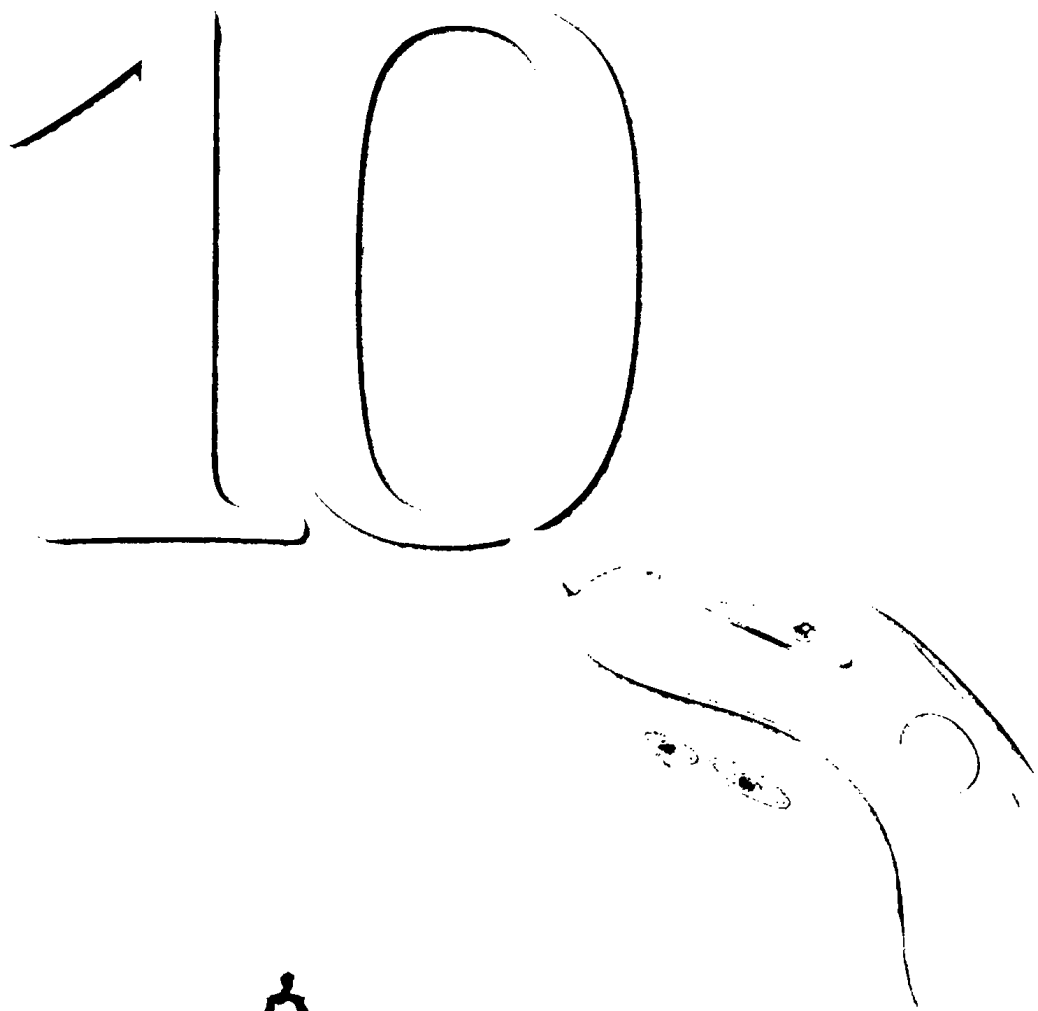


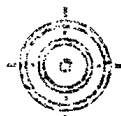
Н. Д. Угринович



ИНФОРМАТИКА

и ИКТ





ПРОФИЛЬНЫЙ
УРОВЕНЬ

Н. Д. Угринович

ИНФОРМАТИКА и ИКТ

Учебник для 10 класса

3-е издание, исправленное

Допущено

Министерством образования и науки Российской Федерации
к использованию в образовательном процессе
в образовательных учреждениях,
реализующих образовательные программы
общего образования



Москва
БИНОМ. Лаборатория знаний
2008

УДК 004.9
ББК 32.97
У27

Угринович Н. Д.
У27 Информатика и ИКТ. Профильный уровень : учебник для 10 класса / Н. Д. Угринович. — 3-е изд., испр. — М. : БИНОМ. Лаборатория знаний, 2008. — 387 с. : ил.

ISBN 978-5-94774-828-4

Учебник по курсу «Информатика и ИКТ» ориентирован на информационно-технологический и физико-математический профили обучения в общеобразовательных учреждениях. Учебник полностью соответствует новому образовательному стандарту и примерной программе профильного обучения, утвержденным Министерством образования и науки РФ. Содержание учебника соответствует программе вступительных экзаменов по информатике в вузы, и он может быть использован для подготовки к экзаменам.

Рассматриваются архитектура компьютера и методы защиты информации, понятие «информация» и системы счисления, основы логики и логические основы компьютера, а также объектно-ориентированное программирование на четырех языках: Visual Basic, Delphi, Visual C# и Visual J#. Все необходимое для преподавания курса программное обеспечение содержится на CD-дисках, которые входят в состав методического пособия для учителя.

УДК 004.9
ББК 32.97

По вопросам приобретения обращаться:

«БИНОМ. Лаборатория знаний»
(499) 157-52-72, e-mail: Lbz@aha.ru
<http://www.Lbz.ru>

ISBN 978-5-94774-828-4 © Угринович Н. Д., 2008
© БИНОМ. Лаборатория знаний, 2008

Оглавление

Рекомендации по использованию учебника	7
Глава 1. Архитектура компьютера и защита информации	9
1.1. Магистрально-модульный принцип построения компьютера	10
1.2. Процессор и оперативная память	16
1.2.1. Процессор	16
1.2.2. Оперативная память	23
1.3. Внешняя (долговременная) память	29
1.3.1. Магнитная память	29
1.3.2. Оптическая память	31
1.3.3. Флэш-память	34
1.4. Файл и файловые системы	37
1.4.1. Логическая структура носителя информации	37
1.4.2. Файл	44
1.4.3. Иерархическая файловая система	49
1.5. Операционная система	54
1.5.1. Назначение и состав операционной системы	54
1.5.2. Загрузка операционной системы	59
1.6. Защита информации от вредоносных программ	61
1.6.1. Вредоносные программы и антивирусные программы	61
1.6.2. Компьютерные вирусы и защита от них	64
1.6.3. Сетевые черви и защита от них	71
1.6.4. Троянские программы и защита от них	75
1.6.5. Рекламные и шпионские программы и защита от них	79
1.6.6. Спам и защита от него	84
1.6.7. Хакерские утилиты и защита от них	88
Глава 2. Информация. Системы счисления	94
2.1. Понятие «информация» в науках о неживой и живой природе, обществе и технике	94
2.1.1. Информация в физике	94
2.1.2. Информация в биологии	99
2.1.3. Информация в общественных науках	101
2.1.4. Информация в кибернетике	103


2.2. Количество информации как мера уменьшения неопределенности знания	105
2.3. Алфавитный подход к определению количества информации	110
2.4. Формула Шеннона	111
2.5. Кодирование текстовой, графической и звуковой информации	115
2.6. Хранение информации	119
2.7. Кодирование числовой информации. Системы счисления	121
2.7.1. Непозиционные системы счисления	121
2.7.2. Позиционные системы счисления	125
2.8. Перевод чисел из одной системы счисления в другую	129
2.8.1. Перевод целых чисел из десятичной системы в двоичную, восьмеричную и шестнадцатеричную	129
2.8.2. Перевод дробей из десятичной системы в двоичную, восьмеричную и шестнадцатеричную	133
2.8.3. Перевод чисел из двоичной системы в восьмеричную и шестнадцатеричную и обратно	136
2.9. Арифметические операции в позиционных системах счисления	140
2.10. Представление чисел в компьютере	143
2.10.1. Представление чисел в формате с фиксированной запятой	143
2.10.2. Представление чисел в формате с плавающей запятой	148
Глава. 3. Основы логики и логические основы компьютера	151
3.1. Формы мышления	151
3.2. Алгебра логики	156
3.2.1. Логическое умножение, сложение и отрицание	156
3.2.2. Логические выражения	164
3.2.3. Логические функции	169
3.2.4. Логические законы и правила преобразования логических выражений	174
3.2.5. Решение логических задач	177
3.3. Логические основы устройства компьютера	180
3.3.1. Базовые логические элементы	180
3.3.2. Сумматор двоичных чисел	183
3.3.3. Триггер	188

Алгоритмизация и основы объектно-ориентированного программирования	190
4.1. Алгоритм и кодирование основных алгоритмических структур	190
4.1.1. Алгоритм и его свойства.	190
4.1.2. Алгоритмические структуры «ветвление» и «выбор»	192
4.1.3. Алгоритмическая структура «цикл»	195
4.2. История развития языков программирования.	197
4.3. Введение в объектно-ориентированное визуальное программирование	201
4.3.1. Объекты: свойства и методы.	201
4.3.2. События	203
4.3.3. Проекты и приложения	205
4.4. Система объектно-ориентированного программирования Microsoft Visual Studio .NET	208
4.4.1. Платформа .NET Framework	208
4.4.2. Интегрированная среда разработки языков Visual Basic .NET, Visual C# и Visual J#	212
4.5. Система объектно-ориентированного программирования Turbo Delphi	216
4.6. Переменные	221
4.7. Графический интерфейс	226
4.8. Пространство имен .NET.	237
4.9. Процедуры и функции	243
4.9.1. Процедуры	243
4.9.2. Функции	248
4.10. Итерация и рекурсия	252
4.11. Делегаты.	257
4.12. Алгоритмы перевода чисел и их кодирование на языках объектно-ориентированного программирования	262
4.12.1. Алгоритм перевода целых чисел.	262
4.12.2. Алгоритм перевода дробных чисел.	272
4.13. Графика в объектно-ориентированных языках программирования	281
4.13.1. Графика в языках программирования Visual Basic .NET, Visual C# и Visual J#	281
4.13.2. Графика в языке программирования Turbo Delphi	300

4.13.3. Компьютерная и математическая системы координат	311
4.13.4. Анимация.	316
4.14. Модульный принцип построения решений (групп) и проектов.	320
4.15. Чтение и запись данных в файлы	329
4.16. Массивы	332
4.16.1. Заполнение массивов	332
4.16.2. Поиск элемента в массивах	341
4.16.3. Сортировка числовых массивов	344
4.16.4. Сортировка строковых массивов.	354
Глава. 5. Практические задания для тематического и итогового контроля	359
Тема «Архитектура компьютера и защита информации»	359
Тема «Информация. Системы счисления»	367
Тема «Логика и логические основы компьютера»	368
Тема «Алгоритмизация и основы объектно-ориентированного программирования»	369
Ответы и решения	375
Плакаты	377
Таблица 1. Процессоры	377
Таблица 2. Элементы управления для ввода и вывода данных	379
Таблица 3. Элементы управления для организации интерактивного диалога	380
Таблица 4. Элементы управления, не отображаемые на форме в процессе выполнения проекта.	381
Таблица 5. Типы переменных	382
Таблица 6. Встроенные функции (методы).	383
Таблица 7. Кодирование алгоритмических структур «ветвление» и «выбор»	384
Таблица 8. Кодирование алгоритмической структуры «цикл»	385
Словарь компьютерных терминов	386


Рекомендации по использованию учебника

1. Учебник «Информатика и ИКТ-10. Профильный уровень» обеспечивает изучение профильного курса «Информатика и ИКТ» в 10 классе на профильном уровне в соответствии с образовательным стандартом. Учебник входит в состав учебно-программного комплекса, включающего:
 - учебники для основной школы: «Информатика и ИКТ-8» и «Информатика и ИКТ-9»;
 - учебники по профильному курсу для старшей школы на базовом уровне: «Информатика и ИКТ-10. Базовый уровень» и «Информатика и ИКТ-11. Базовый уровень»;
 - учебники по профильному курсу для старшей школы на профильном уровне: «Информатика и ИКТ-10. Профильный уровень» и «Информатика и ИКТ-11. Профильный уровень»;
 - учебное пособие и CD-ROM по элективному курсу для старшей школы «Исследование информационных моделей»;
 - методическое пособие для учителей «Преподавание курса «Информатика и ИКТ» в основной и старшей школе»;
 - Windows-CD, содержащий программную поддержку курса, готовые компьютерные проекты, рассмотренные в учебниках, тесты и методические материалы для учителей;
 - Visual Studio-CD (выпускается по лицензии корпорации Microsoft), содержащий дистрибутивы систем объектно-ориентированного программирования языков *Visual Basic .NET*, *Visual C#* и *Visual J#*;
 - Linux-DVD (выпускается по лицензии компании Alt-Linux), содержащий операционную систему Linux и программную поддержку курса;
 - TurboDelphi-CD (выпускается по лицензии компании Borland), содержащий систему объектно-ориентированного программирования TurboDelphi.
2. В практических работах указано необходимое для их выполнения программное обеспечение, которое обозначается значком приложения.
3. В учебнике используются ссылки на внешние источники информации (учебники, CD-диски и Интернет), а также на параграфы и пункты самого учебника.
4. В тексте пособия приняты следующие выделения:
 - Шрифтом Arial выделены имена программ, файлов и Интернет-адреса.

