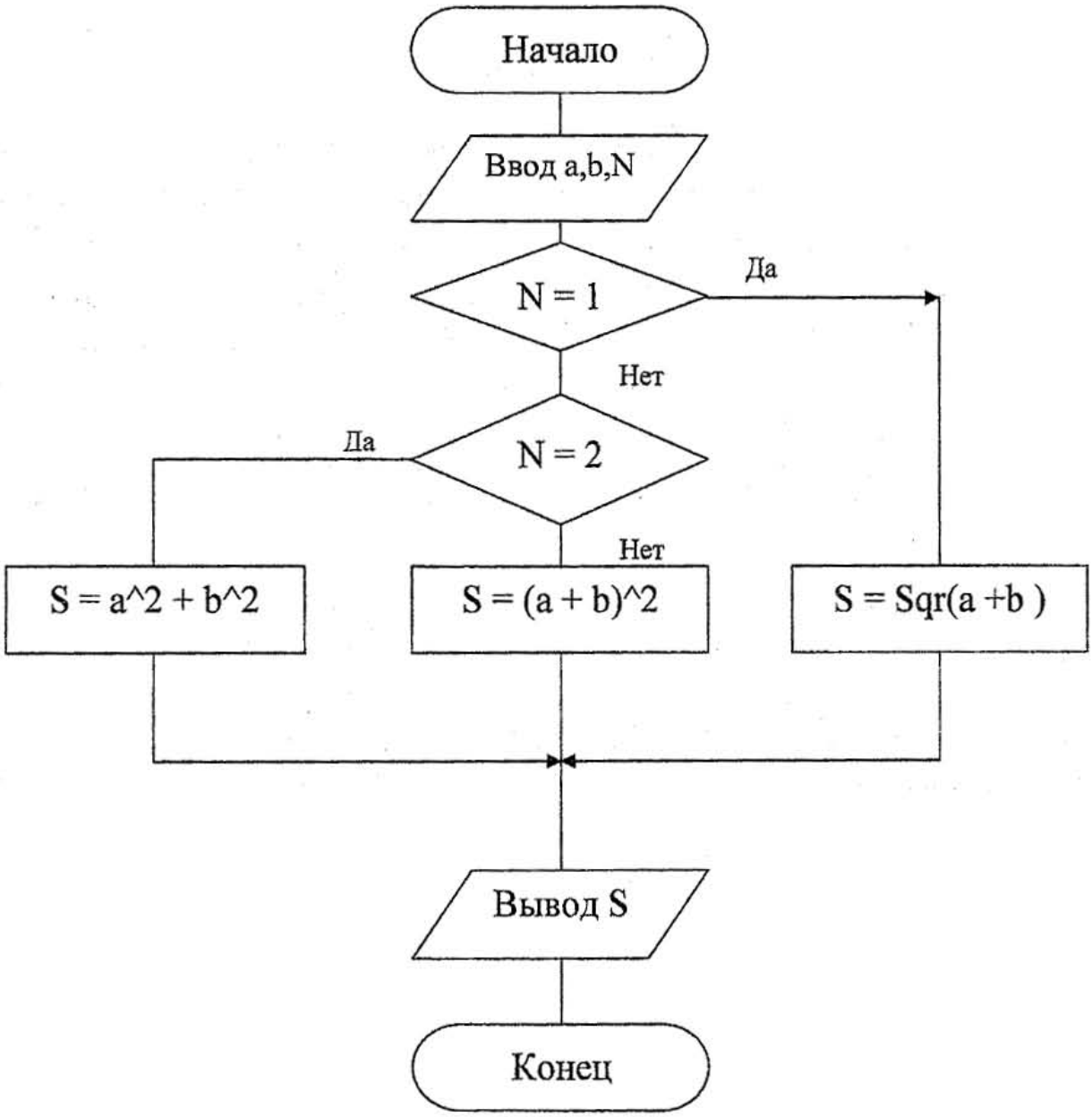
Пример



8.3.2. Ветвящаяся конструкция

Ветвящаяся конструкция – это выполнение операций по одному из нескольких направлений в зависимости от заданных условий.

Пример



8.3.3. Циклические конструкции

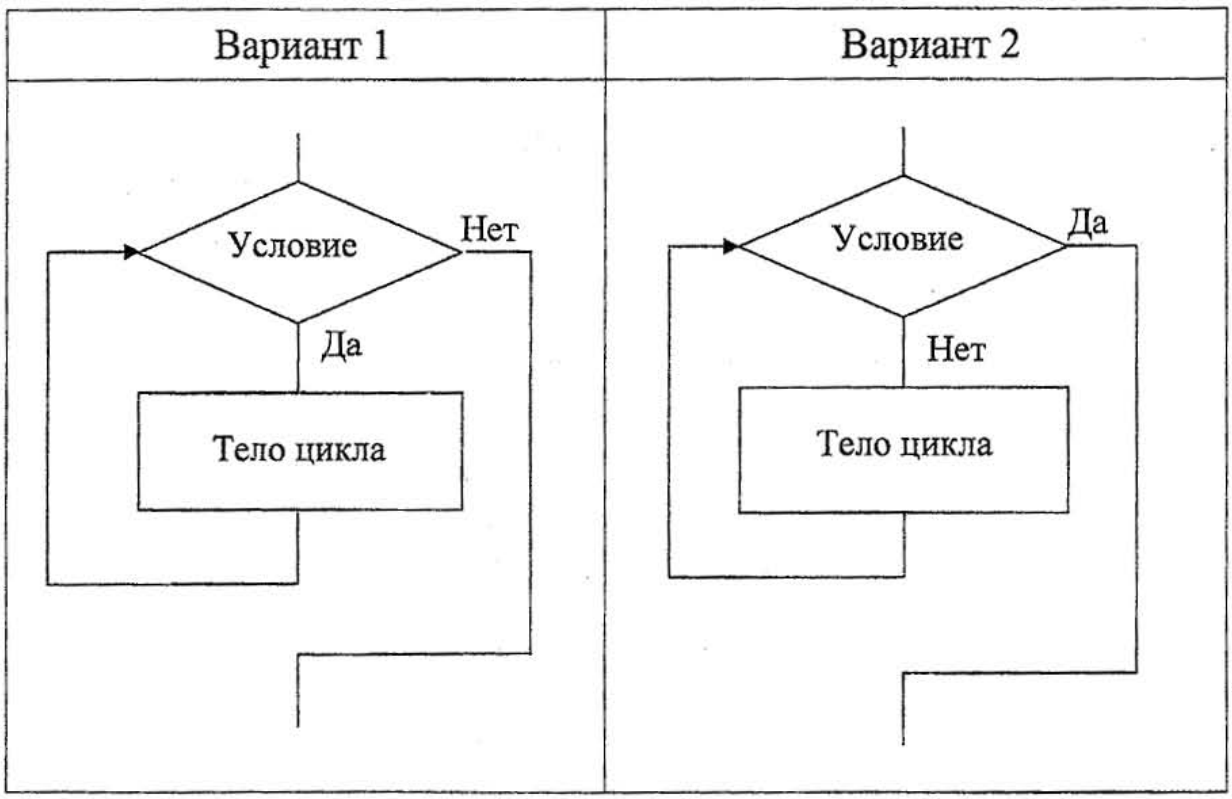
Циклы используются для организации повторного выполнения какой-либо операции (инструкции) или блока операций (инструкций). Цикл состоит из двух частей: **условие цикла** и **тело цикла**. У любого цикла есть **параметр**. **Параметр цикла** – это переменная, которая

изменяется в теле цикла, а также участвует в условии его окончания. Для организации повторов могут применяться следующие виды циклических конструкций: цикл с предусловием, цикл с постусловием, безусловный цикл (цикл с фиксированным количеством повторов или цикл по счетчику).

8.3.3.1. Цикл с предусловием

Конструкция цикла с предусловием в зависимости от результата выполнения условия может быть двух вариантов. В первом варианте повторение осуществляется до тех пор, пока условие имеет значение Истина (True). В другом варианте повторение осуществляется до тех пор, пока условие имеет значение Ложь (False).

Примеры:

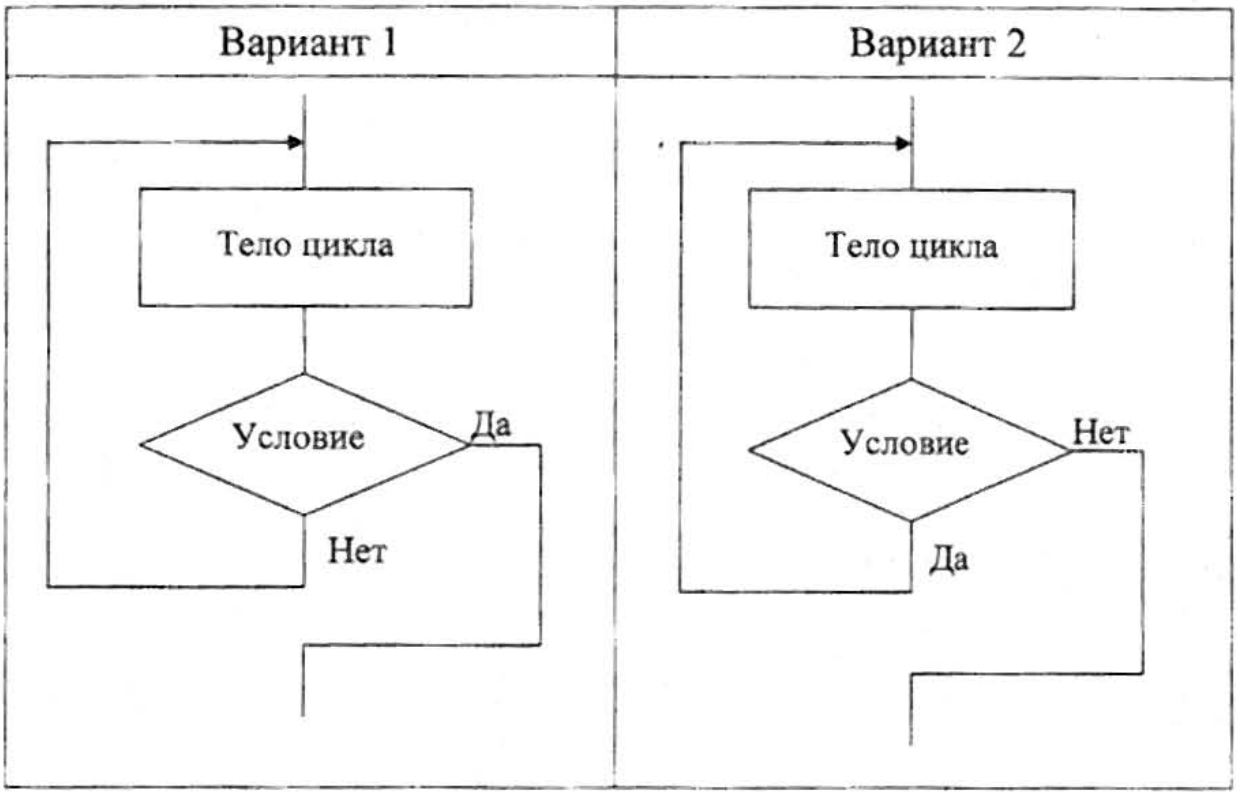


8.3.3.2. Цикл с постусловием

Конструкция цикла с постусловием в зависимости от результата выполнения условия может быть двух вариантов. В первом варианте повторение осуществляется до тех пор, пока условие имеет значение

Ложь (False). В другом варианте повторение осуществляется до тех пор, пока условие имеет значение Истина (True).