- Шрифтом Courier New выделены программы на языках программирования.
 - *Курсивом* выделены названия диалоговых окон, пунктов меню и элементов управления (текстовых полей, кнопок и т. д.) графического интерфейса операционных систем и приложений.
 - **Полужирным шрифтом** выделены важные термины и понятия.
 - Подчеркиванием выделены термины, краткое объяснение которых содержится в Словаре компьютерных терминов.
5. Важная информация выделена в тексте восклицательным знаком, а формулы — цифровым обозначением.
 6. Абзацы, содержащие дополнительную интересную информацию, выделены значком .
 7. Проекты создаются в системах объектно-ориентированного программирования Visual Basic .Net, Visual C#, Visual J# и Turbo Delphi и хранятся на CD-дисках Windows-CD и TurboDelphi-CD.
 8. Дополнительные материалы и интерактивные тесты для проверки усвоения материала находятся в Интернете по адресу: <http://iit.metodist.ru>

Глава 1

Архитектура компьютера и защита информации

Windows-CD 

В процессе изучения данной темы рекомендуется установить программное обеспечение в операционной системе Windows:

- программы тестирования компьютера SiSoft Sandra, CPU-Z, EasyTune, SIV;



- файловый менеджер Total Commander;



- архиватор 7-Zip;



- программу записи CD- и DVD-дисков: DeepBurner;



- браузеры Internet Explorer и SeaMonkey;



- антивирусные программы: avast! и Antivir Personal Edition;



- программу удаления рекламных и шпионских программ: Ad-Aware;



- программу восстановления системы: CCleaner;



- межсетевой экран: Outpost Firewall.



1.1. Магистрально-модульный принцип построения компьютера

В основу архитектуры современных персональных компьютеров положен магистрально-модульный принцип. Этот принцип предусматривает построение компьютера из функциональных блоков, взаимодействующих посредством общего канала (каналов) — шины. В сочетании с открытой (общезвестной) архитектурой это позволяет потребителю собирать машину нужной конфигурации.

Магистраль включает в себя три многопроводные шины: шину данных, шину адреса и шину управления, которые представляют собой многопроводные линии (рис. 1.1). К магистрали подключаются процессор и оперативная память, а также периферийные устройства ввода, вывода и хранения информации, которые обмениваются информацией в форме последовательностей нулей и единиц, реализованных в виде электрических импульсов.

Шина данных. По этой шине данные передаются между различными устройствами. Например, считанные из оперативной памяти данные могут быть переданы процессору для обработки, а затем полученные данные могут быть отправлены обратно в оперативную память для хранения. Таким образом, данные по шине данных могут передаваться от устройства к устройству через области оперативной памяти.

Разрядность шины данных определяется разрядностью процессора, т. е. количеством двоичных разрядов, которые процессор обрабатывает за один такт. Разрядность процессоров постоянно увеличивалась по мере развития компьютерной техники и в настоящее время составляет 64 бита.

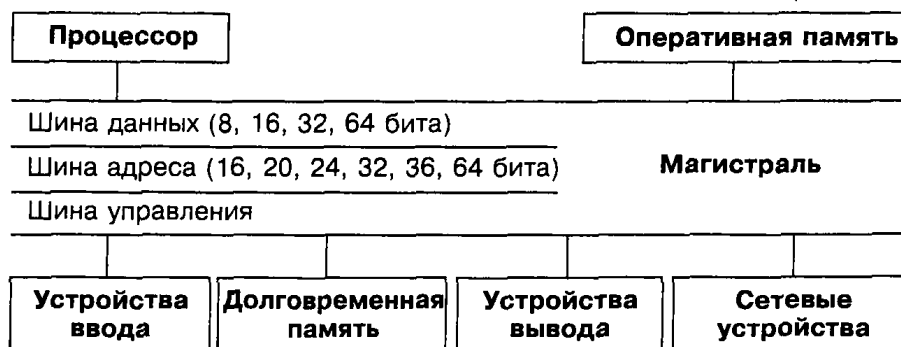


Рис. 1.1. Магистрально-модульное устройство компьютера

Шина адреса. Выбор устройства или ячейки памяти, куда пересылаются или откуда считываются данные по шине данных, производит процессор. Каждое устройство или ячейка оперативной памяти имеет свой адрес. Адрес передается по адресной шине, причем сигналы по ней передаются в одном направлении от процессора к оперативной памяти и устройствам (однонаправленная шина).

Разрядность шины адреса определяет объем адресуемой памяти, т. е. количество ячеек оперативной памяти, которые могут иметь уникальные адреса. Количество адресуемых ячеек памяти можно рассчитать по формуле:

$$N = 2^I, \text{ где } I \text{ — разрядность шины адреса.}$$

Разрядность шины адреса постоянно увеличивалась и в процессорах Pentium Extreme Edition составляет 64 бита. Таким образом, количество адресуемых ячеек памяти в таких процессорах равно:

$$N = 2^{64} \text{ ячеек.}$$

Шина управления. По шине управления передаются сигналы, определяющие характер обмена информацией по магистрали. Сигналы управления определяют, какую операцию — считывание или запись информации из памяти нужно производить, синхронизируют обмен информацией между устройствами и т. д.

Системная плата. Важнейшим аппаратным компонентом компьютера является системная плата (рис. 1.2, 1.3). На системной плате реализована магистраль обмена информацией, имеются разъемы для установки процессора, слоты для уста-

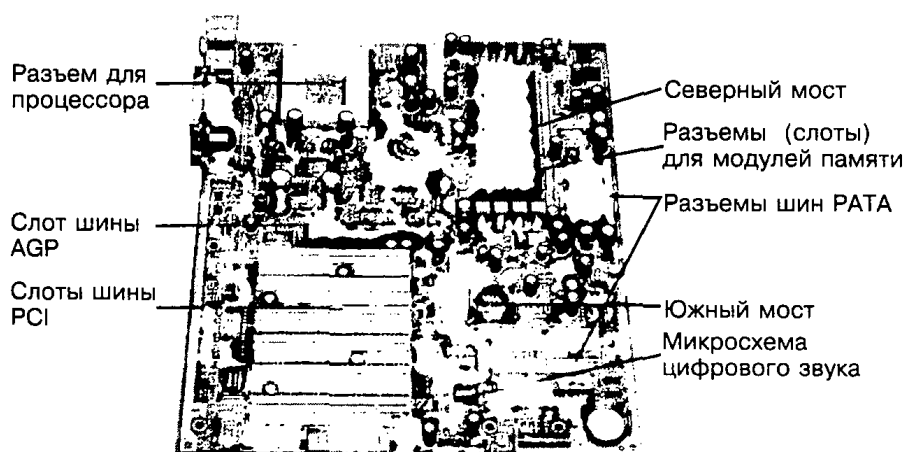


Рис. 1.2. Системная плата

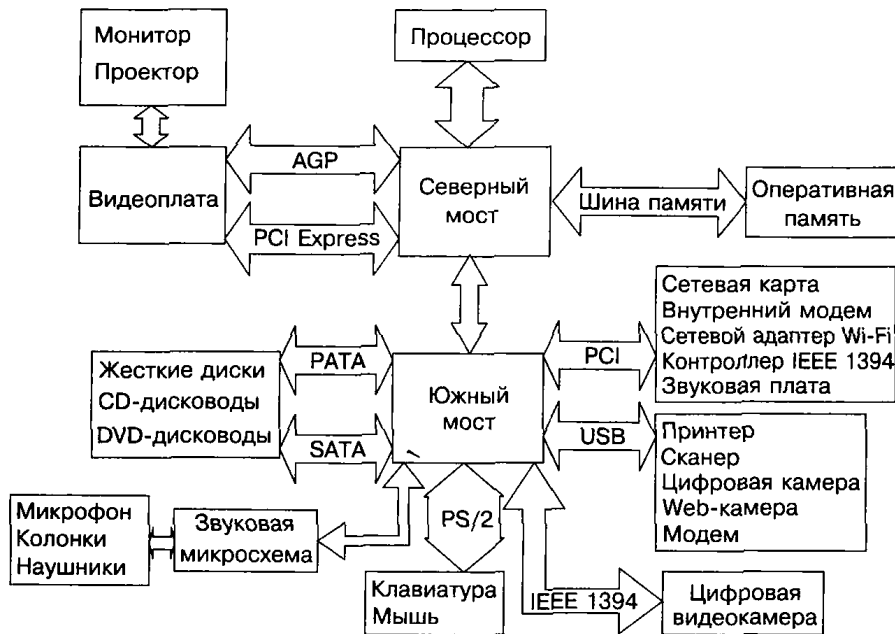


Рис. 1.3. Логическая схема системной платы

новки оперативной памяти, а также контроллеров внешних устройств.

Пропускная способность. Быстродействие устройства зависит от тактовой частоты тактового генератора (обычно измеряется в мегагерцах — МГц) и разрядности, т. е. количества битов данных, которые устройство может обрабатывать или передавать одновременно (измеряется в битах). Дополнительно в устройствах используется внутреннее умножение частоты с разными коэффициентами.

Соответственно, скорость передачи данных (пропускная способность) соединяющих эти устройства шин также должна различаться. Пропускная способность шины данных (измеряется в бит/с) равна произведению разрядности шины (измеряется в битах) и частоты шины (измеряется в Гц = 1/с):

$$\begin{aligned} \text{Пропускная способность шины} &= \\ &= \text{Разрядность шины} \times \text{Частота шины}. \end{aligned}$$

Северный и южный мосты. Для согласования тактовой частоты и разрядности устройств на системной плате устанавливаются специальные микросхемы (их набор называется чипсетом), включающие в себя контроллер оперативной памяти и видеоплаты (так называемый северный мост) и контроллер периферийных устройств (южный мост).

Частота процессора. Северный мост обеспечивает обмен данными с процессором, оперативной памятью и видеопамятью. Частота процессора в несколько раз больше, чем базовая частота магистрали (иногда ее называют шиной FSB от англ. FrontSide Bus). Например, в наиболее быстрых компьютерах (2006 год) частота шины FSB составляет 266 МГц, коэффициент умножения частоты 14, следовательно, частота процессора $266 \text{ МГц} \times 14 \approx 3,7 \text{ ГГц}$.

Системная шина. Между северным мостом и процессором данные передаются по системной шине с частотой, которая в четыре раза больше частоты шины FSB. Таким образом, процессор может получать и передавать данные с частотой $266 \text{ МГц} \times 4 = 1064 \text{ МГц}$. Так как разрядность системной шины равна разрядности процессора и составляет 64 бита, то пропускная способность системной шины равна:

$$64 \text{ бит} \times 1064 \text{ МГц} = 68\,096 \text{ Мбит/с} \approx \\ \approx 66 \text{ Гбит/с} \approx 8 \text{ Гбайт/с.}$$

Шина памяти. Обмен данными между процессором и оперативной памятью производится по шине памяти, частота которой может быть меньше, чем частота шины процессора. Например, частота шины памяти может составлять 533 МГц, т. е. оперативная память получает данные в два раза реже, чем процессор. Так как разрядность шины памяти равна разрядности процессора и составляет 64 бита, то пропускная способность шины памяти равна:

$$64 \text{ бит} \times 533 \text{ МГц} = 34\,112 \text{ Мбит/с} \approx \\ \approx 33 \text{ Гбит/с} \approx 4 \text{ Гбайт/с.}$$

Шины AGP и PCI Express. По мере усложнения графики приложений требования к быстродействию шины, связывающей видеопамять с процессором и оперативной памятью, возрастают. Для подключения видеоплаты к северному мосту может использоваться 32-битовая шина AGP (Accelerated Graphic Port — ускоренный графический порт). Эта шина первоначально передавала данные с частотой 66 МГц, в настоящее время возможно использование шины AGP×8, частота которой $66 \text{ МГц} \times 8 = 528 \text{ МГц}$. В этом случае пропускная способность шины видеоданных составляет:

$$32 \text{ бит} \times 528 \text{ МГц} = 16\,896 \text{ Мбит/с} = \\ = 16,5 \text{ Гбит/с} \approx 2 \text{ Гбайт/с.}$$

В настоящее время для подключения видеоплаты к северному мосту все большее распространение получает шина PCI

Express (Peripheral Component Interconnect bus Express — ускоренная шина взаимодействия периферийных устройств). Пропускная способность этой шины значительно выше пропускной способности PCI и AGP.

К видеоплате с помощью аналогового разъема VGA (Video Graphics Array — графический видеоадаптер) или цифрового разъема DVI (Digital Visual Interface — цифровой видеоинтерфейс) подключается электронно-лучевой или жидко-кристаллический монитор или проектор.

Шина PCI. К северному мосту подключается по специальной шине южный мост, к которому, в свою очередь, подключаются периферийные устройства. Шина PCI (Peripheral Component Interconnect bus — шина взаимодействия периферийных устройств) обеспечивает обмен информацией с контроллерами периферийных устройств, которые устанавливаются в слоты расширения системной платы.

Наиболее часто эта шина используется для установки устройств доступа к локальной сети (сетевая карта), глобальной сети Интернет (встроенный модем) и беспроводной сети (сетевой адаптер Wi-Fi, произносится «вай-фай», сокр. от Wireless Fidelity — протокол и стандарт на оборудование для широкополосной радиосвязи).

Разрядность шины PCI может составлять 32 бита или 64 бита, а частота — 33 МГц или 66 МГц. Таким образом, максимальная пропускная способность шины PCI составляет:

$$64 \text{ бит} \times 66 \text{ МГц} = 4224 \text{ Мбит/с} = 528 \text{ Мбайт/с.}$$

Шина IEEE 1394 (другие названия FireWire, i-Link). Последовательная высокоскоростная шина, предназначенная для обмена цифровой информацией между компьютером и цифровыми устройствами (цифровыми видеокамерами, DVD-плеерами и др.) без потери качества изображения и звука. (Эту функцию может выполнять также контроллер IEEE 1394, который подключается к шине PCI.) Скорость передачи данных по этой шине может достигать 200 Мбайт/с и более.

Шина ATA. Устройства внешней памяти (жесткие диски, CD- и DVD-дисководы) подключаются к южному мосту по шине ATA (англ. Advanced Technology Attachment — шина подключения накопителей). Ранее использовалась параллельная шина PATA (англ. Parallel ATA), скорость передачи данных по которой может достигать 133 Мбайт/с. В настоящее время широкое распространение получила последовательная шина SATA (англ. Serial ATA), скорость передачи данных по которой может достигать 300 Мбайт/с.

Шина USB. Для подключения принтеров, сканеров, цифровых камер и других периферийных устройств обычно используется шина USB (Universal Serial Bus — универсальная последовательная шина). Эта шина обладает пропускной способностью до 60 Мбайт/с и обеспечивает подключение к компьютеру одновременно нескольких периферийных устройств (принтер, сканер, цифровая камера, Web-камера, модем и др.).

Клавиатура и мышь. Клавиатура и мышь подключаются с помощью порта PS/2 или шины USB (в том числе с помощью беспроводного адаптера).

Звук. К южному мосту может подключаться интегрированная в системную плату микросхема, которая обеспечивает обработку цифрового звука (эту функцию может выполнять также звуковая плата, которая подключается к шине PCI). С помощью аудиоразъемов к системной плате могут подключаться микрофон, колонки или наушники.

Практическое задание «Тестирование системной платы». С помощью программы тестирования EasyTune определить у вашего компьютера частоту шины FSB, коэффициент умножения частоты процессора, частоту процессора, частоту шины памяти, частоту шин AGP и PCI Express и частоту шины PCI. Вычислить пропускную способность шины памяти.



Тестирование системной платы

1. Запустить программу тестирования EasyTune5 и выбрать вкладку *Advanced Mode*.

На панели отобразятся параметры компьютера, в данном случае (рис. 1.4):

- частота шины FSB — 133 МГц;
- коэффициент умножения частоты процессора — 20;
- частота процессора — 2680 МГц;
- частота шины памяти — 266 МГц;
- частота шин AGP и PCI Express — 66 МГц;
- частота шины PCI — 33 МГц.

2. Пропускная способность шины памяти = 64 бита × × 266 МГц = 17024 Мбит/с = 2128 Мбайт/с ≈ 2 Гбайт/с.

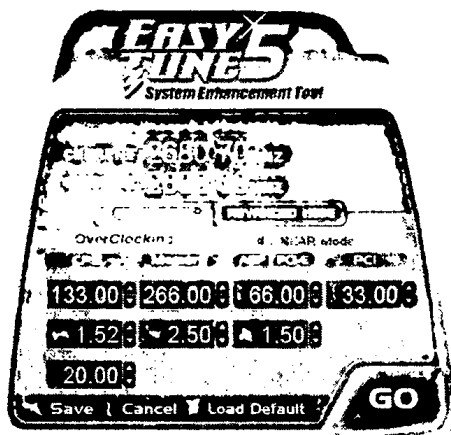


Рис. 1.4. Тестирование системной платы

Контрольные вопросы

1. Что определяют разрядности шины данных и шины адреса?
2. Почему различаются скорости передачи данных по шине FSB, системной шине, шине памяти, шинам AGP и PCI Express и шине PCI?

Компьютерный практикум

Windows-CD



- 1.1. С помощью программы тестирования компьютера EasyTune выполнить практическое задание «Тестирование системной платы».



1.2. Процессор и оперативная память

1.2.1. Процессор

Логическая схема процессора (рис. 1.5). Процессор является центральным устройством компьютера и выполняет команды программы, которая хранится в оперативной памяти. Команда программы поступает в процессор по шине данных и декодируется, т. е. определяется, какие действия необходимо выполнить и какие данные для этого требуются.

Данные запрашиваются из оперативной памяти, для этого по шине адреса передаются их адреса, а по шине управления — сигнал на считывание. Считанные данные передаются в процессор по шине данных.

Декодированная команда и данные передаются в АЛУ (арифметико-логическое устройство), где отдельно обрабатываются целочисленные данные, и отдельно — данные в форме чисел с плавающей запятой.

2.10. Представление чисел в компьютере

Результаты обработки передаются по шине данных в оперативную память, одновременно по шине адреса передаются адреса ячеек памяти, куда данные необходимо записать, а по шине управления передается сигнал на запись.

Быстродействие процессора существенно больше быстродействия оперативной памяти, поэтому процессор часть времени простаивает в ожидании данных. Чтобы этого не происходило, в современные процессоры встроена более быстрая, чем оперативная, кэш-память.

Кэш-память разделена на два уровня. В кэш-память 2-го уровня (большую по объему и менее быстродействующую) считывается из оперативной памяти очередная порция команд и данных. Кэш-память 1-го уровня (меньшая по объему, но более быстродействующая) разделена на две части, в одну часть считываются наиболее нужные процессору данные, а в другую часть — наиболее нужные процессору команды.

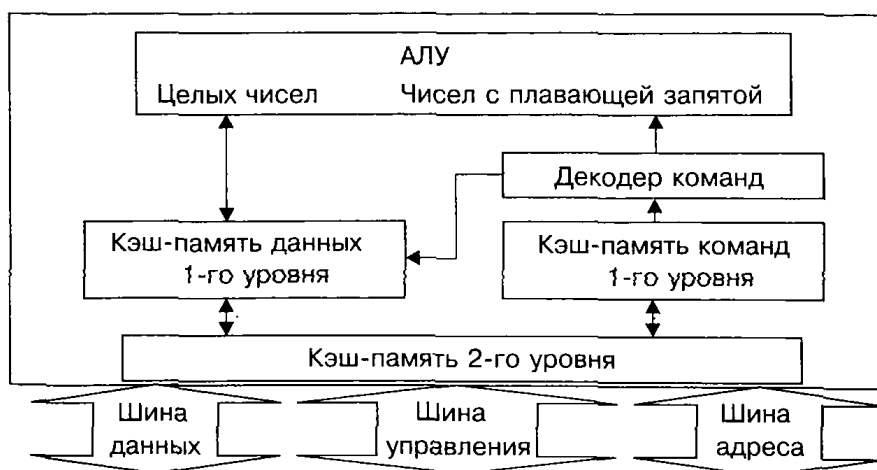


Рис. 1.5. Упрощенная логическая схема одноядерного процессора

Практическое задание «Определение объемов кэш-памяти процессора». С помощью программы тестирования CPU-Z определить объемы кэш-памяти 1-го и 2-го уровней вашего процессора.



Определение объемов кэш-памяти процессора

1. Запустить программу тестирования CPU-Z и выбрать вкладку *Cache* (рис. 1.6).

На панели отобразятся параметры кэш-памяти, в данном случае:

- кэш-память данных 1-го уровня — 8 Кбайт;
- кэш-память команд 1-го уровня — 12 Кбайт;
- кэш-память 2-го уровня — 128 Кбайт.

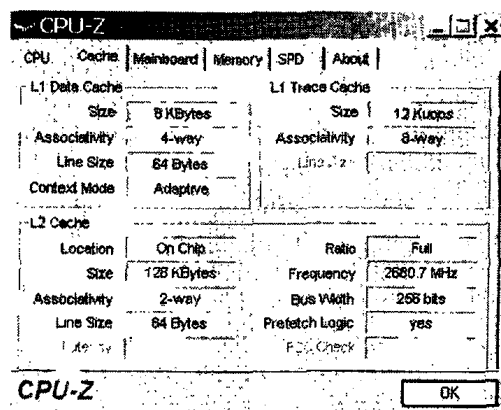


Рис. 1.6. Определение объемов кэш-памяти процессора

Технология. Процессор является большой (в смысле количества размещенных на ней элементов) полупроводниковой микросхемой. Процессоры создаются на основе кремниевых подложек, которые вырезаются из слитка чистого кремния. Кремний является полупроводником, который в разных условиях может вести себя и как проводник электрического тока, и как изолятор.

Физика-11

Сначала на кремниевой подложке под воздействием высокой температуры и кислорода формируется слой диоксида кремния. Этот процесс очень похож на возникновение ржавчины на железе, погруженном в воду. Разница заключается в том, что слой диоксида кремния формируется на подложке

гораздо быстрее и не виден невооруженным глазом (из-за того, что очень тонок).

Химия-9

Затем кремниевая подложка покрывается фотослоем. В процессе **фотолитографии** ультрафиолетовое излучение, проходя сквозь маску (которая выполняет функцию шаблона), формирует на подложке рисунок электрической схемы. Засвеченные участки фотослоя и диоксида кремния полностью удаляются с помощью растворителя. С помощью процесса ионной имплантации области кремниевой подложки, обработанные ультрафиолетом, бомбардируются ионами различных примесей. Ионы проникают в подложку, обеспечивая необходимую электрическую проводимость этих областей. В результате на кремниевой подложке создается электрическая схема.

После этого наносится слой поликристаллического кремния и еще один фотослой. Наложение новых слоев с последующим вытравливанием схемы осуществляется несколько раз, при этом для межслойных соединений в слоях оставляются «окна». Эти «окна» заполняются атомами металла (алюминия или меди). Так устанавливаются связи между несколькими десятками слоев, формирующими сложную трехмерную электрическую схему.

На конечном этапе процессор встраивается в защитный корпус (для электрического соединения кремниевой микросхемы с корпусом используется золото), который обеспечивает электрическое соединение процессора с системной платой.

В первом процессоре 4004 (1971 год) размеры каждого элемента составляли $10 \text{ мк} = 10^{-5} \text{ м}$ (были сравнимы с толщиной человеческого волоса), а их количество было равно 2300. В современных процессорах (2006 год) размеры каждого элемента составляют всего $65 \text{ нм (нанометров)} = 0,065 \text{ мк} = 6,5 \cdot 10^{-8} \text{ м}$ (сравнимы с размерами нескольких десятков атомов), а их количество равно 376 000 000.