Примеры:



8.3.3.3. Безусловный цикл

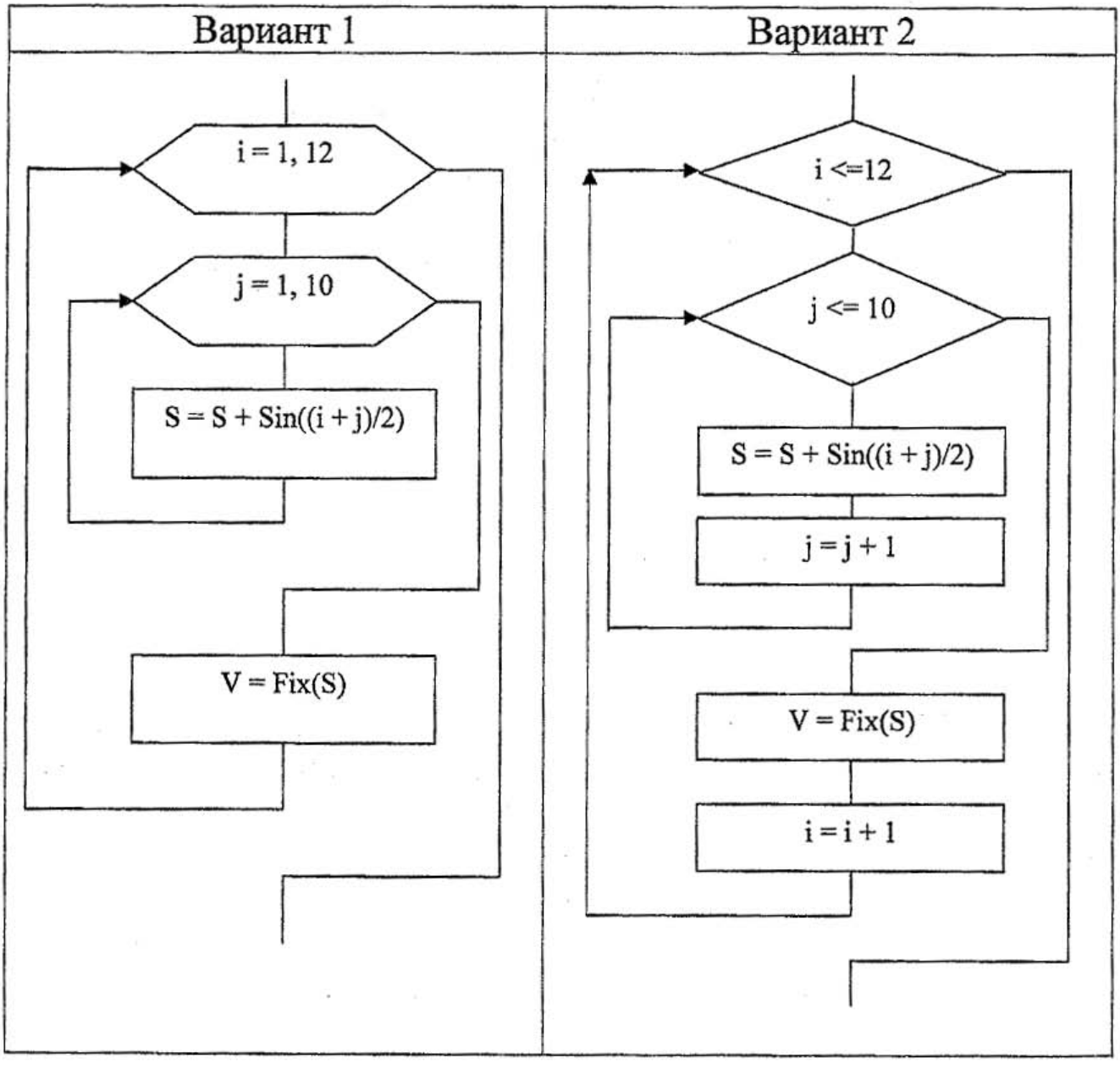
В этом цикле выполнение и повторение операций происходит от начального значения параметра (счетчика) до его конечного значения с указанным шагом. Если шаг не указан, то его значение полагается равным единице.

Пример:

Простые циклические конструкции могут вкладываться в другую простую циклическую конструкцию, образуя тем самым вложенный (сложный) цикл. При этом необходимо выполнять следующие правила:

- имена параметров всех простых циклов не должны повторяться;
- нельзя войти во внутренний цикл, минуя внешний;
- простые циклы в сложном цикле не должны пересекаться, то есть внешний цикл должен заканчиваться после внутреннего (инструкции тела внешнего цикла не должны быть в теле внутреннего цикла).

Пример:



Базовые конструкции алгоритмов в чистом виде на практике не применяются, а используются в сочетании между собой.

Глава 9. Основы программирования

9.1. Языки программирования

Языки программирования — это искусственные языки, которые отличаются от естественных языков ограниченным числом "слов", понятных 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. Правильно написанные команды языка, но не отвечающие алгоритму приводят к семантическим (логическим) ошибкам, которые называются ошибками времени выполнения. Процесс поиска ошибок в программе называется *тестированием*, а процесс устранения ошибок — *отладкой*.

9.1.1. Языки программирования низкого уровня

Языками программирования низкого уровня называют языки, которые ориентированы на конкретный тип процессора и учитывают его особенности. В таких языках операторы близки к машинному коду и ориентированы на конкретные команды процессора.

С помощью языков низкого уровня создаются очень эффективные и компактные программы. К их числу относятся: небольшие системные приложения, драйверы устройств, модули стыковки с нестандартным оборудованием и тому подобное. То есть в тех случаях, когда важнейшими требованиями становятся компактность, быстродействие и возможность прямого доступа к аппаратным ресурсам. Языком самого низкого уровня является язык *ассемблера*.

9.1.2. Языки программирования высокого уровня

Языками программирования высокого уровня называют языки, которые ориентированы на человека, более близки и понятнее ему. В таких языках не учитываются особенности компьютерных архитектур. Создаваемые программы на уровне исходных текстов средствами этих языков легко переносимы на другие платформы, имеющие транслятор

соответствующего языка. Создание языков высокого уровня по времени совпадает с появлением языков программирования третьего поколения (60-е годы XX века). Это универсальные языки, с их помощью удается решать задачи из любых областей. Относительная простота, независимость от конкретного компьютера и возможность использования мощных синтаксических конструкций позволили резко повысить производительность труда программистов. К числу языков этого поколения относится Basic (Бейсик). Для этого языка имеются и компиляторы, и интерпретаторы, а по популярности он занимает первое место в мире. Этот язык очень прост в изучении.

С начала 70-х годов XX века по настоящее время продолжается период языков четвертого поколения. Эти языки предназначены для реализации крупных проектов, повышения их надежности и скорости создания. Как правило, в эти языки встроены мощные операторы, позволяющие одной строкой описать такую функциональность, для реализации которой на языках младших поколений потребовались бы тысячи строк исходного кода. К числу языков этого поколения относятся: Pascal (Паскаль), C (Си), C++ (Си++), Java (Джава, Ява).

Паскаль (Pascal) во многом напоминает Алгол (язык 3-го поколения), но в нем ужесточен ряд требований к структуре программы и имеются возможности, позволяющие успешно применять его при создании крупных проектов.

Язык Си (C) 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. Этот язык во многом похож на Паскаль, однако имеет дополнительные средства (указатели) для прямой работы с памятью.

Си++ (C++) — это объектно-ориентированное расширение языка Си (C). Он имеет множество мощных возможностей, позволяющих резко повысить производительность программистов. Однако этот язык от программистов высокого уровня профессиональной подготовки.

Java (Джава, Ява) был создан в начале 90-х годов XX века компанией *Sun* на основе Си++. В этом языке исключены все низкоуровневые возможности языка Си++. Главной особенностью языка Java состоит в том, что компиляция происходит 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 (Java Virtual Machine),

версии которой сегодня созданы для любых платформ. Благодаря этому программы на Java можно переносить не только на уровне исходных текстов, но и на уровне двоичного байт-кода. Поэтому по популярности этот язык сегодня занимает второе место в мире после Бейсика.

Особое внимание в развитии языка Java уделяется двум направлениям:

- поддержке мобильных устройств и микрокомпьютеров, встраиваемых в бытовую технику (технология Jini);
- созданию платформно-независимых программных модулей, способных работать на серверах в глобальных и локальных компьютерных сетях с различными операционными системами (технология Java Beans).

В середине 90-х годов XX века появились языки пятого поколения. Это языки объектно-ориентированного программирования. К ним относятся системы автоматического создания прикладных программ с помощью визуальных средств разработки, без знания программирования. Главная идея, которая заложена в эти языки, — возможность автоматического формирования результирующего текста на универсальных языках программирования. Этот результирующий текст программы потом необходимо откомпилировать. Инструкции в компьютер вводятся в максимально наглядном виде с помощью методов, наиболее удобных для человека, не знакомого с программированием. Из этого поколения наиболее популярны следующие визуальные системы (среды) быстрого проектирования программ для Windows:

- ***Visual Basic for Applications*** (VBA) на базе языка программирования Basic (Бейсик);
- ***Borland Delphi*** (Делфи) на базе языка программирования Pascal (Паскаль);
- ***Borland C++Builder*** на базе языка программирования C++ (Си++);
- ***Symantec Cafe*** на базе языка программирования Java (Джава, Ява).

Для разработки серверных и распределенных приложений создана система программирования Microsoft Visual C++ .

9.1.3. Языки программирования баз данных

Эта группа языков отличается от алгоритмических языков 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). 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, удаление существующих записей.

Все эти операции возможны и осуществляются с помощью системы управления базами данных (СУБД). Основной поддержкой любой СУБД является 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.

Помимо поддержки языка ***SQL*** практически в каждой СУБД имеется также свой уникальный встроенный язык, ориентированный на особенности данной СУБД и не переносимый на другие системы. Ведущими производителями СУБД являются:

- Microsoft (встроенный язык ***SQL Server***);
- IBM (встроенный язык ***DB2***);
- Oracle (встроенный язык ***PL/SQL***);
- Adabas (встроенный язык ***Natural***);
- Informix (встроенный язык ***INFORMIX 4GL***).

Для персональных компьютеров (ПК) в настоящее время применяются языки ***Visual FoxPro*** фирмы Microsoft и ***Visual dBase*** фирмы Inprise.

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 называют ***CASE-системами***. В них активно применяются нотации ***IDEF*** и популярный язык графического моделирования ***UML***.

9.1.4. Языки программирования для Интернета

Языки программирования для Интернета называют скрипт-языками.

Для оформления документов предназначен язык ***HTML***. Он очень прост и содержит элементарные команды формирования текста, добавления рисунков, задания шрифтов и цветов, организации ссылок и таблиц.

Для эффективной обработки больших текстовых файлов, генерации текстовых отчетов и управления задачами предназначен язык ***Perl***. В него введено много часто используемых функций работы со строками, массивами, всевозможные средства преобразования данных, управления процессами, работы с системной информацией и др. По мощности этот язык значительно превосходит языки типа Си.

Для автоматизации рутинных процессов предназначен язык ***Tcl/Tk*** и состоит из мощных команд, предназначенных для работы с абстрактными нетипизированными объектами. Язык независим от типа системы и позволяет создавать программы с графическим интерфейсом.

Для организации виртуальных трехмерных интерфейсов в Интернете предназначен язык ***VRML***. Он позволяет описывать в текстовом виде различные трехмерные сцены, освещение и тени, текстуры (покрытия объектов), создавать свои миры, путешествовать по ним, вращать в любых направлениях, масштабировать, регулировать освещенность и т.п.

9.2. Технологии программирования

9.2.1. Средства создания программ

Для создания программ на любом выбранном языке программирования нужно иметь следующие компоненты:

- ***Текстовый редактор***. Формировать текст программы в принципе можно в любом редакторе. Однако 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 популярных языков и дополнительно могут автоматически проверять правильность синтаксиса программы непосредственно во время ее ввода.

- ***Программа-компилятор***. С ее помощью исходный текст программы переводится в машинный код. Если в исходном тексте будут обнаружены синтаксические ошибки, то результирующий не будет создан. Для небольшой программы на этом этапе уже возможно получение готовой программы. Однако исходный текст большой программы, как правило, состоит из нескольких модулей (файлов с исходными текстами), так как все исходные тексты хранить в одном файле неудобно, потому что в них трудно ориентироваться. Поэтому каждый модуль компилируется в отдельный файл с объектным кодом (двоичный файл со стандартным расширением **.OBJ**). Такие файлы затем необходимо объединять в одно целое. Кроме того, к ним нужно добавить машинный код подпрограмм, реализующих стандартные функции. Такие функции содержатся в библиотеках (файлах со стандартным расширением .LIB), которые поставляются вместе с компилятором. Код модулей и подключенные к нему стандартные функции необходимо объединить с учетом требований операционной системы, то есть получить программу, отвечающую определенному формату.
- ***Программа редактор связей (сборщик)***. Эта программа 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, то *сборщик* сообщает об ошибке и тогда готовой программы не создается. Итоговый файл имеет расширение .EXE или .COM.

Совокупность всех изложенных выше составляющих представляет собой интегрированную систему программирования. В ней почти все этапы создания программы автоматизированы. Таким образом, после ввода исходного текста программы его компиляция и сборка выполняются одним щелчком мыши или нажатием клавиши.

9.2.2. Алгоритмическое программирование

Алгоритм — это формальное описание способа решения задачи путем разбиения ее на конечную по времени последовательность действий (элементарных операций).

Формальное описание это абсолютно полное и должно учитывать все возможные ситуации, которые могут встретиться по ходу решения задачи.

Элементарные операции — это действия, которые выполняются по заранее определенным критериям и не требуют детализации.

Основная идея алгоритмического программирования — это разбиение программы на последовательность модулей, каждый из которых выполняет одно или несколько действий. При этом выполнение каждого модуля всегда начиналось с первой команды и всегда заканчивалось на самой последней.

Алгоритм на выбранном языке программирования записывается с помощью команд описания данных, вычисления значений и управления последовательностью выполнения программы. При этом текст программы представляет собой линейную последовательность операторов присваивания, циклов и условных операторов. Таким способом можно решать не очень сложные задачи и составлять программы, содержащие несколько сот строк кода.

Для больших программ такая технология программирования мало приемлема. Так как возникает много проблем. К ним относятся:

- резкое снижение понятности исходного текста 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 запутанной. При попытке исправить один ошибочный оператор вносится несколько новых ошибок, связанных с особенностями работы этого оператора, результаты выполнения которого могут учитываться в самых разных местах программы;
- набрать и отладить длинную (более 1000 строк кода) линейную последовательность операторов крайне затруднено или практически невозможно.

9.2.3. Структурное программирование

Структурное программирование применяется при создании средних по размеру приложений (несколько тысяч строк исходного текста).

Идея структурного программирования заключается в том, что структура программы должна отражать структуру решаемой задачи так, чтобы алгоритм решения был ясно виден из исходного текста. С этой целью в программирование введено понятие подпрограммы — набора операторов, выполняющих нужное действие и не зависящих от других частей исходного кода. При использовании подпрограмм программа разбивается на мелкие подпрограммы (до 50 операторов — критический порог для быстрого понимания цели подпрограммы). Каждая такая подпрограмма 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. При этом обращаться к таким блокам можно по их названиям.

Наличие подпрограмм позволяет вести проектирование и разработку приложения сверху вниз — ***нисходящим проектированием***. Технология нисходящего проектирования предусматривает вначале выделение нескольких подпрограмм, решающих самые глобальные задачи, затем каждая из этих подпрограмм детализируется путем ее разделения на небольшое число других подпрограмм. Этот процесс продолжается до тех пор, пока вся задача будет реализована.

Такой подход позволяет постоянно мыслить на предметном уровне, не опускаясь до конкретных операторов и переменных. При этом имеется возможность не реализовывать сразу некоторые подпрограммы, временно их откладывать, пока не будут реализованы другие части. Преимуществом данной технологии является также простота отладки небольших подпрограмм, что существенно повышает общую надежность всей программы. Готовые подпрограммы можно использовать повторно, если такая возможность имеется в решаемой задаче. Кроме того, вместе с интегрированными системами программирования поставляются большие библиотеки стандартных подпрограмм, которые позволяют значительно повысить производительность труда программиста за счет чужой работы по созданию часто применяемых подпрограмм.

С появлением визуальных RAD-сред в системе Windows, идеология которой основана на событиях, широкую популярность приобрел событийный подход к созданию программ — ***событийно-ориентированное программирование***. Оно является развитием идей нисходящего проектирования, когда постепенно определяются и детализируются реакции программы на различные события.

События могут быть *пользовательскими*, возникающими в результате действий пользователя, *системными*, возникающими в операционной системе, и *программными*, генерируемыми самой программой.

Предельный объем приложений, создаваемых в разумные сроки (несколько месяцев) с помощью структурного и событийного программирования составляет сотни тысяч строк программного кода.

9.2.4. Объектно-ориентированное программирование

Реальные объекты окружающего мира обладают тремя базовыми характеристиками:

- имеют набор определенных свойств;
- способны разными методами изменять эти свойства;
- реагировать на события, возникающие как вне объекта, так и внутри его.

На этих базовых характеристиках и основано понятие объекта в языках программирования.

Объект — это совокупность свойств (структур данных, характерных для этого объекта), ***методов их обработки*** (подпрограммы изменения свойств) ***и событий, на которые данный объект может реагировать и которые приводят, как правило, к изменению свойств объекта.***

Суть парадигмы объектно-ориентированного программирования состоит в том, что не программы управляют данными, а данные, связанные друг с другом и описывающие какой-либо объект (реальный или воображаемый), на базе которого реализована модель того или иного явления, управляют программами.

Примечание: парадигма (греч. *paradeigma*)— это пример, образец, тип, модель. В нашем случае это модель.

В основе этой ключевой идеи лежат три основных механизма:

- инкапсуляция;

- наследование;
- полиморфизм.

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). Такой тип называется ***классом***, а каждый конкретный объект, имеющий структуру этого класса, называется ***экземпляром класса***. Класс содержит данные и функции, которые обрабатывают эти данные. Данные называются свойствами класса, а функции — методами класса. Методы воздействуют на объект. При работе с объектами возникают события. При появлении события система посылает объекту сообщение, которое может быть обработано методом, создаваемым при разработке класса. Такие методы называют обработчиками событий или процедурами обработки события. События могут возникать как в результате действий пользователя, так и в результате работы системы.

Объединение данных с методами в одном классе (типе) называется инкапсуляцией. Это механизм, связывающий вместе код и данные, которыми он манипулирует и одновременно защищает их от внешних помех и некорректного использования.

Важнейшей характеристикой класса является создание на его основе новых классов, сохраняющих все свойства и методы исходного класса и добавляющих собственные свойства и методы. ***Эта характеристика называется наследованием.*** Наследование позволяет создавать новые классы, повторно используя уже готовый исходный код и не тратя времени на его переписывание. ***Класс, не имеющий предшественников, называется базовым.***

В большинстве случаев методы базового класса приходится переопределять для классов-наследников, так как методы для одних наследников могут выполняться совсем не так, как для других. ***Свойство объектов переопределять методы наследуемого класса называется полиморфизмом.***

Технологии объектного, событийного и структурного программирования применены и объединены в визуальных RAD-системах. Эти системы содержат множество готовых классов, представленных в виде визуальных компонентов. Эти компоненты могут добавляться в программу одним щелчком мыши.

В рамки объектно-ориентированного подхода удачно вписывается концепция визуального программирования. Все визуальные составляющие, такие как формы, элементы управления, меню и панели инструментов, являются объектами со своими свойствами и методами. Эти объекты реагируют на определенные события. При визуальном подходе к разработке форм не требуется программного задания их основных свойств. Эти свойства можно задать при помощи мыши или установить их в окне свойств. Таким образом, визуальное программирование делает проектирование интерфейса программы намного более наглядным и быстрым.

Появление технологии объектно-ориентированного программирования (середина 80-х годов XX века) позволило создавать в разумные сроки приложения до нескольких миллионов строк программного кода.

9.3. Отладка программ

Любая достаточно объемная программа требует отладки, состоящей в поиске и устранении ошибок. В программах не делал ошибок только тот, кто никогда не писал их. Итак, ошибки – это объективная неизбежность или реальное воплощение этой неизбежности.

В любой разрабатываемой программе могут присутствовать различные ошибки. Однако, несмотря на их многообразие, наиболее характерные ошибки принято условно подразделять на три типа: ошибки компиляции, ошибки выполнения и логические ошибки.

Ошибки при компиляции возникают в неправильно составленных программных конструкциях. К ним могут относиться нарушения правил языка VBA (ошибочно записанные ключевые слова, пропущенные разделители, неверные типы данных, нарушения правил грамматики или пунктуации и другие).

Ошибки при выполнении проявляются на стадии выполнения программы после успешной компиляции. К таким ошибкам, как правило, относятся недопустимые математические операции (деление на ноль, вычисление логарифма отрицательного числа или нуля, вычисление квадратного корня отрицательного числа), некорректные данные, введенные пользователем, и им подобные.

Логические ошибки обусловлены нарушением логики решения той или иной задачи. Например, искажением метода решения задачи, ошибочной записью математической функции и другие. Эти ошибки не мешают выполнению программы (не приводят к ее прерыванию). Их проявление можно обнаружить по результатам решения задачи, ибо они будут неверными.

Некоторые из ошибок первого типа выявляются на стадии написания текста программы в окне **Code** (Программа). При этом неправильно написанная строка текста программы после нажатия клавиши **Enter** выделяется *красным цветом* и на экране отображается диалоговое окно с сообщением о возможной причине, вызвавшей ошибку. Эту ошибку надо определить и тут же устранить. Другие ошибки, не выявленные при написании текста программы, обнаруживаются либо при компиляции программы без ее выполнения **Debug/Compile** (Отладка, Компилировать), либо при выполнении программы, так как VBA каждый раз автоматически ее компилирует при запуске на выполнение. В этих случаях предполагаемое место ошибки выделяется *синим цветом* и на экране отображается диалоговое окно **Microsoft Visual Basic** с сообщением о возможной причине, вызвавшей ошибку. Эту ошибку надо определить и тут же устранить.

При выявлении ошибок во время выполнения программы на экране отображается диалоговое окно **Microsoft Visual Basic** с сообщением о номере ошибки и возможной причине, ее вызвавшей. В этом случае, если в диалоговом окне нажать кнопку **Debug** (Отладка), то в тексте программы желтым цветом будет выделена строка, вызвавшая ошибку, и по причине которой было прервано выполнение программы. Кроме того, на левом поле окна **Code** против этой строки появится стрелочка. При такой ситуации VBA переходит в режим прерывания, который позволяет более детально определить причину ошибки. Здесь появляется возможность узнать текущие значения переменных и свойств. Для этого достаточно расположить указатель мыши на имени переменной или свойства, что вызовет появление всплывающей подсказки с текущим значением переменной или свойства. Однако для установки режима всплывающей подсказки предварительно необходимо с помощью команды **Tools/Options** (Сервис, Параметры) в диалоговом окне **Options** (Параметры) установить флажок **Auto Data Tips** (Подсказки значений данных).

Труднее всего обнаружить и устранить логические ошибки. Их локализация связана с тщательным анализом алгоритма программы и привлечением средств отладки VBA.

К логическим ошибкам относятся и случайные ошибки, возникающие при написании в составе имен переменных букв другого алфавита, одинаковых по их начертанию. Например, русская буква **С** вместо латинской буквы **C** или наоборот. Визуально имена с такими ошибками не отличаются друг от друга, но при выполнении программы они воспринимаются как имена разных переменных. Простейшим средством предотвращения таких ошибок является инструкция **Option Explicit**, которая пишется самой первой в начале программы. Эта инструкция предписывает явно описывать все переменные, встречающиеся в программе, что позволяет компилятору указать на переменную с неправильной буквой как на неописанную переменную. Тогда ошибка легко выявляется и быстро устраняется.

Другие логические ошибки могут быть выявлены при пошаговом выполнении программы, то есть отладка программы в пошаговом режиме. Использование этого режима позволяет видеть результаты выполнения каждой строки программы.

Чтобы приступить к отладке программы в пошаговом режиме, необходимо сначала приостановить ее выполнение. При этом сама программа остается загруженной, но происходит ее остановка перед выполнением очередной инструкции. Такой режим называется режимом прерывания. Для прерывания выполнения программы можно воспользоваться одним из следующих способов:

- в процессе выполнения программы выполнить команду **Run/Break** (Запуск, Прервать). В этом случае отобразится строка программы, на которой было прервано ее выполнение;
- установить точку останова;
- запустить программу не через команды меню **Run** (Запуск), а через команды меню **Debug** (Отладка), команды которого предназначены для пошаговой отладки программ;
- добавить инструкцию (команду) **Stop** в программу (в прогнозируемом месте ошибки).

При поиске логической ошибки в большой по объему программе иногда целесообразно ее отладку производить по частям. Для этого предусмотрена точка останова. Точка останова – это строка программы,

на которой автоматически приостанавливается выполнение программы.

Для установки точки останова необходимо выполнить следующие действия:

- в окне программы установить курсор на строке, где требуется остановить программу;
- выполнить команду **Debug/Toggle Breakpoint** (Отладка, Точка останова) или нажать кнопку **Toggle Breakpoint** на панели инструментов **Debug**.

Установить точку останова можно более быстрым способом. Для этого напротив строки, в которой требуется установить точку останова, в левом поле окна программы (на поле полосы индикатора) необходимо щелкнуть кнопкой мыши.

После установки точки останова строка с установленной точкой останова будет помечена кирпичным цветом, а на полосе индикатора, в поле напротив выделенной строки, появится круг того же цвета.

Для отмены (снятия) точки останова необходимо выполнить команду **Debug/Clear All Breakpoint** (Отладка, Снять все точки останова) или установить курсор на строку с точкой останова и нажать кнопку **Toggle Breakpoint** на панели инструментов **Debug**, а также щелчком кнопкой мыши в области полосы индикатора напротив нужной строки.

Для выполнения программы в пошаговом режиме через меню **Debug** возможны четыре варианта.

1. Осуществление последовательной шаг за шагом отладки всей программы с заходом во все действующие процедуры, написанные при составлении программы. Для этого используется команда **Debug/Step Into** (Отладка, Шаг с заходом), либо кнопка **Step Into** (Шаг с заходом) на панели инструментов **Debug** (Отладка).
2. Осуществление последовательной шаг за шагом отладки всей программы, исключая заход во все действующие процедуры из числа стандартных (встроенных) процедур. Такие процедуры выполняются целиком. Для этого используется команда **Debug/Step Over** (Отладка, Шаг с обходом), либо кнопка **Step Over** (Шаг с обходом) на панели инструментов **Debug** (Отладка).

3. Осуществление последовательной шаг за шагом отладки с завершением выполнения текущей процедуры и выходом с остановкой на строку, следующую за строкой, вызывавшей процедуру. Для этого используется команда **Debug/ Step Out** (Отладка, Шаг с выходом), либо кнопка **Step Out** (Шаг с выходом) на панели инструментов **Debug** (Отладка).
4. Выполнение программы до строки, на которой установлен курсор. Для этого используется команда **Debug/Run to Cursor** (Отладка, Выполнить до текущей позиции).