Плакаты. Таблица 1 «Процессоры»

Производительность. Производительность процессора является его интегральной характеристикой и характеризует скорость выполнения приложений. Производительность процессора прямо пропорциональна разрядности процессора, его частоте, а также количеству команд, выполняемых за один такт:

$$\text{Производительность} \sim \text{Разрядность} \times \text{Частота} \times \\ \times \text{Кол-во команд за такт.}$$

Разрядность процессора определяется количеством двоичных разрядов, которые процессор обрабатывает за один такт. С момента появления первого процессора 4004 (за 35 лет) разрядность процессоров увеличилась в 16 раз (с 4 битов до 64 битов).

Однако в настоящее время увеличение разрядности процессоров (а следовательно, производительности) практически нецелесообразно, так как только сейчас происходит переход от 32-битовых операционных систем и приложений к 64-битовым.

Частота соответствует количеству тактов обработки данных, которые процессор производит за 1 секунду. С момента появления первого процессора частота процессоров увеличилась в 37 000 раз (с 0,1 МГц до 3700 МГц).

Однако увеличение производительности процессоров за счет увеличения частоты имеет свой предел из-за тепловыделения. Выделение процессором теплоты Q пропорционально потребляемой мощности P , которая, в свою очередь, пропорциональна квадрату частоты ν^2 :

$$Q \sim P \sim \nu^2.$$

Уже в настоящее время (2006 год) для отвода тепла от процессора используются массивные воздушные системы охлаждения (их называют кулерами) (рис. 1.7), состоящие из вентилятора и металлических теплоотводящих ребер. Разрабатываются и применяются также водяные системы охлаждения процессоров.



Рис. 1.7. Кулер для процессора

Практическое задание «Определение температуры процессора». С помощью программы тестирования SIV определить температуру процессора и скорость вращения кулера процессора.



Определение температуры процессора

1. Запустить программу тестирования SIV и выбрать вкладку *Hardware Monitor* (рис. 1.8).

На панели отобразится температура процессора и количество оборотов в минуту кулера процессора, в данном случае:

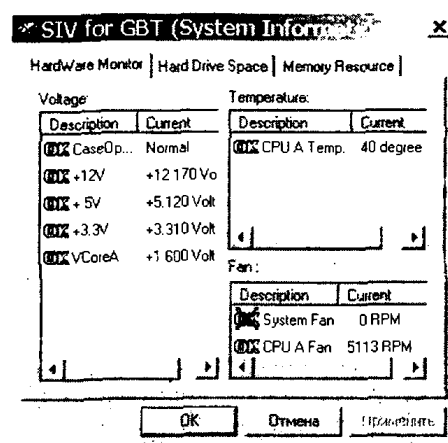


Рис. 1.8. Определение температуры процессора

- температура процессора — 40 градусов по Цельсию;
- количество оборотов в минуту кулера процессора — 5 113 об/мин.

В настоящее время производительность процессоров увеличивается путем совершенствования архитектуры процессора. Во-первых, в структуру процессора вводится кэш-память 1-го и 2-го уровней, которая позволяет ускорить выборку команд и данных и тем самым уменьшить время выполнения одной команды. Во-вторых, вместо одного ядра процессора (АЛУ и др.) используется два ядра, что позволяет повысить производительность процессора примерно на 80%.

Производительность процессора нельзя вычислить, она определяется в процессе тестирования, по скорости выполнения процессором определенных операций в какой-либо программной среде.

Практическое задание «Производительность процессора». С помощью программы тестирования SiSoft Sandra определить производительность вашего процессора и сравнить ее с производительностью:

- аналогичного процессора;
- высокопроизводительного двухъядерного процессора фирмы Intel;
- высокопроизводительного двухъядерного процессора фирмы AMD;
- процессора Pentium начального уровня.



Производительность процессора

1. Запустить программу тестирования SiSoft Sandra и выбрать тестирующий модуль *Арифметический тест процессора*.
2. Из списка *Эталонный процессор 1* выбрать аналогичный процессор.
3. Из списка *Эталонный процессор 2* выбрать высокопроизводительный двухъядерный процессор фирмы Intel.
4. Из списка *Эталонный процессор 3* выбрать высокопроизводительный двухъядерный процессор фирмы AMD.
5. Из списка *Эталонный процессор 4* выбрать процессор Pentium начального уровня.
6. Щелкнуть по кнопке *Обновить*, и через несколько секунд появится информация о производительности вашего процессора.
7. Для каждого процессора будут графически и численно выведены производительности (рис. 1.9):
 - количество целочисленных операций, выраженных в MIPS (Million Instructions Per Second — миллионах операций в секунду);
 - количество операций с плавающей точкой, выраженных в MFLOPS (Million FLoat Operations Per Second — миллионах операций с плавающей точкой в секунду).

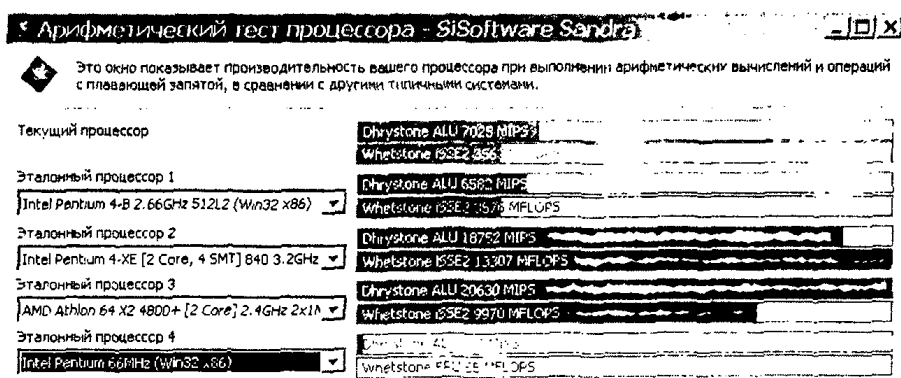






Рис. 1.9. Производительность процессора

Контрольные вопросы

1. Чем будет отличаться упрощенная логическая схема двухъядерного процессора от упрощенной логической схемы одноядерного процессора?
2. Опишите технологию производства процессоров.
3. Какие факторы влияют на производительность процессоров и как можно ее увеличить?

Компьютерный практикум

Windows-CD 

- 1.2. Ознакомиться с историей развития процессоров, используя материалы виртуального музея фирмы Intel. <http://www.intel.ru>
- 1.3. С помощью программы тестирования компьютера CPU-Z выполнить практическое задание «Определение объемов кэш-памяти процессора». 
- 1.4. С помощью программы тестирования компьютера SIV выполнить практическое задание «Определение температуры процессора». 
- 1.5. С помощью программы тестирования компьютера SiSoft Sandra выполнить практическое задание «Производительность процессора». 

1.2.2. Оперативная память

Логическая структура оперативной памяти. Оперативная память представляет собой множество ячеек, причем каждая ячейка имеет свой уникальный адрес (нумерация ячеек начинается с нуля).

Каждая ячейка памяти имеет объем 1 байт, следовательно, максимальный объем адресуемой памяти равен произведению количества ячеек N на 1 байт. Для процессоров Pentium 4, у которых разрядность шины адреса составляет 36 битов, максимальный объем адресуемой памяти равен (см. табл. 1.1):

$$\begin{aligned} N \times 1 \text{ байт} &= 2^l \times 1 \text{ байт} = 2^{36} \times 1 \text{ байт} = \\ &= 68\ 719\ 476\ 736 \text{ байт} = 67\ 108\ 864 \text{ Кбайт} = \\ &= 65\ 536 \text{ Мбайт} = 64 \text{ Гбайт}. \end{aligned}$$

Таблица 1.1. Оперативная память

Объем памяти	Ячейки	Десятичный адрес ячейки	Шестнадцатеричный адрес ячейки
64 Гбайт	10101010	68 719 476 735	FFFFFFFF
...
4 Гбайт	10101010	4 294 967 295	FFFFFFFF
...
	10101010	0	0

Модули оперативной памяти. Оперативная память предназначена для хранения информации. Изготавливается в виде модулей памяти. Модули памяти представляют собой пластины с рядами контактов, на которых размещаются микросхемы памяти. Модули памяти (DDR, DDR2 и др., рис. 1.10) устанавливаются

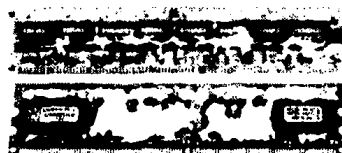


Рис. 1.10. Модули памяти DDR и DDR2

в специальные разъемы на системной плате и могут различаться между собой по количеству контактов, по быстродействию, по информационной емкости и т. д.

В персональных компьютерах величина адресного пространства процессора (объем адресуемой памяти) и величина фактически установленной оперативной памяти (модулей оперативной памяти) практически всегда различаются. Хотя объем адресуемой памяти у большинства современных процессоров равен 64 Гбайт, величина фактически установленной оперативной памяти может быть значительно меньше, например, может быть установлено 4 модуля памяти по 1 Гбайт («всего» 4 Гбайт).

Пропускная способность. Важнейшей характеристикой модулей оперативной памяти является пропускная способность, которая равна произведению разрядности шины данных и частоты операций записи или считывания информации из ячеек памяти:

$$\begin{aligned} \text{Пропускная способность} &= \\ &= \text{Разрядность шины данных} \times \text{Частота}. \end{aligned}$$

Разрядность шины данных составляет 64 бита, а максимально возможная в настоящее время (2006 год) частота шины данных совпадает с частотой системной шины и равна 1064 МГц. Тогда пропускная способность модулей памяти, подключенных к этой шине, равна:

$$\begin{aligned} \text{Пропускная способность} &= 64 \text{ бита} \times 1064 \text{ МГц} = \\ &= 68\,096 \text{ Мбит/с} = 8512 \text{ Мбайт/с} \approx 8 \text{ Гбайт/с}. \end{aligned}$$



Модули памяти маркируются своей пропускной способностью, выраженной в Мбайт/с: PC3200, PC4200, PC8500 и др.

Физическая и виртуальная память. Объем используемой программами памяти можно увеличить путем добавления к физической памяти (модулям оперативной памяти) виртуальной памяти. Виртуальная память выделяется в форме области жесткого диска. В ОС Windows это файл подкачки. По своей логической организации виртуальная память является частью оперативной памяти.

Размер файла подкачки и его размещение в иерархической файловой системе можно изменять. Однако необходимо учитывать, что быстродействие жесткого диска и, соответственно, виртуальной памяти существенно меньше быстродействия модулей оперативной памяти.

Замедление быстродействия виртуальной памяти может происходить в результате фрагментации данных в файле. Для того чтобы этого не происходило, рекомендуется произвести дефрагментацию диска и установить для файла подкачки постоянный размер.

Практическое задание «Виртуальная память». В операционной системе Windows и с использованием программы тестирования компьютера:

- установить объем виртуальной памяти и местоположение файла подкачки;
- определить объем виртуальной памяти и местоположение файла подкачки;
- определить загруженность процессора и объем используемой виртуальной памяти.



Виртуальная память

1. Щелкнуть правой кнопкой мыши по значку *Мой компьютер* и в контекстном меню активизировать пункт *Свойства*.
2. В диалоговом окне *Свойства системы* выбрать вкладку *Дополнительно* и в разделе *Быстродействие* щелкнуть по кнопке *Параметры*.
3. В появившемся диалоговом окне *Параметры быстродействия* выбрать вкладку *Дополнительно*. В разделе *Виртуальная память* щелкнуть по кнопке *Изменить*.

4. В появившемся диалоговом окне *Виртуальная память* (рис. 1.11) выбрать место размещения файла подкачки.