9.4. Основы языка программирования Visual Basic for Applications (VBA)

9.4.1. Алфавит VBA

Как и все языки программирования высокого уровня, Visual Basic имеет свой набор допустимых для использования символов – алфавит. Алфавит языка содержит в себе:

- прописные и строчные буквы латинского алфавита: **A, B, …, Z a, b, …, z;**
- прописные и строчные буквы русского алфавита: **А, Б, …, Я а, б, …, я;**
- арабские цифры: **0, …, 9.**

Для построения конструкций языка используются также нижеперечисленные специальные символы:

- + суммирование в математических выражениях и сцепление в строковых выражениях;
- - разность (вычитание);
- * умножение;
- / деление;
- \ целое деление;
- ^ возведение в степень;
- ( ) для составления сложных выражений;
- = оператор присваивания или знак “равно”, применяемый в логических выражениях;

- **&** сцепление подстрок в строку (конкатенация);
- **‘** комментарий;
- **_** разрыв строки исходного кода программы или как связка в идентификаторах;
- **>** больше
- **<** меньше;
- **<=** меньше или равно (не больше);
- **>=** больше или равно (не меньше);
- **<>** не равно;
- **пробел** разделитель слов (операторов) в языке.

Из вышеописанных символов, относящихся к алфавиту языка, конструируются все структуры языка. К их числу относятся: имена типов, встроенных констант, процедур, функций, операторов, объектов, свойств, методов и др. Прописные и строчные буквы не различаются. Все слова, задействованные в конструкциях языка, являются зарезервированными словами и не могут применяться для других целей.

9.4.2. Переменные и константы

В **VBA**, как и в других языках программирования высокого уровня, для хранения значений используют переменные и константы.

Переменная – именованная область памяти, используемая для хранения значения, которое можно изменить при выполнении программы.

Переменные подразделяются на простые и индексированные (переменные с индексом). Индексированными переменными являются элементы массивов.

Константа – именованная область памяти, используемая для хранения фиксированного значения, которое невозможно изменить при выполнении программы.

Имена переменных и констант должны удовлетворять следующим правилам:

- первым символом всегда должна быть буква;
- в составе имени нельзя использовать символы: **!, @, &, $, # , пробел**;

- в качестве имени нельзя использовать ключевые (зарезервированные) слова, входящие в конструкции языка VBA;
- длина имени не может быть более 255 символов;
- имя нельзя повторять в пределах области его видимости (действия).

9.4.2.1. Область действия переменных и констант

Область действия переменных и констант определяется с помощью ключевых слов: **Private, Public** и **Static.**

Private – область действия в пределах конкретного модуля. После завершения выполнения модуля память, отведенная под эти переменные, освобождается.

Public – область действия в пределах приложения.

Static – область действия в пределах конкретного модуля внешних процедур, используемых в данном модуле. После завершения выполнения модуля значения этих переменных сохраняются и могут быть использованы при повторном выполнении модуля.

Переменные и константы, в зависимости от области действия, подразделяются на **глобальные** и **локальные**.

Если переменная или константа описана внутри процедуры, то она является **локальной**, то есть она определена и может использоваться только в пределах данной процедуры.

Если переменная или константа описана вне процедуры, то она будет **глобальной**. Такая переменная или константа может быть использована в нескольких процедурах.

9.4.2.2. Объявление (описание) переменных и констант

Переменная объявляется (описывается) с помощью ключевых слов **Private, Public, Static, Dim**. Чтобы явно указать тип переменной, используется ключевое слово **As**.

Примеры описания простых переменных:

```
Private X
Public i As Integer, r As Long, c As Date
Static Строка As String
Dim Y
```

Dim Z As Single ‘Явный способ объявления переменной. Самый простой и надежный

Примеры описания индексированных переменных:

Dim Массив1() As Integer
Dim Массив2(12) As String
Dim Массив3(1 to 20) As Single
Dim Массив4(1 to 5, 1 to 7) As Byte

Константа объявляется с помощью ключевого слова **Const**. При этом можно указать ее тип, область действия и присвоить ей значение.

Синтаксис объявления:

Const <имя константы> **As** <тип> = <значение>
или
Const <имя константы> = <значение>

Если в константе явно не указан тип данных, то VBA назначает ей тип, который соответствует значению выражения.

Примеры:

Private Const q = 44,55
Public Const pi = 3,1459
Static Const QWER = 2,54
Const y = 34
Const Con As Byte = 34

Const z As Single = -3,8374E -22

Все строковые константы указываются в кавычках.

Примеры:

Const prv As String = “Язык программирования VBA”
Public Const prv = “Язык программирования VBA”

9.4.2.3. Время жизни (сохранения значений) переменных и констант

Переменные и константы, объявленные как **Private**, сохраняют свое значение только во время выполнения блока кода, в котором они определены.

Переменные и константы, объявленные как **Public**, сохраняют свое значение и вне блока кода, в котором они определены, то есть до конца выполнения программы.

Переменные и константы, объявленные как **Static**, сохраняют свое значение и между вызовами процедур.

Переменные и константы, объявленные без ключевых слов Public, Private, Static, сохраняют свое значение согласно месту их объявления (описания).

9.4.2.4. Типы переменных и констант

Таблица 9.1

Тип данных	Описание	Диапазон
Byte	целое число	От 0 до 255
Integer	целое число	От –32768 до 32767
Long	длинное целое число	От –2147483648 до 2147483647
Single	число с плавающей запятой одинарной точности	а) для отрицательных чисел: от –3,402823E38 до –1,401298E-45 б) для положительных чисел: от 1,401298E-45 до 3,402823E38
Double	число с плавающей запятой двойной точности	а) для отрицательных чисел: от –1,79769313486231E308 до –4,94065645841247E-324 б) для положительных чисел: от 4,94065645841247E-324 до 1,79769313486231E308
Currency (денежный)	число с фиксированной десятичной точкой	от –922.337.203.685.477.5808 до 922.337.203.685.477.5807
String	строка символов	от 0 до 147483647 символов
Variant	универсальный	Значения любого типа
Boolean	логический	True или False
Date	дата	от 1.01.100 года до 31.12. 9999 года
Object	объект	Ссылка на любой объект

Примечание: для дробных чисел существует экспоненциальная форма записи, например, $1{,}456 \cdot 10^{23}$, в VBA число будет выглядеть так: **1,456E23**, где **E** обозначает основание **10**, а после **E** указывается

степень числа. Запись числа с отрицательной степенью будет выглядеть: **1,456E –23**.

Вопросы для самоконтроля

1. Основные правила записи имен переменных и констант.
2. Виды переменных и констант.
3. Способы объявления (описания) переменных и констант.
4. Типы данных, используемые при описании переменных и констант.

9.4.3. Управляющие конструкции

9.4.3.1. Выражения

Выражение – это последовательность операндов, объединенных между собой знаками операций. В качестве операнда могут быть использованы: константы, переменные, функции или заключенные в круглые скобки выражения.

В зависимости от типа операндов и используемых операций выражения делятся на: **арифметические**, **логические** и **строковые (текстовые)** выражения.

Арифметическим называется выражение, содержащее операнды только арифметического (вещественного и/или целого) типа и знаки математических операций.

Знаки математических операций

Таблица 9.2

Знак	Операция	Пример	Результат
-	Знак числа (смена знака)	**-6**	**-6**
^	Возведение в степень	**3^2**	**9**
*	Умножение	**4*8**	**32**
/	Деление	**10/3**	**3,333333…**
\	Целочисленное деление	**10\3**	**3**
mod	Остаток от деления по модулю	**10 Mod 3**	**1**
+	Сложение	**3 + 2**	**5**
-	Вычитание	**7 - 5**	**2**

Примеры: y = x*10, f = (x + y)/(3 + x^2), g = Sin(x).

Выражение, содержащее хотя бы один знак логической операции, называется **логическим**.

Результатом выполнения такого выражения является логическая величина со значением либо **True** (истина), либо **False** (ложь).

Логическими операциями могут быть:

- операции отношения (сравнения). Операнды этих операций могут иметь любой тип, кроме логического типа.

Логические операции отношения

Таблица 9.3

Знак	Операция	Примеры(при значениях a = 6, b = 3)	Результат
<	Меньше	b < 7	True
>	Больше	"Вася" > "Маша"	False
<=	Меньше или равно	a + b <= 12	True
>=	Больше или равно	b^2 >= 9	True
<>	Не равно	a <> b	True
=	Равно	b =10 – a	False

- специальные логические операции. Операнды этих операций могут иметь только логический тип.

Специальные логические операции

Таблица 9.4

Знак	Операция	Пример	Значение A	Значение B	Результат
Not	Логическое отрицание	not A	True False		False True
And	Логическое умножение (И)	A and B	True True False False	True False True False	True False False False
Or	Логическое сложение (ИЛИ)	A or B	True True False False	True False True False	True True True False

Продолжение табл. 9.4

Знак	Операция	Пример	Значение A	Значение B	Результат
Xor	Исключающее ИЛИ	A xor B	True True False False	True False True False	False True True False

Строковое (текстовое) выражение может содержать операнды только символьного (текстового или строкового) типа. В языках программирования, в том числе и в **VBA**, имеется только одна строковая операция, которая носит название **конкатенация** или **слияние подстрок в строку**. Запись этой операции можно производить как знаком "+", так и знаком "**&**" (в **VBA** принято использовать знак "**&**").

Примеры строкового выражения

Таблица 9.5

Знак	Пример	Результат
&	**"Крас"&"ное"**	**"Красное"**
+	**"Чер"+"ное"**	**"Черное"**

9.4.3.2. Приоритеты операций

Таблица 9.6

Приоритет	Операция
1	Вызов функции и выражения в скобках
2	^ (возведение в степень)
3	– (смена знака числа)
4	* (умножение), / (деление), \ (целочисленное деление), **Mod** (деление, результат остаток от деления – целое число)
5	+ и – (сложение и вычитание)
6	<, >, >=, <=, <>, =
7	Not
8	And

Продолжение табл. 9.6

Приоритет	Операция
9	Or
10	Xor

Примечание. Приоритет операции "**&**" не определен, так как она не может использоваться в сочетании с другими операциями.

9.4.3.3. Встроенные функции

В VBA имеется большой набор встроенных функций, использование которых существенно упрощает программирование. Эти функции можно разделить на следующие основные категории:

- математические функции;
- функции проверки типов;
- функции преобразования типов выражений;
- функции обработки строк;
- функции времени и даты;
- функции выбора.

9.4.3.3.1. Математические функции

Таблица 9.7

Функция	Математическая интерпретация функции
Abs(x)	|x| (модуль числа **x**)
Atn(x)	Arctgx (арктангенс **x**)
Cos(x)	Cosx (косинус **x**)
Exp(x)	Экспонента (**e** в степени **x**)
Fix(x)	Отбрасывает дробную часть числа **x**
Int(x)	Округляет вещественное число **x** до целого
Log(x)	Lnx (натуральный логарифм **x**)
Rnd()	Генерирует случайное число от 0 до 1
Rnd(x)	Генерирует случайное число от 0 до **x**
Sgn(x)	Знак числа **x** (+ или —)
Sin(x)	Sinx (синус **x**)

Продолжение табл. 9.7

Функция	Математическая интерпретация функции
Sqr(x)	Корень квадратный числа **x**
Tan(x)	Tgx (тангенс **x**)

9.4.3.3.2. Функции проверки типов

Таблица 9.8

Функция	Проверка
IsArray(переменная)	Является ли переменная массивом?
IsDate(переменная)	Является ли переменная датой?
IsEmpty(переменная)	Является ли пустой ячейка рабочего листа?
IsError(переменная)	Является ли переменная кодом ошибки?
IsNull(переменная)	Является ли переменная пустым значением ноль (Null)?
IsNumeric(переменная)	Является ли переменная числовым значением?
IsObject(переменная)	Является ли переменная объектом?

9.4.3.3.3. Функции преобразования типа выражения

Таблица 9.9

Функция	Тип, в который преобразуется выражение
CBool(Выражение)	Boolean (логический)
CByte(Выражение)	Byte (байтовый)
CCur(Выражение)	Currency (денежный)
CDate(Выражение)	Date (дата)
CDbl(Выражение)	Double (вещественное число двойной точности)
CInt(Выражение)	Integer (целое число)
CLng(Выражение)	Long (длинное целое число)
CSng(Выражение)	Single (число с плавающей запятой одинарной точности)
CStr(Выражение)	String (строка)

Продолжение табл. 9.9

Функция	Тип, в который преобразуется выражение
CVar(Выражение)	Variant (вариант)

9.4.3.3.4. Функции обработки строк

Таблица 9.10

Функция	Описание
Mid(<строка>,<начало>[, <длина>])	Возвращает из строки подстроку указанной длины, начиная с заданного символа. Если длина не указана, то возвращается вся подстрока, начиная от заданного символа
Left(<строка>,<длина>)	Возвращает из заданной строки подстроку указанной длины, начиная с левого края строки
Right(<строка>,<длина>)	Возвращает из заданной строки подстроку указанной длины, начиная с правого края строки
Len(<строка>)	Возвращает длину указанной строки
InStr(<начало>,<исходная строка>,<искомая подстрока>,<тип сравнения>)	Ищет подстроку в заданной строке, начиная с указанного символа. Тип сравнения: 0 – с учетом регистра (vbBinaryCompare) 1 – без учета регистра(vbTextCompare)
Trim(строка>)	Удаляет пробелы из начала и конца заданной строки
Ltrim(<строка>)	Удаляет все пробелы из начала заданной строки
Rtrim(<строка>)	Удаляет все пробелы из конца заданной строки
Space(<количество>)	Повторяет пробел указанное количество раз
String(<количество>, <символ>)	Повторяет заданный символ указанное количество раз
Lcase(<строка>)	Преобразует в заданной строке все прописные буквы в строчные

Продолжение табл. 9.10

Функция	Описание
Ucase(<строка>)	Преобразует в заданной строке все строчные буквы в прописные
StrReverse(<строка>)	Изменяет последовательность символов в заданной строке на обратную
Asc(<Символ>)	Возвращает код символа
Chr(<Код>)	Возвращает символ по его коду
Str(<Число>)	Преобразует заданное число в строковое представление числа
Val(<Строка>)	Преобразует строку, которая представляет число, в число

9.4.3.3.5. Функции времени и даты

Таблица 9.11

Функция	Возвращаемое значение
Date	Возвращает значение типа **Variant(Date)**, содержащее текущую системную дату
Time	Возвращает значение типа **Variant(Date)**, содержащее текущее время по системным часам компьютера
Now	Возвращает значение типа **Variant(Date)**, то есть текущую дату и время по системному календарю и часам компьютера
Hour, Minute, Second	Возвращают значения типа **Variant(Date)**, содержащие целые числа, представляющие часы, минуты и секунды в значении времени. Пример: ВР=#4:35:17 PM# Час=Hour(ВР) Минута=Minute(ВР) Секунда=Second(ВР)

Продолжение табл. 9.11

Функция	Возвращаемое значение
Day, Month, Year	Возвращает значение типа Variant(Integer), содержащее целое число и представляющее день, месяц и год в значении даты. Синтаксис: Day(Дата), Month(Дата), Year(Дата)

9.4.3.3.6. Функции выбора

Таблица 9.12

Функция	Возвращаемое значение
Iif	Возвращает одну из альтернатив. Синтаксис: **Iif(expr, truepart, falsepart),** где **expr** – проверяемое значение; **truepart** – возвращаемое значение или выражение, если **expr** имеет значение **true**; **falsepart** – возвращаемое значение или выражение, если **expr** имеет значение **false**
Choose	Возвращает значение, выбранное из списка аргументов. Синтаксис: **Choose(индекс, вариант – 1, вариант – 2, …, вариант – n).** Функцию **Choose** можно использовать для выбора одного из возможных значений, представленных в виде списка
Switch	Возвращается значение, соответствующее первому истинному выражению в списке. Синтаксис: **Switch(выражение – 1, значение –1, выражение – 2, значение – 2 , …, выражение – n, значение – n).** Возвращается то значение, выражение которого принимает **True**

9.4.3.4. Операторы альтернативы (ветвления)

Как и в любом другом языке программирования, в VBA можно проверять условия и выполнять действия в соответствии с результатами

проверки условий. Для данной цели применяются следующие операторы (инструкции) принятия решения.

9.4.3.4.1. Условный оператор

IF <условие> THEN <оператор (код)>

Позволяет выполнить один или несколько операторов, если условие истинно. Можно использовать однострочный или блочный вариант. Если необходимо выполнить более одной строки кода, нужно использовать блочный вариант с оператором **End IF**.

Синтаксис:

```
IF <условие> Then <оператор (код)>
IF <условие> Then
    <блок кода>
End IF
```

Примеры:

```
IF x<10 Then z = 0
IF x>10 Then
    z = 2
    z = z + x
End IF
```

Позволяет определить два блока операторов. Первый выполняется, когда условие истинно, а второй, когда оно ложно.

Пример:

```
IF x<>0 Then
    y = Sin(x)/x
Else
    y =1
End IF
```

В блоке **IF** допускается любое количество предложений **ElseIF**, но ни одно не может находиться после предложения **Else**.

Пример ветвления по двум и более направлениям:

```
IF <условие 1> Then
    <блок кода 1>
ElseIF <условие 2> Then
    <блок кода 2>
ElseIF <условие 3> Then
    <блок кода 3>
```

```
    Else <блок кода 4
End IF
```

Пример:

```
IF x = -1.57 Then
    y = -1
ElseIF x = 0 Then
    y = 0
ElseIF x = 1.57 Then
    y =1
Else y = Sin(x)
End IF
```

9.4.3.4.2. Оператор выбора варианта

При выборе для выполнения одного из нескольких операторов (блоков операторов) целесообразно и удобно использовать инструкцию **Select Case**.

Синтаксис:

```
Select Case<переменная или выражение>
        Case <значение 1>
            <оператор (блок операторов) 1>
        Case <значение 2>
            <оператор (блок операторов) 2>
        Case <значение 3>
            <оператор (блок операторов) 3>
End Select
```

Пример применения оператора выбора в подпрограмме-функции:

```
Function PR(ByVal S As Single, ByVal P As Integer) As Single
    Select Case P
        Case 0
            PR = S*0
        Case 1
            PR = S*0.10
        Case 2
            PR = S*0.20
    End Select
End Function
```

```
Sub Krb() 'Пример программы, показывающий вызов
          подпрограммы-функции.
Dim S As Single
Dim Sum As Single
Dim P As Integer
Dim Prom As Variant
S = 5000
Do
    Prom = InputBox("Введите номер варианта <от 0 до 5>")
    IF Not IsNumeric(Prom) Then MsgBox("Повторите ввод!")
Loop Until IsNumeric(Prom)
P = Prom
Sum = PR(S,P) 'Вызов процедуры-функции. Оператор вызова PR(S,P).
MsgBox("Значение S = ") & CSng(S)
End Sub
```

Допускается вложенность инструкций **Select Case.** При этом каждой вложенной инструкции **Select Case** должна соответствовать инструкция **End Select**.

Вопросы для самоконтроля

1. В чем отличие между строчным и блочным вариантами оператора ветвления?
2. Можно ли вставлять инструкцию **Else** перед инструкцией **ElseIF** в блочном варианте оператора ветвления?
3. Сколько инструкций **ElseIF** может быть в блочном варианте оператора ветвления?
4. В чем преимущество оператора выбора варианта перед многовариантным оператором ветвления?

9.4.3.5. Операторы циклов

9.4.3.5.1. Циклы с предусловием

DO WHILE <условие> ... LOOP

Оператор **DO WHILE** <условие> … **LOOP** позволяет проверить условие перед началом цикла и выполнять цикл до тех пор, пока оно

имеет значение **True.** Как только условие цикла примет значение **False,** выполнение цикла прекратится.

Пример:

```
Dim X As Integer 'Описание переменной X целого типа.
X = 0 'Начальное значение переменной X.
DO WHILE X<=10 'Пока X<=10, цикл будет повторяться.
    X = X + 1 'Изменение значения переменной X.
LOOP 'Конец цикла.
```

Другой вариант инструкции такого цикла:

WHILE <условие>... WEND

Пример:

```
X = 0
WHILE X<12
        y = Cos(x)
        X = X + 1
WEND
```

DO UNTIL <условие> ... LOOP

Оператор **Do Until** <условие> **... Loop** позволяет проверить условие перед началом цикла и выполнять цикл до тех пор, пока оно **False**. Как только условие цикла станет равно **True**, выполнение цикла прекратится.

Пример:

```
Dim X As Integer 'Описание переменной X целого типа
X = 0 'Начальное значение переменной X
Do Until X>10 'До тех пор, пока X<=10, цикл повторяется
     X = X + 1 'Изменение значения переменной X
Loop 'Конец цикла.
```

9.4.3.5.2. Циклы с постусловием

DO ... LOOP WHILE <условие>

Если операторы цикла необходимо выполнить хотя бы раз, то нужно применять цикл с постусловием.

Инструкция **Do ... Loop While** <условие> позволяет проверить условие после выполнения операторов тела цикла.

Цикл будет повторяться до тех пор, пока выражение в условии цикла имеет значение **True**. Как только условие цикла примет значение **False,** выполнение цикла прекратится.

Пример:

```
Dim X As Integer
X = 0
Do
   X = X + 1
Loop While X<=10 'До тех пор, пока X<=10, цикл повторяется
```

DO ... LOOP UNTIL <условие>

В отличие от предыдущего этот цикл будет выполняться до тех пор, пока значение условия равно **False**.

Пример:

```
Dim X As Integer
X = 0
Do
      X = X + 1
Loop Until X>10 'Как только переменная станет больше десяти,
                выполнение цикла прекратится.
```

9.4.3.5.3. Безусловный цикл (Цикл по счетчику)

FOR ... NEXT

Цикл с определенным количеством повторений. Цикл выполняется от начального до конечного значения параметра с заданным шагом.

Пример:

```
Dim X As Integer
For X=1 To 10 Step 1 'Повторять цикл от 1 до 10 с шагом 1.
    Beep 'Звук (тело цикла).
Next X 'Конец цикла.
Exit For или Exit Do 'Досрочный выход из цикла.
```

FOR EACH ... NEXT

Цикл **For Each ... Next** предназначен для перебора всех элементов из заданного массива или набора объектов.

Пример использования цикла применительно к массивам:

```
Sub Mas6()
Dim i As Integer
Dim j As Integer
Dim x(1 To 5, 1 To 5) As Single
Dim S As Single
Worksheets("Лист1").Activate
For I =1 To 5
    For j =1 To 5
        x(i,j) = Cells(i,j)
    Next j
Next i
For Each e In x
    S = S + e
Next e
MsgBox("S = ") & CStr(s)
End Sub
```

9.4.3.5.4. Сложные (вложенные) циклы

Совокупность простых циклов, вложенных один в другой, называется сложным (вложенным) циклом. При конструировании сложных циклов необходимо руководствоваться следующими правилами:

- нельзя войти во внутренний цикл, минуя вход внешнего цикла;
- имена параметров простых циклов не должны повторяться в конструкции сложного цикла;
- простые циклы не должны пересекаться в конструкции сложного цикла, то есть окончание внешнего цикла не должно предшествовать окончанию внутреннего цикла.

Примеры:

```
For i =1 to n
    For j =1 to m
        A(i , j) = Int(Sin(j*i)*100)
    Next j
Next i
Do
    X =1
```

```
    Z = 0
    Do
       S = Int(Rnd(x)*100)
       Z = Z + S
       X = X + 1
    Loop Until X>=20
    Zsr = Z/20
Loop Until Zsr>=25
```

Вопросы для самоконтроля

1. Что такое цикл и для чего он нужен?
2. В чем основное отличие между циклами с предусловием и с постусловием?
3. В каких случаях целесообразно использовать циклы с предусловием, циклы с постусловием и циклы по счетчику?
4. В чем отличие между имеющимися вариантами циклов с предусловием?
5. В чем отличие между имеющимися вариантами циклов с постусловием?
6. Что такое сложный цикл и каковы основные правила его конструирования?

9.4.3.6. Подпрограммы-процедуры и подпрограммы-функции

Подпрограмма – это блок кода между операторами **Sub** и **End Sub** или **Function** и **end Function**.

Подпрограмма-процедура – это блок кода, заключенный между операторами **Sub** и **End Sub**. Обычно подпрограмму-процедуру принято называть процедурой.

При написании программы нужно учесть одно правило: “Внутри одной процедуры не может быть описана другая процедура”.

Синтаксис:

Sub <имя> **(ByVal** <параметр 1> **As** <тип>, **ByVal** <параметр 2> **As** <тип>, **…, ByVaL** <параметр 3>, **ByRef** <параметр N>)

<блок кода процедуры>

End Sub

В скобках указываются необходимые параметры, если параметров нет, то просто пустые скобки. Например, напишем программу, выводящую на экран окно с приветствием:

```
Sub Программа_Привет()
    MsgBox("ПРИВЕТ")
End Sub
Sub qwer(ByVal x As Single, y As Single, ByRef S As Single)
    Dim Z As Single
    Dim P As Single
    Z = Sin(2*x + 3*y)
    P = Cos(x^2 + y^3)
    S = Z + P
End Sub
```

Параметры, указанные в скобках заголовка процедуры, называются **формальными.**

Параметры, значения которых передаются из программы (другой процедуры) в процедуру, называются **параметрами-переменными**.

Параметры, значения которых передаются из процедуры в программу (другую процедуру), называются **параметрами-значениями**.

Параметры, указанные в списке оператора вызова процедуры, называются **фактическими параметрами.**

Ключевые слова **ByVal** и **ByRef** определяют способ передачи значений параметров.

ByVal указывает, что аргумент передается по значению.

ByRef указывает, что аргумент передается по ссылке.