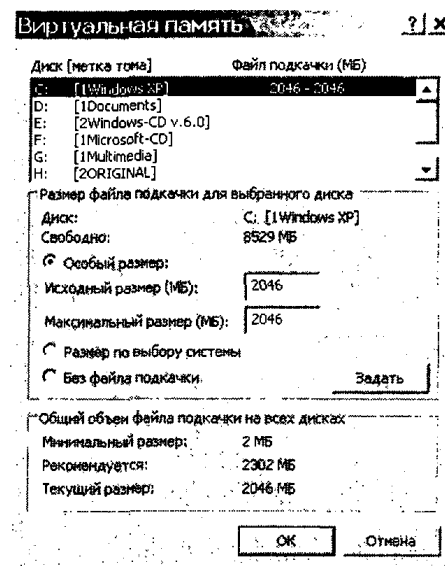


Рис. 1.11. Установка размера и местоположения виртуальной памяти

5. Для предотвращения фрагментации файла подкачки установить одинаковые значения для его исходного и максимального размеров.

Важное значение для производительности компьютера имеет объем оперативной памяти, причем не только объем установленных модулей оперативной памяти, но и общий объем системной оперативной памяти с учетом объема файла подкачки `C:\pagefile.sys`.



Виртуальная память

6. Запустить программу тестирования SiSoft Sandra и выбрать информационный модуль *Информация о памяти Windows*.

В окне появится информация об объемах памяти тестируемого компьютера, например (рис. 1.12):

всего физической памяти — 1,5 Гбайт;
файл подкачки — `C:\pagefile.sys 2046 2046`.

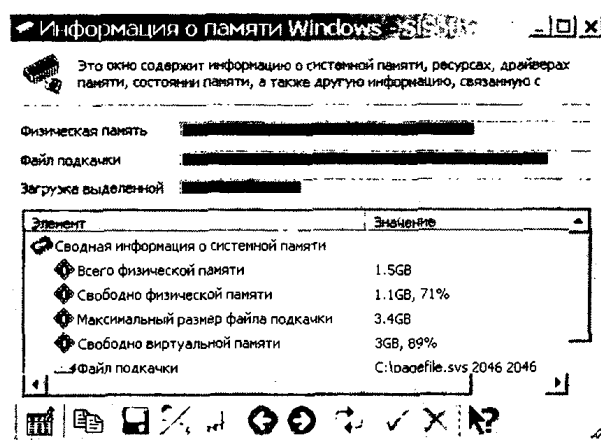


Рис. 1.12. Определение размера и местоположения виртуальной памяти

Производительность компьютера зависит не только от частоты процессора и объема системной оперативной памяти, но и от загрузки процессора и процента использования виртуальной памяти. Производительность компьютера уменьшается, если загрузка процессора достигает 100%, а также начинает использоваться более медленная, чем физическая, виртуальная память на жестком диске. Загрузку процессора и использование виртуальной памяти в операционной системе Windows 9x/Me можно наблюдать с помощью служебной программы Системный монитор, а в операционной системе Windows NT/2000/XP для этого используется Диспетчер задач Windows.



Виртуальная память

- Для вызова диспетчера задач одновременно нажать три клавиши $\{Ctrl\} + \{Alt\} + \{Del\}$.

На диалоговой панели *Диспетчер задач Windows* (рис. 1.13) выбрать вкладку *Быстродействие* и наблюдать с помощью гистограмм и графиков изменение во времени загрузки процессора и использования виртуальной памяти.

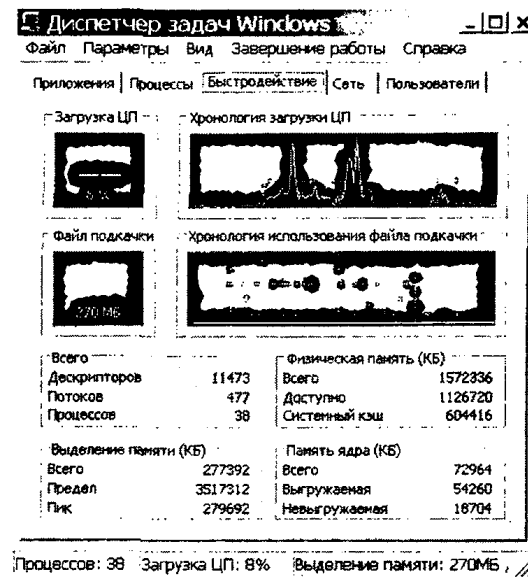


Рис. 1.13. Определение загруженности процессора и использования виртуальной памяти

Контрольные вопросы

1. В чем состоит различие между максимальным объемом адресуемой оперативной памяти и физически установленной оперативной памятью?
2. В чем состоит различие между физически установленной оперативной памятью и виртуальной памятью?
3. От каких факторов зависит пропускная способность модулей оперативной памяти?




Задачи

- 1.1. Вычислить частоту шины памяти, к которой подключены модули оперативной памяти PC3200.

Компьютерный практикум



Windows-CD

- 1.6. В операционной системе Windows выполнить первую часть практического задания «Виртуальная память» — установить объем виртуальной памяти и местоположение файла подкачки. 
- 1.7. С помощью программы тестирования компьютера SiSoft Sandra выполнить вторую часть практического задания «Виртуальная память» — определить объем виртуальной памяти и местоположение файла подкачки. 
- 1.8. В операционной системе Windows выполнить третью часть практического задания «Виртуальная память» — определить загруженность процессора и объем используемой виртуальной памяти. 

1.3. Внешняя (долговременная) память

1.3.1. Магнитная память

Основной функцией внешней памяти компьютера является долговременное хранение большого объема информации (программы, документы, аудио- и видеоклипы и т. д.). Устройство, которое обеспечивает запись/считывание информации, называется **накопителем** или **дисководом**, а хранится информация на **носителях** (например, дисках).

Магнитный принцип записи и считывания информации. В накопителях на гибких магнитных дисках (НГМД) и накопителях на жестких магнитных дисках (НЖМД), или «винчестерах», в основу записи информации положено намагничивание ферромагнетиков в магнитном поле, хранение информации основывается на сохранении намагниченности, а считывание информации базируется на явлении электромагнитной индукции.

Физика-10

В процессе записи информации на гибкие и жесткие магнитные диски головка дисковода с сердечником из магнитомягкого материала (малая остаточная намагниченность) перемещается вдоль магнитного слоя магнито жесткого носителя (большая остаточная намагниченность). На магнитную головку поступают последовательности электрических импу-

льсов (последовательности логических единиц и нулей), которые создают в головке магнитное поле. В результате последовательно намагничиваются (логическая единица) или не намагничиваются (логический нуль) элементы поверхности носителя.

При считывании информации при движении магнитной головки над поверхностью носителя намагниченные участки носителя вызывают в ней импульсы тока (явление электромагнитной индукции). Последовательности таких импульсов передаются по магистрали в оперативную память компьютера.

В отсутствие сильных магнитных полей и высоких температур элементы носителя могут сохранять свою намагниченность в течение долгого времени (лет и десятилетий).

Гибкие магнитные диски. Гибкие магнитные диски помещаются в пластмассовый корпус. Такой носитель информации называется дискетой. В центре дискеты имеется приспособление для захвата и обеспечения вращения диска внутри пластмассового корпуса. Дискета вставляется в дисковод, который вращает диск с постоянной угловой скоростью.

При этом магнитная головка дисковода устанавливается на определенную концентрическую дорожку диска, на которую и производится запись или с которой производится считывание информации. Информационная емкость современной дискеты невелика и составляет всего 1,44 Мбайт. Скорость записи и считывания информации также мала (составляет всего около 50 Кбайт/с) из-за медленного вращения диска (360 об/мин).

В целях сохранения информации гибкие магнитные диски необходимо предохранять от воздействия сильных магнитных полей и нагревания, так как такие физические воздействия могут привести к размагничиванию носителя и потере информации.

Жесткие магнитные диски. Жесткие магнитные диски (рис. 1.14) представляют собой несколько дисковых магнитных пластин, размещенных на одной оси, заключенных в металлический корпус и вращающихся с большой угловой скоростью. Обе стороны каждой пластины покрыты тонким слоем намагничиваемого материала.

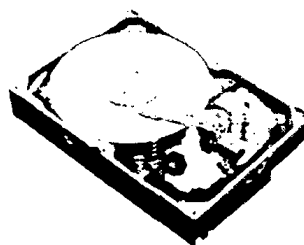


Рис. 1.14. Жесткий диск

У каждой магнитной стороны каждой пластины есть своя магнитная головка чтения/записи, головки также соединены вместе и движутся радиально по отношению к магнитным пластинам, обеспечивая, таким образом, доступ к любой дорожке любой пластины. Встроенная электроника предназначена для управления движением магнитных головок чтения/записи.

За счет использования нескольких дисковых магнитных пластин и гораздо большего количества дорожек на каждой стороне магнитных пластин информационная емкость жестких дисков может в сотни тысяч раз превышать информационную емкость дискет и достигать 500 Гбайт. Скорость записи и считывания информации на жестких дисках может достигать 300 Мбайт/с (по шине SATA) за счет быстрого позиционирования магнитной головки и высокой скорости вращения дисков (до 7200 об/мин).


В жестких дисках используются достаточно хрупкие и миниатюрные элементы (магнитные пластины носителей, магнитные головки и т. д.), поэтому в целях сохранения информации и работоспособности жесткие диски необходимо оберегать от ударов и резких изменений пространственной ориентации в процессе работы.

Контрольные вопросы

1. Почему сердечник магнитной головки изготавливается из магнитомягкого материала, а магнитный слой носителя — из магнитожесткого материала?
2. Как можно увеличить информационную емкость жестких дисков?

1.3.2. Оптическая память

Оптический принцип записи и считывания информации.
В лазерных CD- и DVD-дисководах используется оптический принцип записи и считывания информации.

Физика-11 

В процессе записи информации на оптические диски для создания участков поверхности с различными коэффициентами отражения применяются различные технологии: от простой штамповки до изменения отражающей способности участков поверхности диска с помощью мощного лазера.

Информация на лазерном диске записывается на одну спиральную дорожку, начинающуюся от центра диска и содержащую чередующиеся участки с различной отражающей способностью.

В процессе считывания информации с оптических дисков луч лазера, установленного в дисководе, падает на поверхность вращающегося диска и отражается. Так как поверхность оптического диска имеет участки с различными коэффициентами отражения, то отраженный луч также меняет свою интенсивность (логический 0 или 1). Затем отраженные световые импульсы преобразуются с помощью фотоэлементов в электрические импульсы и по магистрали передаются в оперативную память.

При соблюдении правил хранения (хранения в футлярах в вертикальном положении) и эксплуатации (без нанесения царапин и загрязнений) оптические носители могут сохранять информацию в течение десятков лет.

Оптические диски (рис. 1.15). Оптические CD-диски рассчитаны на использование инфракрасного лазера с длиной волны 780 нм и имеют информационную емкость 700 Мбайт.

Оптические DVD-диски рассчитаны на использование красного лазера с длиной волны 650 нм. Они имеют большую информационную емкость по сравнению с CD-дисками (4,7 Гбайт) за счет меньшей ширины и более плотного размещения оптических дорожек. DVD-диски могут быть двухслойными (емкость 8,5 Гбайт), при этом оба слоя имеют отражающую поверхность, несущую информацию.



Рис. 1.15. CD- и DVD-диски

В настоящее время (2006 год) на рынок поступили оптические диски (HD DVD и Blu-Ray), информационная емкость которых в 3–5 раз превосходит информационную емкость DVD-дисков за счет использования синего лазера с длиной волны 405 нанометров.

На дисках CD-ROM и DVD-ROM хранится информация, которая была записана на них в процессе изготовления. Запись на них новой информации невозможна, что отражено во второй части их названий: ROM (Read Only Memory — память только для чтения). Производятся такие диски путем штамповки на дорожке микроскопических физических углублений (участков с плохой отражающей способностью).

На дисках CD-R и DVD±R (R — recordable, записываемый) информация может быть записана, но только один раз. Данные записываются на диск лучом лазера повышенной мощности, который разрушает органический краситель записывающего слоя и меняет его отражательные свойства. Управляя мощностью лазера, на записывающем слое получают чередование темных и светлых пятен, которые при чтении интерпретируются как логические 0 и 1.

На дисках CD-RW и DVD±RW (RW — ReWritable, перезаписываемый) информация может быть записана и стерта многократно. Записывающий слой изготавливается из специального сплава, который можно нагреванием приводить в два различных устойчивых агрегатных состояния — аморфное и кристаллическое. При записи (или стирании) луч лазера нагревает участок дорожки и переводит его в одно из устойчивых агрегатных состояний, которые характеризуются различной степенью прозрачности. При чтении луч лазера имеет меньшую мощность и не изменяет состояние записывающего слоя, а чередующиеся участки с различной прозрачностью интерпретируются как логические 0 и 1.