Значения параметров-переменных, передаваемых по способу **ByVal**, не могут изменяться в теле процедуры во время ее выполнения, то есть последние значения переменных в программе сохраняются неизменными во время выполнения процедуры.

Значения параметров, передаваемых по способу **ByRef**, изменяют значения соответствующих переменных программы.

Вызов процедуры из другой процедуры можно произвести несколькими способами.

Первый способ: **<Имя процедуры> <Список фактических параметров>**. Список должен соответствовать списку, заданному в заголовке процедуры, по количеству и типу.

Пример: **qwer x,y,s** ‘оператор вызова процедуры.

Если требуется использовать несколько процедур с одинаковыми названиями, расположенными в разных модулях, то при их вызове перед именем процедуры через точку необходимо указывать имя модуля, в котором расположена процедура.

Синтаксис:

<Имя модуля>.<Имя процедуры> <Список фактических параметров>.

Второй способ вызова процедуры производится с помощью инструкции **Call**.

Синтаксис:

Call <Имя процедуры> (<Список фактических параметров>).

В отличие от первого способа здесь список фактических параметров заключается в скобки.

Пример: **Call qwer(x,y,s)**

Подпрограмма-функция – это блок кода, заключенный между операторами **Function и End Function.** Она выполняет какую-то операцию, но при этом обязательно возвращает какое-нибудь значение. Значение возвращается через имя функции.

Синтаксис:

Function <имя функции> **(ByVal** <параметр> **As** <тип>**) As** <Тип>
<код функции>
End Function

Пример:

```
Function f(ByVal x As Single) As Single
        f = Sin(x^2) + Cos(3*x)
End Function
```

Оператор вызова функции состоит из имени функции и списка фактических параметров, заключенных в скобки.

Пример: y = **f(x)** ‘Оператор вызова функции.

При необходимости можно указать область определения процедуры или функции:

Private Sub Программа_Привет() – закрытая процедура. Возможен вызов из модуля, где она находится, то есть доступна для других процедур только того модуля, в котором она описана.

Public Sub Программа_Привет() – открытая процедура. Возможен вызов из любого модуля, то есть доступна для всех других процедур во всех модулях.

Static Sub Программа_Привет() – указывает, что значения локальных переменных процедуры сохраняются в промежутках времени между вызовами этой процедуры.

Private Function f(ByVal x As Single, ByVal y As Single) As Single – закрытая функция. Возможен вызов из модуля, где она находится.

Public Function f(ByVal x As Single, ByVal y As Single) As Single – открытая функция. Возможен вызов из любого модуля.

9.4.3.7. Процедуры ввода-вывода через диалоговые окна

В сокращенном варианте инструкции ввода-вывода имеют вид:

InputBox(<"сообщение">)
MsgBox(<"сообщение">)

Пример:

```
Sub Krb()
Dim S As Integer
Dim Sum As Single
Dim P As Integer
Dim Prom As Variant
S = 5000
Do
    Prom = InputBox("Введите значение P=")
    IF Not IsNumeric(Prom) Then MsgBox("Повторите ввод!")
Loop Until IsNumeric(Prom)
P = Prom
Sum = S + S*0.01*P
MsgBox("Результат S = ") & CSng(Sum
End Sub
```

Вопросы для самоконтроля

1. Что понимается под параметрами-переменными и параметрами-значениями?
2. Какое различие между понятиями "формальные" и "фактические" параметры?
3. Способы передачи параметров-переменных и параметров значений.

4. Могут ли быть разными имена формальных и фактических параметров?
5. В какой последовательности должны располагаться имена параметров в операторах вызова процедуры и функции?
6. Способы вызова процедуры.

9.4.3.8. Массивы

Массив – набор однотипных переменных с одним именем, каждая из которых называется элементом массива и имеет свой номер (индекс).

Массивы могут быть: одномерные (для нумерации элементов используется один индекс), двумерные (для нумерации элементов используются два индекса: номер строки, номер столбца) и N-мерные. Число измерений может достигать 60.

Кроме того, массивы подразделяются на статические и динамические.

9.4.3.8.1. Статические массивы

Статическим называется массив с заранее известным количеством элементов.

Синтаксис описания (объявления) статического массива:

Dim <Имя массива>(<верхняя граница>) As <Тип>

По умолчанию нижняя граница равна 0.

Dim <Имя массива>(<Нижняя граница> To <Верхняя граница>) As <Тип>

Примеры:

```
Dim a(10) As Single 'Одномерный массив с начальной границей, равной 0.
Dim S(3 To 5) As String 'Одномерный массив с явными границами.
Dim Z(1 To 3, 1 To 5) As Byte 'Двумерный массив.
```

Для задания по умолчанию нижней границы массива, равной 1, используется команда **Option Base 1**, которая вводится в начале модуля.

Примеры:

```
Option Base 1
```

```
Sub Mas1()
Dim a(5) As Integer
Dim i As Integer, k As Integer
WorkSheets("Лист1").Select 'Выбрать Лист1 из семейства листов.
Cells.Clear 'Очистить ячейки рабочего листа.
K = 2
For i =1 To 5
    a(i) = Int(Rnd(i)*100) 'Формирование массива a(i).
    Cells(k , i + 1) = a(i) 'Вывод массива a(i) на рабочий лист.
Next i
End Sub
Sub Mas2()
Dim a(1 to 5) As Integer, k As Integer
Dim i As Integer
WorkSheets("Лист1").Select
Cells.Clear
K = 2
For i = 1 To 5 'Формирование массива a(i).
    a(i) = Int(Rnd(i)*100)
Next i
For i =1 To 5 'Вывод массива a(i) на рабочий лист.
    Cells(k , i + 1) = a(i)
Next i
End Sub
Sub Dmas1()
Dim i As Integer
Dim j As Integer
Dim a2(1 to 3, 1 to 5) As Integer
WorkSheets("Лист1").Select
Cells.Clear
For i =1 To 3 'Формирование массива a2(i , j).
    For j =1 To 5
        a2(i , j) = Int(Rnd(i*j)*100)
    Next j
Next i
```

```
For i = 1 To 3 'Вывод массива a2(i , j) на рабочий лист.
    For j = 1 To 5
        Cells(i + 1 , j + 1) = a2(i , j)
    Next j
Next i
End Sub
Sub Dmas2()
Dim a(1 to 5) As Integer, k As Integer
Dim i As Integer, prom As Variant
WorkSheets("Лист1").Select
Cells.Clear
K = 2
For i =1 To 5 'Ввод элементов массива с клавиатуры.
    Do
       Prom = InputBox("Введите элемент a("& CInt(i) & ") =")
       If Not IsNumeric(prom) Then MsgBox("Повторите ввод!")
    Loop Until IsNumeric(prom)
    A(i) = prom
Next i
For i =1 To 5 'Вывод массива a(i) на рабочий лист.
    Cells(k,2) = "a("& CInt(i) & ") = "
    Cells(k,3) = a(i)
    k = k + 1
Next i
End Sub
```

9.4.3.8.2. Динамические массивы

Динамическим называется массив, размерность которого определяется в ходе выполнения программы.

Синтаксис описания массива: **Dim <Имя массива>() As <Тип>**

Размерность массива устанавливается и изменяется с помощью оператора:**ReDim <Имя массива>(<размерность>)**

Для установки и изменения размерности массива без потери его содержимого применяется оператор:

ReDim Preserve<Имя массива>(<размерность>)

Для определения параметров динамического массива используются функции:

LBound(<Имя>[,<Размерность>]) 'Возвращает нижнюю границу указанной размерности массива, которую следует задавать равной 1.

UBound(<Имя>[,<Размерность>]) 'Возвращает верхнюю границу указанной размерности массива, которую следует задавать равной 2. Если размерность не указана, то подразумевается значение, равное 1.

Пример:

```
Sub Mas4()
Dim a() As Integer
Dim i As Integer, k As Integer, j As Integer, n Integer
Dim prom As Variant
WorkSheets("Лист1").Select
Cells.Clear
k = 2
i = 0
Do
    Prom =InputBox("Введите количество элементов N =)
    If Not IsNumeric(prom) Then MsgBox("Повторите ввод!")
Loop Until IsNumeric(prom)
N = prom
ReDim a(1 to n) As Integer 'Устанавливается фактическая
                           размерность массива a(i).
Do'Формирование массива a(i).
  ReDim Preserve a(1 to n) As Integer 'Это другой вариант
  установки размерности массива с сохранением значений его
  элементов.
  a(i) = Int(Rnd(i)*100)
  i = i + 1
Loop Until i = N
For j =1 To i 'Вывод массива a(j) на рабочий лист.
     Cells(k,j + 1) = a(j)
Next j
End Sub
```

Array(<Список аргументов>) 'Создает массив типа Variant. Список аргументов представляет разделенный запятыми список значений, присваиваемых элементам массива.

Пример:

```
Dim День As Variant
День = Array("Пн", "Вт", "Ср", "Чт", "Пт", "Сб")
```

IsArray(<Имя переменной>) 'Функция используется для проверки значений переменных типа **Variant**, содержащих массивы. Возвращает **True**, если переменная содержит массив, и **False**, если переменная не содержит массив (не является массивом).

Erase(<Список массивов>) 'Эта инструкция повторно инициализирует элементы статических массивов и освобождает память, отведенную для динамических массивов. Список массивов представляет имена очищаемых массивов, разделенных запятой. В статических массивах их элементам вместо чисел или строк фиксированной длины присваиваются значения, равные 0, а для строк переменной длины присваивается значение пустой строки. В массивах типа **Variant** каждому элементу присваивается значение **Empty**.

Вопросы для самоконтроля

1. Варианты описания статических массивов.
2. Как установить начальную границу размерности статического массива, равную 1, если при описании массива начальная граница задается равной 0?
3. Какое предельное количество измерений многомерного статического массива?
4. В каких случаях возникает потребность в применении динамического массива?
5. Как установить или изменить размерность динамического массива?
6. Как изменить размерность динамического массива без потери имеющихся значений его элементов?

9.5. Основные объекты VBA

К числу основных объектов VBA, которые описываются в этом разделе, относятся: рабочая книга (Workbook) и семейство рабочих книг (Workbooks), рабочий лист (Worksheet) и семейство рабочих листов (Worksheets), диапазон ячеек или ячейка (Range).

После объекта, обычно через точку ".", указывается свойство или метод. Иногда, чтобы добраться до определенного объекта, нужно пройти иерархию вышестоящих объектов.

Пример:

Workbooks(“Книга1.xls”).Worksheets(“Лист1”).Activate

Свойства и методы, которые обеспечивают ссылку на нужный объект в иерархии объектов, называются семействами (наборами).

Семейство **WorkBooks**("Книга1") ‘Обеспечивает доступ к рабочей книге. В скобках указывается имя книги.

Семейство **WorkSheets**("Лист1") ‘Обеспечивает доступ к рабочему листу. В скобках указывается имя листа.

Семейство **Range**("диапазон") ‘Обеспечивает доступ к диапазону ячеек или к ячейке. В скобках указывается диапазон ячеек или имя ячейки.

Семейство **Cells(№ строки, № столбца)** ‘Обеспечивает доступ к ячейке. В скобках указываются координаты ячейки.

Примеры:

WorkBooks("Книга1")
WorkSheets("Лист1")
Range("A1")
Range("A1:B10")
Cells(2,3)
Cells(k,i+1)

9.5.1. Объект Workbook и семейство Workbooks

В иерархии Excel объект Workbook (рабочая книга) идет сразу после объекта Application и представляет файл рабочей книги. Рабочая книга хранится либо в файлах формата XLS (стандартная рабочая книга) или XLA (полностью откомпилированное приложение). Свойства и методы рабочей книги позволяют работать с файлами. Объект входит в семейство (набор) Workbooks.

Ссылку на объект можно получить через свойства:

Workbooks(<Индекс>) ‘Возвращает объект по индексу в наборе;

Workbooks(“<Имя>”) ‘Возвращает объект по имени в наборе;

ActiveWorkbook ‘Возвращает ссылку на активную книгу в момент выполнения команды;

ThisWorkbooks ‘Возвращает ссылку на книгу, в которой находится текст исполняемого модуля.

9.5.1.1 Свойства объекта Workbook и семейства Workbooks

ActiveSheet 'Возвращает активный лист книги. Для получения имени листа используется свойство **Name** объекта **Sheet.**

Примеры:

MsgBox("Имя активного листа ") & ActiveSheet.Name

или

MsgBox ActiveWorkbook.ActiveSheet.Name 'Выводит имя активного рабочего листа в диалоговом окне.

ActiveDialog 'Возвращает активное диалоговое окно.

ActiveChart 'Возвращает активную диаграмму.

Sheets 'Возвращает семейство всех листов книги.

Worksheets 'Возвращает семейство всех рабочих листов книги.

Пример:

```
Sub xjfchd()
For Each s In ActiveWorkbook.Sheets
    MsgBox s.Name
Next s
End Sub
```

Charts 'Возвращает семейство всех диаграмм книги, которые не внедрены в рабочие листы.

Count 'Возвращает число объектов семейства Workbooks (количество открытых приложением книг).