Оптические дисководы. Оптические CD- и DVD-дисководы используют лазер для чтения или записи информации. Скорость чтения/записи информации в оптических дисководах зависит от скорости вращения диска.

Первые CD-дисководы были однокоростными и обеспечивали скорость считывания информации 150 Кбайт/с. В настоящее время широкое распространение получили CD-дисководы, которые обеспечивают в 52 раза большую скорость чтения и записи дисков CD-R (до 7,8 Мбайт/с). Запись CD-RW дисков производится на меньшей скорости (например, 32-кратной), поэтому CD-дисководы маркируются тремя числами «скорость чтения × скорость записи CD-R × скорость записи CD-RW» (например, «52×52×32»).

Первое поколение DVD-накопителей обеспечивало скорость считывания информации примерно 1,3 Мбайт/с. В настоящее время широкое распространение получили DVD-дис-

ководы, которые обеспечивают в 16 раз большую скорость чтения (примерно 21 Мбайт/с), в 8 раз большую скорость записи DVD±R дисков и в 6 раз большую скорость записи DVD±RW дисков. DVD-дисководы маркируются тремя числами (например, «16×8×6»).

Контрольные вопросы

1. Почему в CD-дисководах используется инфракрасный лазер, в DVD-дисководах — красный лазер, а в HD DVD- и Blu-Ray-дисководах — синий лазер?
2. В чем состоит различие между дисками CD-ROM, CD-R и CD-RW?
3. Что означают числа маркировки DVD-дисководов?

1.3.3. Флэш-память


Флэш-память — особый вид полупроводниковой энерго-независимой перезаписываемой памяти:

- полупроводниковая (твердотельная) — не содержащая механически движущихся частей (как обычные жесткие диски или CD), построенная на основе полупроводниковых микросхем;
- энергонезависимая — не требующая дополнительной энергии для хранения данных (энергия требуется только для записи);
- перезаписываемая — допускающая изменение (перезапись) хранимых в ней данных.



Название флэш-памяти было дано во время разработки первых микросхем (в начале 1980-х годов) как характеристика скорости стирания флэш-памяти (от англ. «in a flash» — в мгновение ока).

Принцип записи и считывания информации на картах флэш-памяти. Во флэш-памяти для записи и считывания информации используются электрические сигналы.

Физика-11 

В простейшем случае каждая ячейка флэш-памяти хранит один бит информации и состоит из одного полевого транзистора со специальной электрически изолированной областью («плавающим» затвором) (рис. 1.16).

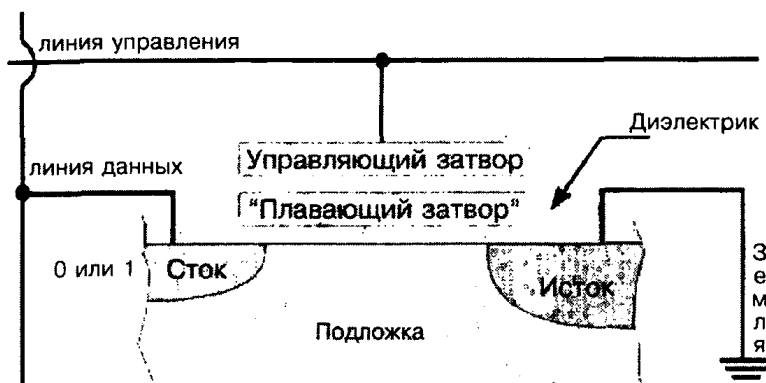


Рис. 1.16. Ячейка флэш-памяти на полевом транзисторе

При отсутствии сигнала на линии управления (на управляющем затворе) ячейка памяти хранит один бит информации (0 или 1) на стоке полевого транзистора. Между стоком и истоком ток не идет.

При записи данных на линию управления подается положительное напряжение и электроны в результате эффекта туннелирования попадают на плавающий затвор. Между стоком и истоком возникает электрический ток и в результате на стоке полевого транзистора записывается один бит данных.



Эффект туннелирования — один из квантомеханических эффектов, использующих волновые свойства электрона. В полевом транзисторе этот эффект заключается в «перескакивании» электрона через слой диэлектрика между стоком и «плавающим» затвором под действием электрического поля.

Карты флэш-памяти (рис. 1.17). Флэш-память представляет собой микросхему, помещенную в миниатюрный плоский корпус. Микросхемы флэш-памяти могут содержать миллиарды ячеек, каждая из которых хранит 1 бит информации. Информационная емкость карт флэш-памяти может достигать 4 Гбайт (2006 год).

Информация, записанная на флэш-память, может храниться очень длительное время (от 20 до 100 лет) и способна выдерживать значительные механические нагрузки (в 5–10 раз превышающие предельно допустимые для жестких дисков). Флэш-память компактнее и потребляет значительно меньше энергии (примерно в 10–20 раз), чем магнитные и оптические дисководы.

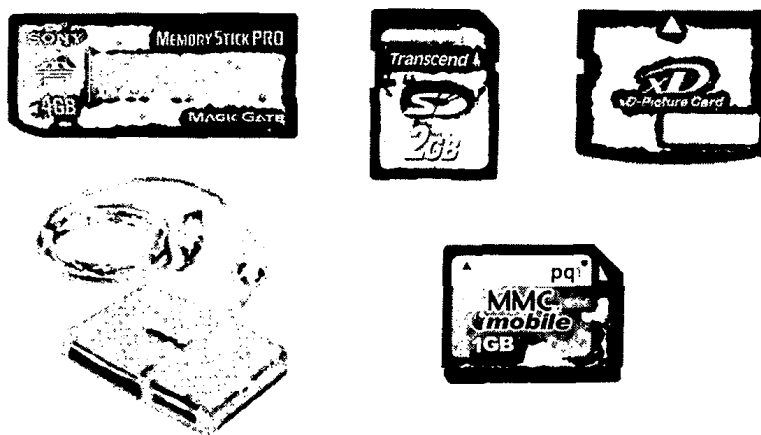


Рис. 1.17. Карты флэш-памяти и картридер

Благодаря низкому энергопотреблению, компактности, долговечности и относительно высокому быстродействию, флэш-память идеально подходит для использования в качестве носителя информации в портативных устройствах: цифровых фото- и видеокамерах, сотовых телефонах, портативных компьютерах, MP3-плеерах и цифровых диктофонах.

К недостаткам флэш-памяти следует отнести то, что не существует единого стандарта и различные производители изготавливают несовместимые друг с другом по размерам и электрическим параметрам карты памяти (Memory Stick, MultiMedia Card, Secure Digital, xD-Picture и др.). Для считывания или записи информации карта памяти вставляется в специальные накопители (картридеры), встроенные в мобильные устройства или подключаемые к компьютеру через USB-порт.

USB флэш-диски (рис. 1.18). Накопители на флэш-памяти представляет собой микросхему флэш-памяти, дополненную контроллером USB, и подключаются к последовательно-

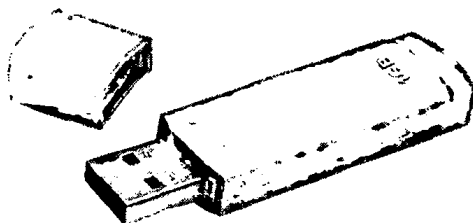


Рис. 1.18. USB флэш-диск

му порту USB. USB флэш-диски могут использоваться в качестве внешнего сменного носителя информации.

USB флэш-диски могут содержать переключатель защиты от записи, поддерживать парольную защиту, а также могут быть загрузочными. Некоторые USB флэш-диски имеют кроме индикатора работы жидкокристаллический экранчик, на котором отображается, сколько свободного места осталось на диске.

Контрольные вопросы

1. В чем состоит отличие микросхем флэш-памяти от микросхем оперативной памяти?
2. В чем состоит преимущество флэш-памяти перед магнитной и оптической долговременной памятью?

1.4. Файл и файловые системы

1.4.1. Логическая структура носителя информации

2.3.2. Файловая система

Информатика-8



Логическая структура носителя информации в распространенной файловой системе FAT имеет следующие разделы:

- загрузочный кластер;
- таблицу размещения файлов, которая содержит в своих ячейках цепочку номеров кластеров для каждого файла;
- корневой каталог;
- файлы.

В файловой системе FAT используется следующая организация хранения.

Минимальным адресуемым элементом носителя информации является кластер, который может включать в себя несколько секторов (объем сектора составляет 512 байтов). Размер кластера (от 512 байтов до 64 Кбайт) зависит от типа используемой файловой системы. Кластеры нумеруются в линейной последовательности (на магнитных дисках от первого кластера нулевой дорожки до последнего кластера последней дорожки).

Файловая система организует кластеры в файлы и каталоги (реально являющиеся файлами, содержащими список файлов в этом каталоге). Файловая система отслеживает, ка-

кие из кластеров в настоящее время используются, какие свободны, какие помечены как неисправные.

При записи файлов будет занято всегда целое количество кластеров, соответственно минимальный размер файла равен размеру одного кластера, а максимальный соответствует общему количеству кластеров на диске.

Файл записывается в произвольные свободные кластеры. Например, Файл_1 может занимать кластеры 34, 35 и 47, 48, а Файл_2 — кластеры 36 и 49 (табл. 1.2).

Таблица 1.2. Логическая структура носителя информации

№ кластера																	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54
...																	

Таблица размещения файлов. Полная информация о кластерах, которые занимают файлы, содержится в таблице размещения файлов FAT. В целях более надежного сохранения информации о размещении файлов таблица FAT хранится на носителе информации в двух идентичных копиях. Количество ячеек FAT соответствует количеству кластеров на диске, а значениями ячеек являются цепочки размещения файлов, т. е. последовательности адресов кластеров, в которых хранятся файлы.

Например, для двух рассмотренных выше файлов таблица FAT с 1-й по 54-ю ячейку принимает следующий вид (табл. 1.3).

Таблица 1.3. Фрагмент FAT

															35	47	36
										48	К	К					

Цепочка размещения для файла Файл_1 выглядит следующим образом: в начальной 34-й ячейке FAT хранится адрес следующего кластера (35), соответственно, в следующей 35-й ячейке хранится 47, в 47-й — 48, в 48-й — знак конца файла (К).

FAT12. Файловая система для операционной системы Windows. Выделяет 12 битов для хранения адреса кластера,

соответственно, она может адресовать $2^{12} = 4096$ кластеров. Объем кластера по умолчанию равен размеру одного сектора (512 байтов), и поэтому FAT12 не может использоваться для носителей информации объемом более:

$$\begin{aligned} 512 \text{ байт} \times 4096 &= 2\,097\,152 \text{ байт} = \\ &= 2048 \text{ Кбайт} = 2 \text{ Мбайт}. \end{aligned}$$

Такой объем имеют дискеты, поэтому FAT12 используется именно для дискет.

FAT16. Файловая система для операционной системы Windows. Выделяет 16 битов для хранения адреса кластера, соответственно, она может адресовать $2^{16} = 65\,536$ кластеров. Объем кластера не может быть более 128 секторов (64 Кбайт), и поэтому FAT16 не может использоваться для носителей информации объемом более:

$$\begin{aligned} 64 \text{ Кбайт} \times 65\,536 &= 4\,194\,304 \text{ Кбайт} = \\ &= 4096 \text{ Мбайт} = 4 \text{ Гбайт}. \end{aligned}$$

В настоящее время такой объем имеет флэш-память, поэтому FAT16 используется именно для флэш-памяти.

FAT32. Файловая система для операционной системы Windows. Выделяет 32 бита для хранения адреса кластера, соответственно, она может адресовать $2^{32} = 4\,294\,967\,296$ кластеров. Объем кластера по умолчанию составляет 8 секторов (4 Кбайт), и поэтому FAT32 может использоваться для носителей информации объемом:

$$\begin{aligned} 4 \text{ Кбайт} \times 4\,294\,967\,296 &= 17\,179\,869\,184 \text{ Кбайт} = \\ &= 16\,777\,216 \text{ Мбайт} = 16\,384 \text{ Гбайт} = 16 \text{ Тбайт}. \end{aligned}$$

Таким образом, FAT32 может использоваться для жестких дисков самого большого объема.