FullName 'Возвращает полное имя рабочей книги.

Пример:

MsgBox ActiveWorkbook.FullName

Name 'Возвращает имя активной рабочей книги.

Пример:

MsgBox ActiveWorkbook.Name

Path 'Возвращает путь к файлу рабочей книги.

Пример:

MsgBox ActiveWorkbook.Path

9.5.1.2. Методы объекта Workbook и семейства Workbooks

Activate 'Активизирует рабочую книгу так, что ее первый рабочий лист становится активным.

Пример:
WorkBooks("Книга1").Activate
или
Workbook.Activate
Close 'Закрытие рабочей книги.
Close.SaveChanges FileName – Закрывает книгу.
SaveChanges 'Сохраняет изменения в рабочей книге, если ответ **True**.
'Не сохраняет изменения в рабочей книге, если ответ **False**.
FileName 'Строка, указывающая имя файла, в котором будет сохранена рабочая книга.

Пример:
WorkBooks("Книга1").Close
WorkBooks("Книга1").Close.SaveChanges="Книга2"
NewWindow 'Открывает указанную книгу в новом окне.
Пример:
WorkBooks("Книга1").NewWindow
Save 'Сохраняет изменения в рабочей книге.
Пример:
WorkBooks("Книга1").Save
SaveAs FileName 'Сохраняет книгу под другим именем (в другом файле).
SaveAsCopy 'Сохраняет рабочую книгу в другом файле, оставляя ее в памяти с прежним именем.

Примеры:
WorkBooks("Книга1").SaveAs FileName="kdjf.xls"
ActiveBook.SaveAsCopy FileName="Моя книга"
Open FileName 'Открывает рабочую книгу с именем, указанным в параметре **FileName**.

Пример:
Workbooks.Open "Книга1.xls"

9.5.1.3. События объекта Workbook и семейства Workbooks

Таблица 9.13

Событие	Когда возникает событие
BeforeClose	При закрытии рабочей книги

Продолжение табл. 9.13

Событие	Когда возникает событие
BeforePrint	Перед печатью рабочей книги
BeforeSave	Перед сохранением рабочей книги
NewSheet	При добавлении нового листа
Open	При открытии рабочей книги
SheetActivate	При активизации рабочего листа

Вопросы для самоконтроля

1. Свойства объекта **Workbook** и семейства **Workbooks** для возвращения (вывода) активного рабочего листа, семейства всех листов книги, активной диаграммы, числа объектов семейства.
2. Методы объекта **Workbook** и семейства **Workbooks** для открытия, сохранения и закрытия рабочей книги.

9.5.2. Объект Worksheet и семейство Worksheets

В иерархии Excel объект Worksheet идет сразу после объекта Workbook и представляет рабочий лист книги и входит в семейство (набор) **Worksheets.**

Ссылку на объект можно получить через команды:

Worksheets(Index) 'Возвращает ссылку на объект по индексу в наборе, в качестве индекса может выступать имя листа или его номер в наборе.

Примеры:

Worksheets("Лист1").Activate
Worksheets(1).Activate
Activesheet 'Возвращает ссылку на активный лист.

Пример:

Activesheet.Range("a1")=1

9.5.2.1. Свойства объекта Worksheet и семейства Worksheets

Name 'Возвращает имя рабочего листа.

Пример:

Worksheets(1).Name="Итоги"

ActiveCell ‘Возвращает активную ячейку активного рабочего листа.

Cells ‘Возвращает ссылку на диапазон ячеек листа.

Cells(<строка>,<столбец>) ‘Возвращает ссылку на ячейку с указанными координатами.

Columns(<столбец>) ‘Возвращает ссылку на столбец. В качестве параметра может быть имя или номер столбца.

Пример:

Worksheets(1).Columns(“a”)=1

или

Worksheets(1).Columns(1)=1

Rows(<строка>) ‘Возвращает ссылку на строку. В качестве параметра может быть номер строки.

Пример:

Worksheets(1).Rows(1)=1

Range(<Диапазон ячеек>) ‘Возвращает ссылку на указанный диапазон ячеек.

UsedRange ‘Возвращает ссылку на используемый диапазон листа.

Пример:

Worksheets(“Лист1”).UsedRange.Value=1

Count ‘Возвращает количество листов в книге.

Visible ‘Определяет отображение рабочего листа в книге.

Допустимые значения:

- **True** ‘рабочий лист выводится на экран;
- **False** ‘рабочий лист невидим (скрыт), но его можно отобразить на экране с помощью последовательности команд: **Формат, Лист, Отобразить (Format, Sheet, Show)**;
- **xlVeryHidden** ‘рабочий лист скрыт и его можно отобразить на экране только программно.

Примеры:

```
Sub Пусто()
Worksheets(“Лист3”).Visible=False
End Sub
Sub Открыто()
Worksheets(“Лист3”).Visible=True
```

```
End Sub
Sub Пусто1()
Worksheets("Лист4").Visible=xlVeryHidden
End Sub
Sub Открыто1()
Worksheets("Лист4").Visible=True
End Sub
```

UsedRange 'Возвращает диапазон, то есть объект Range, содержащий данные.

Пример:

```
Worksheets(1).UsedRange.Clear  'Очищается диапазон первого рабочего листа с данными.
```

9.5.2.2. Методы объекта Worksheet и семейства Worksheets

Activate 'Активизирует рабочий лист.

Evaluate 'Преобразует выражение в объект или в значение. Используется при вводе формул (функций) и ячеек из диалоговых окон.

Пример:

```
Sub Ввод_формул()
Dim Значение As Single
Dim Fx As String
Ячейка=InputBox("Введите имя ячейки") 'Ввод адреса ячейки в диалоговое окно.
Значение=Evaluate(Ячейка).Value  'Считывание значения из ячейки.
MsgBox Cstr(Ячейка)&"="&CSng(Значение) 'Вывод значения в диалоговое окно.
Fx=InputBox("Введите формулу (функцию)") 'Ввод функции в диалоговое окно.
Значение=Evaluate(Fx) 'Вычисление значения функции.
MsgBox CStr(Ячейка)&"="&CSng(Значение) 'Вывод значения в диалоговом окне.
End Sub
```

Вопросы для самоконтроля

1. Свойства объекта **Worksheet** и семейства **Worksheets** для возвращения ссылки на строку, на столбец, на диапазон ячеек, на ячейку, на используемый диапазон ячеек.
2. Свойство объекта **Worksheet** и семейства **Worksheets** для отображения рабочего листа на экране.
3. Метод объекта **Worksheet** и семейства **Worksheets** для ввода формул и ячеек через диалоговое окно.

9.5.3. Объект Range

В иерархии **Excel** объект **Range** (диапазон) идет сразу после объекта **Worksheet** и является одним из ключевых объектов VBA. Он не входит в состав никакого семейства объектов.

Объект **Range** описывает диапазон ячеек рабочего листа и возвращает свойства и методы. При работе с объектом **Range** имеется три способа ссылки на ячейки рабочего листа: относительная адресация (начало координат, задающее нумерацию строк и столбцов, связывается с объектом, вызвавшим **Range**), абсолютная и смешанная адресация.

Абсолютная адресация

Формат A1	Признаком абсолютной адресации является символ “**$**”, предшествующий имени: строки (абсолютной адресации на строку ⇒ **A$12**), столбца (абсолютной адресации на столбец ⇒ **$A12**), ячейки (абсолютной адресации на ячейку ⇒ **A12**).
Формат R1C1	Указывается смещение по отношению к активной ячейке. Смещение приводится в квадратных скобках, причем знак указывает на направление смещения. (**R2C3 ⇒ R[1]C[-1] ⇒ R3C2**)

Относительная адресация

Формат A1	Имя ячейки состоит из имени столбца (их 256 от A до IV) и номера строки (от 1 до 16384). Пример: **A1, C2**
Формат R1C1	Адресация задается индексом строки и индексом столбца. Примеры: **R1C1, R2C2, R1C3**

Адресация ячейки рабочего листа является лишь частью полного адреса ячейки, который в общем случае включает имя рабочего листа и адрес книги. При задании полного адреса за именем листа следует знак "!", а адрес книги заключается в скобки.

Примеры:

A1 'Относительная ссылка на ячейку **A1** активного рабочего листа.

Лист2!A1 'Относительная ссылка на ячейку **A1** рабочего листа **Лист2** активной книги.

[ВсепроВсе.xls]Лист2!A1 'Относительная ссылка на ячейку **A1** рабочего листа **Лист2** книги **ВсепроВсе.xls** текущего рабочего каталога.

Если в диапазоне указываются только имена столбцов или строк, то объект **Range** задает диапазон, состоящий из указанных столбцов или строк.

Примеры:

Range("A:C") 'Задает диапазон столбцов A, B, C.

Range("2:4") 'Задает диапазон строк 2, 3, 4.

Так как ячейка является частным случаем диапазона, то объект **Range** позволяет также работать и с ней. Альтернативным способом работы с ячейкой является объект **Cells** (ячейки).

Пример: **Range("A2") или Cells(1, 2)** 'Ячейка **A2** описывается как объект.В свою очередь объект **Cells**, вкладываясь в **Range**, также позволяет записывать диапазон в альтернативном виде.

Пример: **Range("A2:C3") или Range(Cells(1, 2), Cells(3, 3))**

9.5.3.1. Свойства объекта Range при работе с данными

Formula 'Устанавливает формулу в ячейке. Формула задается в виде строки.

FormulaArray 'Устанавливает формулу массива ячеек. Формула задается в виде строки. В качестве ссылок на ячейки используется формат A1.

Формула массива – это формула, которая в качестве исходных данных использует диапазон ячеек и возвращает одно или несколько значений.

Пример:

```
Sub Prima1()
  With Worksheets("Лист1")
    For i = 1 To 3
      For j = 1 To 3
        .Cells(i,j) = Int(Rnd(i*j)*100)
      Next j
    Next i
    .Range("D1:F3").FormulaArray="=MINVERSE(a1:c3)"
  End With
End Sub
```

FormulaR1C1 'Устанавливает формулу в ячейке. Формула задается в строковом виде и в формате **R1C1**.

Пример:

```
Sub Prima1()
With Worksheets("Лист1")
  For i = 1 To 3
      For j = 1 To 3
        .Cells(i,j) = Int(Rnd(i*j)*100)
      Next j
  Next i
  .Range("D1:F3").FormulaR1C1="=MINVERSE(R1C1:R3C3)"
End With
End Sub
```

HasArray 'Свойство возвращает **True**, если указанная ячейка является частью массива.

Массивом на рабочем листе является именованный диапазон ячеек.

HasFormula 'Свойство возвращает **True**, если в указанной ячейке установлена формула.

Примеры:

MsgBox Worksheets(1).Range("a2").HasArray => True или **False**

MsgBox Worksheets(1).Range("a3"). HasFormula =>True или **False**

Text 'Возвращает содержимое ячейки в виде строки. Используется только для чтения

Пример:

MsgBox Worksheets(1).Range("a2").Text => 1234

Value 'Возвращает значение из ячейки или устанавливает значение в ячейку (в ячейки)

Примеры:

x=Range("c1").Value 'Значение из ячейки **c1** присваивается переменной **x**

Range("a1:b4").Value=12 'В диапазон ячеек **a1:b4** устанавливается число 12

Команда With

With используется для указания текущего объекта. Далее внутри команды можно указывать, начиная с точки, только свойства и методы при обращении к текущему объекту.

Синтаксис:

```
With <объект>
    .<свойства и методы>
End With
```

Пример:

```
Sub qwe()
Dim a As Single
Dim b As Single
With Worksheets("Лист1")
    a =.Range("A1").Value
    b =.Range("B1").Value
      .Range("C1").Value = a + b
      .Range("D1").Formula="=A1 + B1"
End With
End Sub
```

Команда Set

Set – предназначена для закрепления объекта за переменной. Переменная должна быть типа Object или с типом объекта, который за ней будет закреплен.

Примеры:

```
Sub Prima1()
Dim Lst As Object
Set Lst=Workbooks("книга1.xls").Worksheets("лист1")
'За переменной Lst закрепляется рабочий лист 1 рабочей книги 1.
Lst.Cells(1,1)=10 'В ячейку a1 рабочего листа 1 устанавливается
                  значение 10.
End Sub
Sub Prima2()
Dim Lst As Worksheet 'Переменная Lst с типом объекта
                     WorkSheet.
Set Lst=Workbooks("книга1.xls").Worksheets("лист1")
Lst.Cells(1,1)=10
End Sub
```

9.5.3.2. Методы объекта Range

```
Activate 'Активизирует указанный диапазон ячеек.
AddComment <текст примечания> 'Добавляет примечание к
                               ячейке (контекстное
                               меню|добавить примечание).
AutoFill <источник> 'Производит автозаполнение диапазона
                     ячеек данными из указанных ячеек.
```

Пример:

```
Sub Prima3()
Worksheets(1).Activate
For i = 1 To 10
    Cells(i,1)=i
Next i
Range("b1").Formula="=sin(a1)"
Range("b1").AutoFill Range("b1:b10")
End Sub
Clear 'Очищает указанный диапазон ячеек.
```

ClearContents ‘Очищает формулы и значения, содержащиеся в ячейках, представляемых объектом Range, то есть очищается только содержимое ячеек, сохраняя их форматирование.