NTFS. Файловая система для операционной системы Windows. Позволяет устанавливать различный объем кластера (от 512 байтов до 64 Кбайт, по умолчанию 4 Кбайт). NTFS по сравнению с FAT32 увеличивает надежность и эффективность использования дискового пространства.

NTFS использует систему журналирования для повышения надежности файловой системы. Журналируемая файловая система сохраняет список изменений, которые она будет проводить с файловой системой, перед фактической записью изменений. Эти записи хранятся в отдельной части файловой системы, называемой «журналом» или «логом». Как только изменения файловой системы будут внесены в журнал, журналируемая файловая система применит эти изменения к файлам.

ext3 и ReiserFS. Журналируемые файловые системы для операционных систем Unix. ReiserFS — высоконадежная файловая система, хорошо приспособленная для хранения большого количества маленьких файлов. Блок (кластер) ext3 может иметь размер от 1 до 8 Кбайт, а в ReiserFS в одном блоке могут быть размещены данные нескольких файлов. С файлами большого размера файловая система ReiserFS также справляется весьма уверенно, максимальный размер файловой системы составляет 16 Тбайт.

HFS. Иерархическая журналируемая файловая система, разработанная Apple Computer для использования на компьютерах, работающих под управлением операционной системы Mac OS.

CDFS. Файловая система для работы с оптическими CD- и DVD-дисками, базирующаяся на стандарте ISO 9660, согласно которому имя файла не может превышать 32 символа и глубина вложения папок — не более 8 уровней.

UDF. Мультисистемная файловая система для работы с файлами на оптических дисках позволяет на перезаписываемых CD-RW и DVD±RW дисках удалять, копировать и сохранять отдельные файлы.

Практическое задание «Объем файла в различных файловых системах». В операционной системе Windows создать текстовый файл и последовательно сохранить его на гибком диске, на флэш-диске и на жестком диске. Ознакомиться с объемом текстового файла и занимаемым им объемом на дисках с различными файловыми системами.



Объем файла в различных файловых системах

1. Запустить Блокнот командой [*Программы-Стандартные-Блокнот*]. Ввести текст в файл, например «информатика».
2. Сохранить файл на гибком диске, на флэш-диске и на жестком диске.

На каждом диске определить объем, занимаемый файлом.

3. Последовательно в контекстном меню дисков активизировать пункт *Свойства*. В появившихся диалоговых окнах ознакомиться с объемом текстового файла и занимаемым им объемом на дисках (рис. 1.19).

Информационный объем текста равен:

$1 \text{ байт} \times 11 = 11 \text{ байт}$.

На гибком диске используется файловая система FAT12, и этот файл будет занимать один сектор, т. е. 512 байтов.

На флэш-диске используется файловая система FAT16, и этот файл будет занимать один кластер, объем которого равен 32 Кбайт.

На жестком диске используется файловая система FAT32 или NTFS, и этот файл будет занимать один кластер, объем которого равен 4 Кбайт.

Размещение:	A:\
Размер:	11 байт (11 байт)
На диске:	512 байт (512 байт)
Размещение:	N:\
Размер:	11 байт (11 байт)
На диске:	32,0 КБ (32 768 байт)
Размещение:	D:\
Размер:	11 байт (11 байт)
На диске:	4,00 КБ (4 096 байт)

Рис. 1.19. Объем файла на дисках с различными файловыми системами

Форматирование носителей информации. Для того чтобы на носителе можно было хранить информацию, он должен быть отформатирован, т. е. создана логическая структура в соответствии с определенной файловой системой.



Жесткие диски должны быть предварительно физически отформатированы (это делается на заводе-изготовителе). Формирование физической структуры диска состоит в создании на диске концентрических дорожек, которые, в свою очередь, делятся на секторы (в определенных местах диска расставляются метки дорожек и секторов).

Большие по объему жесткие диски рекомендуется разбивать на **разделы**, т. е. независимые области на диске. Разделы могут быть отформатированы в различных файловых системах, и, таким образом, созданы логические диски. Логическим дискам ставятся в соответствие заглавные буквы латинского алфавита (C:, D: и т. д.).

На один физический диск в разные разделы могут быть установлены различные операционные системы.

Интерфейс командной строки. Давным-давно (в 80-е годы XX века), когда еще не был разработан графический интерфейс операционных систем, форматирование дисков и другие операции с файлами проводились с использованием командной строки операционной системы MS-DOS. В операционной

системе Windows предусмотрен режим работы с интерфейсом командной строки.

В ответ на приглашение системы можно вводить команды с клавиатуры, в том числе:

- команды работы с дисками (format, defrag и др.);
- команды работы с файлами (copy, del, rename и др.);
- команды работы с каталогами (cd, dir и др.).

Каждая команда имеет свой формат, параметры которого запомнить достаточно трудно. Для того чтобы получить справочную информацию о команде, необходимо после имени команды ввести ключ */?*.

Практическое задание «Форматирование из командной строки». В операционной системе Windows в командной строке с помощью команды `format` отформатировать дискету с нестандартным размером кластера (например, 1024 байта). Предварительно получить справку о формате команды `format`.



Форматирование из командной строки

1. Ввести команду [*Программы-Стандартные-Командная строка*]. Появится окно командной строки.
2. В ответ на приглашение системы перейти в каталог C:\Windows с помощью команды CD:
C:\Documents and Setting\НДУ>CD C:\Windows
3. Получить справку о формате команды `format` (рис. 1.20):
C:\Windows>format */?*

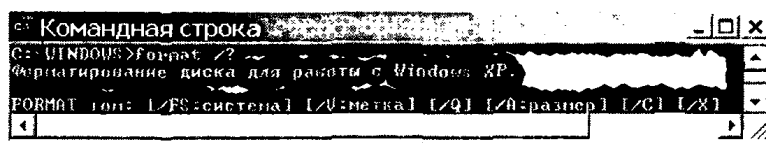


Рис. 1.20. Получение справки по формату команды

4. Отформатировать дискету с нестандартным размером кластера 1024 байта:
C:\Windows>format A:/ A:1024
После окончания форматирования можно узнать количество кластеров, количество байтов в кластере, разрядность ячейки FAT и другие параметры форматирования дискеты (рис. 1.21).

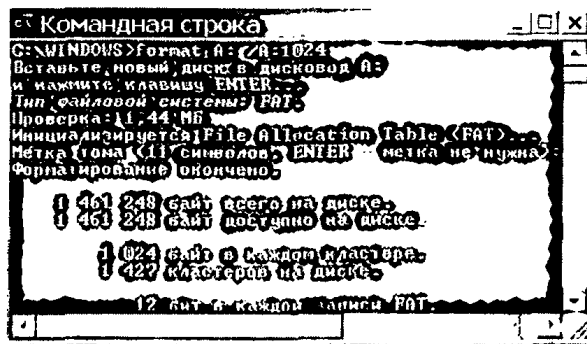


Рис. 1.21. Форматирование дискеты в командной строке




Контрольные вопросы

1. Какую последовательность кластеров занимает файл `Файл_2` из приведенного примера на носителе информации?
2. Чем различаются таблицы размещения файлов FAT12, FAT16 и FAT32?
3. Почему журналируемые файловые системы обеспечивают большую надежность хранения файлов?

Компьютерный практикум



Windows-CD

- 1.9. С помощью средств операционной системы Windows определить для диска C: тип файловой системы, размер кластера и количество кластеров. 
- 1.10. В операционной системе Windows выполнить практическое задание «Объем файла в различных файловых системах». 
- 1.11. В операционной системе Windows выполнить практическое задание «Форматирование из командной строки». 

1.4.2. Файл

Все программы и данные хранятся в долговременной памяти компьютера в виде файлов.

2.3.1. Файл Информатика-8



Файл обязательно имеет имя, формат которого определяется используемой файловой системой. Имя файла состоит из двух частей, разделенных точкой: собственно имени файла и расширения, определяющего тип файла (табл. 1.4). Расширение файла обычно дописывается к имени приложением, в котором этот файл создается. Приложение, с которым связан файл, можно изменить (например, файлы архивов с расширением zip можно связать с различными архиваторами).

Таблица 1.4. Некоторые типы файлов и расширений

Тип файла	Расширения
Исполнимые файлы	exe
Текстовые файлы	txt, doc
Web-страницы	htm, html
Графические файлы	bmp, gif, png, jpg
Звуковые файлы	wav, mid
Видеофайлы	avi, wmf
Архивы	zip, 7z

Файл в каталоге описывается записью, в которой указывается его имя, размер, дата и время создания, а также набор атрибутов:

- «только для чтения» (от англ. «read only») — файл с таким атрибутом доступен операционной системе и приложениям только для чтения, т. е. в нем нельзя сохранить какие-либо исправления;
- «скрытый» (от англ. «hidden») — файл с данным атрибутом не выводится при показе содержимого папок;
- «системный» (от англ. «system») — файлы с таким атрибутом относятся к системным файлам операционной системы, которые нельзя удалять или изменять;
- «архивный» (от англ. «archive») — этот атрибут указывает программам архивации файлы, предназначенные для резервного копирования.



Файл — это некоторая информация (программа или данные), имеющая имя и хранящаяся в долговременной (внешней) памяти.

Практическое задание «Расширение и атрибуты файла».

В операционной системе Windows файлы архивов с расширением zip связать с файловым менеджером Total Commander, а затем с помощью этого файлового менеджера изменить атрибуты и дату/время создания файла архива.



Расширение и атрибуты файла

1. Щелкнуть правой кнопкой мыши по имени файла и в контекстном меню активизировать пункт *Открыть с помощью*.
2. Если в появившемся меню нужное приложение отсутствует, выбрать пункт *Выбрать программу...*
3. В появившемся диалоговом окне *Выбор программы* (рис. 1.22) выбрать программу из списка.

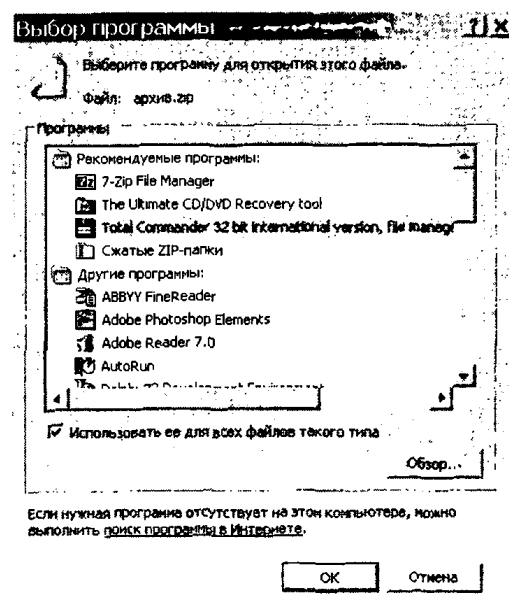


Рис. 1.22. Выбор приложения для открытия файла

4. В случае отсутствия нужного приложения в списке щелкнуть по кнопке *Обзор* и выбрать приложение в иерархической файловой системе.

5. Установить флажок *Использовать ее для всех файлов такого типа*.



Расширение и атрибуты файла

1. Запустить файловый менеджер Total Commander.
2. На панели выбрать файл, например архив.zip.
3. В меню ввести команду [*Файл- Изменить атрибуты...*].
4. В появившемся диалоговом окне *Изменение атрибутов* (рис. 1.23) установить флажок *Скрытый*. Файл перестанет отображаться на панели.

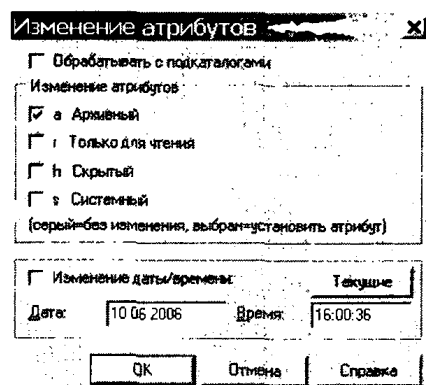


Рис. 1.23. Изменение атрибутов и даты/времени создания файла

5. Установить текущие дату и время создания файла, для этого щелкнуть по кнопке *Текущие*.

Архивация файлов. Для уменьшения места, занимаемого файлами, с целью долговременного хранения или передачи по компьютерным сетям файлы архивируются (сжимаются) с помощью файловых менеджеров или специализированных приложений — **архиваторов**.