ClearFormats ‘Удаляет все форматирование ячеек, сохраняя неизменными хранящиеся в них данные.

Sort Key1=<ячейка>,Order1=<порядок>;
Orientation=<направление> ‘Производит сортировку указанного диапазона ячеек.

Параметры:

Key1 ‘Ключевое поле (колонка или столбец), по которому будет производиться сортировка.

Order1 ‘Указывает порядок сортировки. Может иметь два значания:

- xlAscending ‘По алфавиту;
- xlDescending ‘Наоборот.

Orientation ‘Указывает направление сортировки:

- xlSortRows ‘Сортировка данных в строке;
- xlSortColumns ‘Сортировка данных в столбце.

Пример:

```
Sub bmv()
Worksheets(1).Activate
Range(“A1:D10”).Sort Key1:=Range(“B1”),Order1:=xlDescending
End Sub
```

Вопросы для самоконтроля

1. Свойства объекта Range, обеспечивающие установку формулы в ячейку в форматах A1 и R1C1, а также установку формулы массива ячеек.
2. Свойства объекта Range, обеспечивающие установку в ячейку какого-либо значения или извлечение его из ячейки.
3. Метод объекта Range, обеспечивающий сортировку указанного диапазона ячеек, его параметры и их значения.

9.5.4. Оформление рабочего листа

Внешнее оформление ячеек и данных в них производится с помощью свойств объектов: **Range**, **Interior**, **Font**, **Border** и свойства **NumberFormat**.

9.5.4.1. Оформление текста в ячейке

Format – свойство объекта, задающее формат вывода данных.
Пример:
Range("a1").Format="шаблон"
Могут применяться следующие шаблоны.

Таблица 9.14

Шаблон	Расшифровка шаблона
"#,##0.000"	Формат для вывода чисел с точностью до 3 знака после запятой. Специальные символы: "#" – необязательное число; "0" – обязательное число; ". " – разделитель тысячных; "," – разделитель между целой и дробной частью
"#,##0.00$"	"$" – обозначение национальной валюты
"dd/mm/yy"	Вывод даты в формате 11.01.99, где: "d" – день; "m" – месяц; "yy" – две последние цифры года
"dd/mm/yyyy"	Вывод даты в формате 01.01.1999
"d/mm"	Вывод даты в формате 1.01
"d mmm yy"	1 янв. 99, где mmm – сокращение месяца
"d mmmm yy"	1 января 99, где mmmm – полное название месяца
"h:mm"	Время в формате 1:23, где h – часы, mm – минуты
"h:mm:ss"	Время в формате 1:23:00, где ss – секунды
"0.00%"	Процентный формат, к примеру, 3 – это 300 %
"@"	Текстовый формат
"0.00E+00"	Экспоненциальный формат числа

Примеры:
Range("C1:C6").Select

```
Selection.NumberFormat="0,0"
Range("D1:D6").Select
Selection.NumberFormat="dd/mm/yy"
Range("E1:E6").Select
Selection.NumberFormat="h:mm:ss"
Range("F1:F6").Select
Selection.NumberFormat="@"
```

Выравнивание текста в ячейках задается свойствами **HorizontalAlignment, VerticalAlignment.**

HorizontalAlignment – определяет выравнивание по горизонтали.

Для выравнивания текста в ячейках по горизонтали могут применяться следующие константы.

Таблица 9.15

Константа	Направление выравнивания
xlLeft	По левому краю
xlRight	По правому краю
xlCenter	По центру
xlGeneral	По значению
xlJustify	По ширине
XlFill	С заполнением
xlCenterAcrossSelection	По центру выделения

VerticalAlignment – определяет выравнивание текста по высоте.

Для выравнивания текста в ячейках по высоте могут применяться следующие константы.

Таблица 9.16

Константа	Направление выравнивания
xlTop	По верхнему краю
xlBottom	По нижнему краю
xlCenter	По центру

WrapText – определяет перенос по словам текста внутри ячейки:

- **True** – разрешить перенос;
- **False** – запретить.

MergeCells – объединение ячеек:

- **True** – ячейки объединены;
- **False** – ячейки не объединены.

ShrinkToFit – включает автоматический подбор ширины ячейки по содержимому:

- **True** – включено;
- **False** – выключено.

Orientation – определяет ориентацию текста в ячейке.

Для ориентации текста в ячейках по высоте могут применяться следующие константы.

Таблица 9.17

Константа	Направление ориентации
xlHorizontal	Горизонтальная ориентация
xlVertical	Вертикальная ориентация
-90 до 90	Угол поворота текста в градусах

Примеры:

```
Range("B2:B5").Select
With Selection
    .HorizontalAlignment=xlRight
    .VerticalAlignment=xlBottom
    .WrapText=False
    ,Orientation=0
    .MergeCells=False
End With
Range("C2:C5").Select
With Selection
    .HorizontalAlignment=xlLeft
    .VerticalAlignment=xlBottom
    .WrapText=False
    .Orientation=0
    .MergeCells=False
End With
```

Параметры шрифта определяются через свойства объекта **Font**:

Name – имя шрифта;

Size – размер шрифта;

FontStyle – начертание. Принимает значения: обычный, курсив, полужирный, полужирный курсив;

ColorIndex – определяет цвет шрифта. В качестве значения ставится номер цвета;

xlAutomatic (константа) – цвет по умолчанию;

Color – задает произвольный цвет. Для задания цвета можно использовать функцию **RGB** (красный, зеленый, синий);

Underline – подчеркивание текста в ячейке.

Для подчеркивания текста в ячейках могут применяться следующие константы.

Таблица 9.18

Константа	Способ подчеркивания
xlUnderlineStyleNone	Нет подчеркивания
xlUnderlineStyleSingle	Одинарное по значению
xlUnderlineStyleDouble	Двойное по значению
xlUnderlineStyleSingleAccounting	Одинарное по ширине ячейки
xlUnderlineStyleDoubleAccounting	Двойное по ширине ячейки

Пример:

```
Range("d3:f3).Select
With Selection.Font
    .Name="Arial Cur"
    .Size=14
    .Underline=xlUnderlineStyleNone
    .ColorIndex=xlAutomatic
End With
Selection.Font.Bold=True
Selection.Font.ColorIndex=3
```

9.5.4.2. Оформление границ ячейки

Оформление границ ячеек задается через свойства объекта **Border**. Для диапазона ячеек границы делятся на внешние и внутренние. Доступ к определенным границам производится через набор **Borders** (граница).

Границы могут задаваться следующими константами.

Таблица 9.19

Константа	Граница
xlEdgeBottom	Внешняя нижняя
xlEdgeTop	Внешняя верхняя
xlEdgeLeft	Внешняя левая
xlEdgeRight	Внешняя правая

Продолжение табл. 9.19

xlInsideHorizontal	Внутренняя горизонтальная
xlInsideVertical	Внутренняя вертикальная
xlDiagonalDown	От верхнего правого до нижнего левого угла
xlDiagonalUp	От нижнего левого до верхнего правого угла

Параметры границы определяют следующие свойства.

LineStyle – задает стиль линии.

Для задания стиля линии могут применяться следующие константы.

Таблица 9.20

Константа	Стиль линии
xlContinuous	Сплошная линия
xlDash	Прерывистая линия
xlDashDot	Пунктирная линия
xlDashDotDot	Двойная пунктирная линия
xlDot	Точечная линия
xlDouble	Двойная линия
xlLineStyleNone	Нет линии

Weight – определяет толщину линии.

Для определения толщины линии могут применяться следующие константы.

Таблица 9.21

Константа	Толщина линии
xlHairline	Сверхтонкая
xlThin	Тонкая
xlMedium	Средняя
xlThick	Жирная

Цвет границы задается свойствами **ColorIndex** и **Color**.

Примеры:

```
Range("c1:f3").Select
Selection.Borders(xlDiagonalDown).LineStyle=xlNone
Selection.Borders(xlDiagonalUp).LineStyle=xlNone
With Selection.Borders(xlEdgeLeft)
    .LineStyle=xlContinuous
```

```
        .Weight=xlThin
        .ColorIndex=xlAutomatic
    End With
    With Selection.Borders(xlInsideHorizontal)
        .LineStyle=xlContinuous
        .Weight=xlThin
        .ColorIndex=xlAutomatic
    End With
With Selection.Borders(xlInsideVertical)
        .LineStyle=xlContinuous
        .Weight=xlThin
        .ColorIndex=xlAutomatic
End With
```

9.5.4.3. Заливка ячейки

Цвет и узор, которыми заполняется ячейка, задаются через свойства объекта **Interior**. Цвет заливки задается свойствами **Color** или **ColorIndex**.

Pattern – задает узор заливки ячейки.

Для задания узора в ячейках могут применяться следующие константы.

Таблица 9.22

Константа	Узор заливки
xlSolid	сплошной
xlGray75	75 % – серый
xlGray50	50 % – серый
xlGray25	25 % – серый
xlGray16	12,5 %– серый
xlGray8	6,25 %– серый
xlHorizontal	горизонтальный штриховой
xlVertical	вертикальный штриховой
xlDown	перевернутый диагональный штриховой
XlUp	диагональный штриховой
xlChecker	диагональный клетчатый
xlSemiGray75	толстый диагональный клетчатый
xlLightHorizontal	тонкий горизонтальный штриховой

Продолжение табл. 9.22

xlLightVertical	тонкий вертикальный штриховой
xlLightDown	тонкий перевернутый диагональный
xlLightUp	тонкий диагональный
XlGrid	тонкий горизонтальный клетчатый
xlCrissCross	тонкий диагональный клетчатый

Цвет узора заливки задается свойствами **PatternColorIndex** и **PatternColor**.

Пример:

```
Range("c1:f3").Select
With Selection.Interior
    .ColorIndex=6
    .Pattern=xlSolid
End With
```

Заключение

В современном мире количество компьютеров удваивается в среднем каждые три года. 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, один раз в пять-семь лет меняется база стандартов, интерфейсов и протоколов. Предметная область дисциплины "Информатика" изменяется чрезвычайно динамично, что является кардинальным отличием ее от других технических дисциплин. Поэтому для эффективного использования вычислительной техники от специалистов (пользователей) требуется достаточно высокий уровень базовых знаний и практических навыков.

Предлагаемое учебное пособие позволит получить базовые знания и приобрести практические навыки в ходе лабораторного практикума.

Список рекомендуемой литературы

1. Завгордний, В. И. Комплексная защита информации в компьютерных системах. – М. : Логос, 2001. – 264 с.

2. Симонович, С. В. Информатика. Базовый курс. – СПб. : Питер, 2005. – 640 с.

3. Гарнаев, А. Ю. Самоучитель VBA .– СПб. : БХВ – Петербург, 2001. – 512 с.

4. Васильев, А. VBA в Office 2000 : учеб. курс /А. Васильев, А. Андреев. – СПб. : Питер, 2001. – 432 с.

5. Кузьменко, В. Г. VBA 2000 : самоучитель. – М. : ЗАО "Издательство Бином", 2000. – 407 с.

Федеральное агентство по образованию

**Государственное образовательное учреждение
высшего профессионального образования
"Кузбасский государственный технический университет"**

Л. С. Таганов В. Г. Левин

ИНФОРМАТИКА

Учебное пособие

Кемерово 2006

УДК 681.3.06

Рецензенты:

Старший преподаватель кафедры вычислительной математики Кемеровского государственного университета М. Р. Екимова

Зав. кафедрой технологии автоматизированной обработки информации Кемеровского государственного университета культуры и искусств кандидат педагогических наук, профессор Н. И. Колкова

Информатика : учеб. пособие / Л. С. Таганов, В. Г. Левин : ГУ КузГТУ. – Кемерово, 2006. – 155 с.

ISBN 5-89070-553-9

Подготовлено по дисциплине "Информатика". В пособии изложены в систематизированном виде теоретические основы, обеспечивающие единую методическую базу для изучения дисциплины. Содержит необходимую информацию в объеме учебного курса вуза в соответствии с государственным образовательным стандартом.

Предназначено для студентов специальности 170100 (150402) "Горные машины и оборудование", а также может быть весьма полезным для других специальностей.

Печатается по решению редакционно-издательского совета ГУ КузГТУ.

УДК 681.3.06

ISBN 5-89070-553-9

Таганов Леонид Степанович
Левин Владимир Григорьевич

ИНФОРМАТИКА

Учебное пособие

Редактор З. М. Савина

Подписано в печать 21.06.2006. Формат 60x84/16.
Бумага офсетная. Отпечатано на ризографе.
Уч.-изд. л. 9,7.
Тираж 500 экз. Заказ 42
ГУ КузГТУ, 650026, Кемерово, ул. Весенняя, 28.
Типография ГУ КузГТУ, 650099, Кемерово, ул. Д. Бедного, 4.

Оглавление