Существуют различные алгоритмы архивации данных без потери информации, при которых при разархивации данные будут восстановлены в исходном виде. Самый простой алгоритм сжатия данных основан на замене повторяющихся битов (в тексте может иметься последовательность одинаковых символов, в графическом файле — закрашенная одним цветом область и т. д.) более короткой последовательностью битов. Например, в тексте подряд идут 10 пробелов, которые кодируются 10 байтами. При архивации они заменяются 3 байтами (первый байт кодирует заменяемый символ; второй байт — это

специальный байт «флажка» архивации, который указывает на необходимость развернуть первый байт в последовательность байтов; третий байт указывает количество повторяющихся байтов).

Алгоритм кодирования одинаковых последовательностей символов ищет в текстовых файлах одинаковые подстроки, а в графических — близкие по цвету последовательности пикселей. Каждый такой фрагмент файла представляется более коротким кодом (последовательностью битов) и в процессе архивации при повторных появлениях заменяется ссылкой на первичный код.

Лучше всего сжимаются текстовые и графические файлы, практически не сжимаются файлы архивов и исполнимые файлы.

Существуют различные архиваторы файлов (ZIP, 7-Zip, RAR и др.), которые используют вышеописанные и другие алгоритмы архивации. Архиваторы различаются степенью сжатия файлов, скоростью выполнения и другими параметрами.

При создании нового архива нужно задать параметры архивации:

- задать имя архивного файла и место его сохранения на диске;
- выбрать формат архивации ZIP, 7z, RAR или др.;
- выбрать степень сжатия файлов (от *Без сжатия* — файлы не сжимаются, до *Максимальный* для получения наилучшего сжатия);
- выбрать размер словаря и слова (чем больше их размер, тем больше, но медленнее сжатие);
- можно создать многотомный архив, т. е. архив, состоящий из нескольких частей, которые используются для сохранения большого архива на нескольких дискетах или других сменных носителях;
- можно создать непрерывный (от англ. *solid*) архив (в архиваторе RAR), что позволяет добиться максимальной степени сжатия нескольких файлов;
- можно создать самораспаковывающийся архив **SFX**, который является исполнимым файлом и имеет расширение *exe*. Для разархивации такого архива не нужен архиватор, достаточно запустить файл архива на выполнение.

Для разархивации файлов необходимо открыть файл архива в архиваторе и ввести соответствующую команду.

Практическое задание «Архивация файлов». С помощью архиватора 7-Zip осуществить архивацию файлов.



Архивация файлов

1. Запустить архиватор 7-Zip.
Выделить файлы и в контекстном меню ввести команду [7-Zip-Добавить в архив...].
2. В окне *Добавить к архиву* (рис. 1.24) задать параметры архивации:
 - имя архивного файла;
 - место хранения архива на диске;
 - формат архива;
 - метод сжатия;
 - размер словаря и слова;
 - размер тома для многотомного архива;
 - метод обновления архива;
 - непрерывность архива.

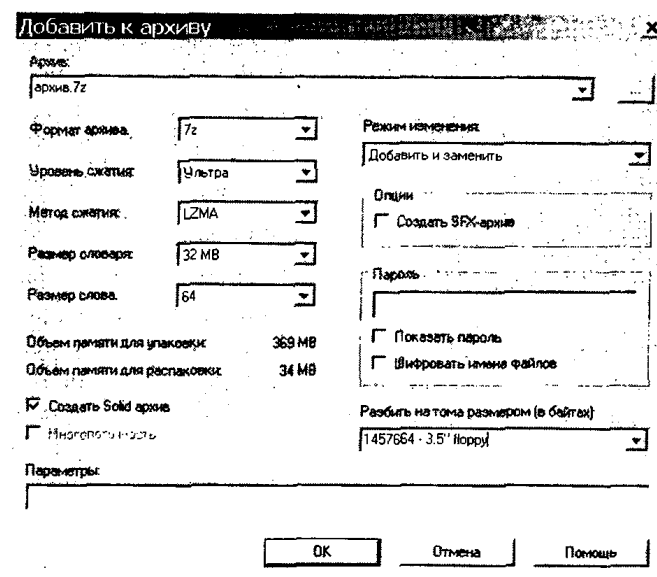


Рис. 1.24. Параметры архивации

3. Для извлечения файлов из архива необходимо выбрать файл архива и в контекстном меню ввести команду [7-Zip-Распаковать].

Контрольные вопросы

1. Как связаны между собой расширение файла и приложение, в котором этот файл был создан?
2. Какие параметры архивации влияют на степень сжатия файлов?

Компьютерный практикум

Windows-CD

- 1.12. В операционной системе Windows с помощью файлового менеджера Total Commander выполнить практическое задание «Расширение и атрибуты файла».



- 1.13. С помощью архиватора 7-Zip выполнить практическое задание «Архивация файлов».



1.4.3. Иерархическая файловая система

На каждом носителе информации (магнитном или оптическом диске, карте флэш-памяти) может храниться большое количество файлов. Для того чтобы можно было найти файл по его имени, на диске создают разделы (тома), а в каждом томе — каталоги.

Информатика-11

Каталог. Каждый том носителя информации содержит корневой каталог (базовый, основной), в котором перечислены хранящиеся на носителе файлы и папки. В простейшем случае, когда файлов мало, каталог представляет собой линейную последовательность записей о файлах.

В разных файловых системах файлы описываются по-разному. Далее рассмотрим файловую систему FAT.

Запись о файле в каталоге содержит (табл. 1.5) имя файла, адрес первого кластера, с которого начинается файл, размер файла, дату и время его создания, а также атрибуты файла (a — архивный, r — только для чтения, h — скрытый, s — системный).

Таблица 1.5. Структура записей в каталоге

Имя файла	Адрес первого кластера	Объем файла (Кбайт)	Дата создания	Время создания	Атрибуты
Файл_1	34	2	14.01.2006	14.29	ar
Файл_2	36	1	20.03.2006	19.45	hs

Если на диске хранятся сотни и тысячи файлов, то для удобства поиска файлы хранятся в многоуровневой иерархической файловой системе, которая имеет древовидную структуру (рис. 1.25).

Корневая папка содержит вложенные папки 1-го уровня (например, папки Архив и Документы), в свою очередь, каждая из них может содержать вложенные папки 2-го уровня (например, папка Документы содержит папки Web-страницы, Видео, Звук и Изображения) и т. д. Необходимо отметить, что в папках всех уровней могут храниться файлы.

Путь к файлу. Для того чтобы найти файл в иерархии каталогов, необходимо указать путь к файлу. Путь к файлу начинается с логического имени диска, затем записывается последовательность имен вложенных друг в друга папок, в последней из которых содержится нужный файл. Имена диска и папок записываются через разделитель «\». Например, путь к файлу image.bmp можно записать следующим образом:

A:\Документы\Изображения\image.bmp

Восстановление файлов и файловой системы. В процессе работы компьютера случаются сбои («зависание» программ, внезапное отключение питания и др.), в результате происходит неправильное завершение работы приложений и операционной системы, что может приводить к повреждению отдельных кластеров и файлов. Могут появиться сбойные

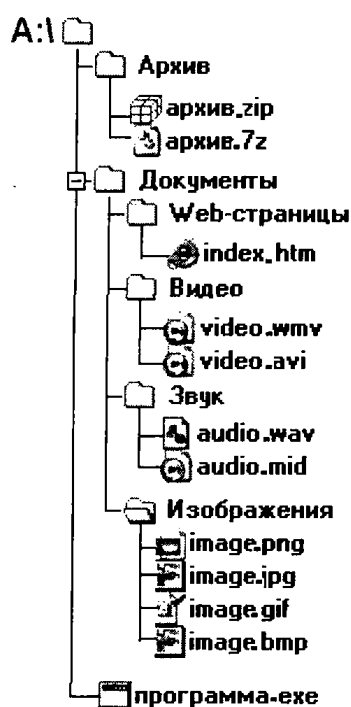


Рис. 1.25. Пример иерархической файловой системы

(нечитаемые) кластеры, в каталогах могут быть изменены имена файлов, а в FAT могут появиться нарушения в цепочках размещения файлов (некоторые цепочки могут быть оборваны, один и тот же кластер может принадлежать различным файлам и др.).

Для восстановления файловой системы используются специальные программы. В операционной системе Windows такой программой является служебная программа Проверка диска, которая автоматически запускается при загрузке Windows после неправильного завершения работы или может быть при необходимости запущена пользователем в произвольный момент.

Практическое задание «Проверка файловой системы диска». В операционной системе Windows осуществить проверку файловой системы диска.



Проверка файловой системы диска

1. В контекстном меню проверяемого диска (например, C:) выбрать пункт *Свойства*.

В появившемся диалоговом окне *Свойства* выбрать вкладку *Сервис* (рис. 1.26) и щелкнуть по кнопке *Выполнить проверку...*

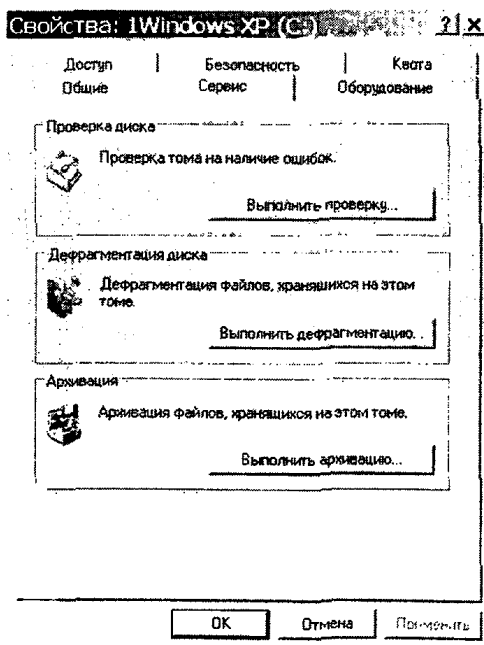


Рис. 1.26. Вкладка *Сервис*

2. В появившемся диалоговом окне *Проверка диска* (рис.1.27) с помощью флажков установить требуемые параметры и щелкнуть по кнопке *Запуск*.

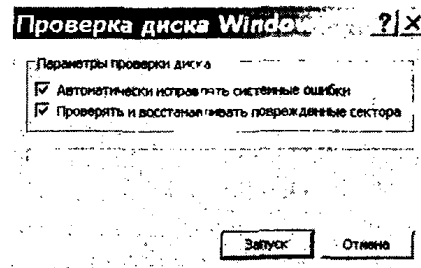


Рис. 1.27. Процесс проверки диска

3. Проверка диска будет выполнена в три этапа. После окончания проверки и восстановления сбойных кластеров и файлов появится окно с информацией о состоянии диска после проверки.

Дефрагментация дисков. Замедление скорости обмена данными с носителем информации может происходить в результате **фрагментации** файлов. Если запись производится на незаполненный диск, то кластеры, принадлежащие одному файлу, записываются подряд. Если диск переполнен, на нем может не быть непрерывной области, достаточной для размещения файла, и файл запишется в виде нескольких фрагментов. Фрагментация файлов (фрагменты файлов хранятся в различных, удаленных друг от друга кластерах) возрастает с течением времени, в процессе удаления одних файлов и записи других.

Так как на диске могут храниться сотни и тысячи файлов в сотнях тысяч кластеров, то фрагментированность файлов будет существенно замедлять доступ к ним (магнитным головкам придется постоянно перемещаться с дорожки на дорожку) и в конечном итоге приводить к преждевременному износу жесткого диска. Рекомендуется периодически проводить дефрагментацию диска, в процессе которой файлы записываются в кластеры, последовательно идущие друг за другом.

Практическое задание «Дефрагментация диска». В операционной системе Windows осуществить дефрагментацию диска.

Дефрагментация диска

1. Для запуска программы Дефрагментация диска необходимо ввести команду [*Стандартные-Служебные- Дефрагментация диска*].
2. Диалоговое окно *Дефрагментация диска* (рис. 1.28) позволяет выбрать диск, нуждающийся в процедуре дефрагментации.

После нажатия кнопки *Дефрагментация* начнется процесс дефрагментации, который можно визуально наблюдать.

Вертикальные полосы красного цвета отображают фрагментированные файлы, синего цвета — нефрагментированные файлы, зеленого цвета — неподвижные файлы и, наконец, белого цвета — свободные кластеры.

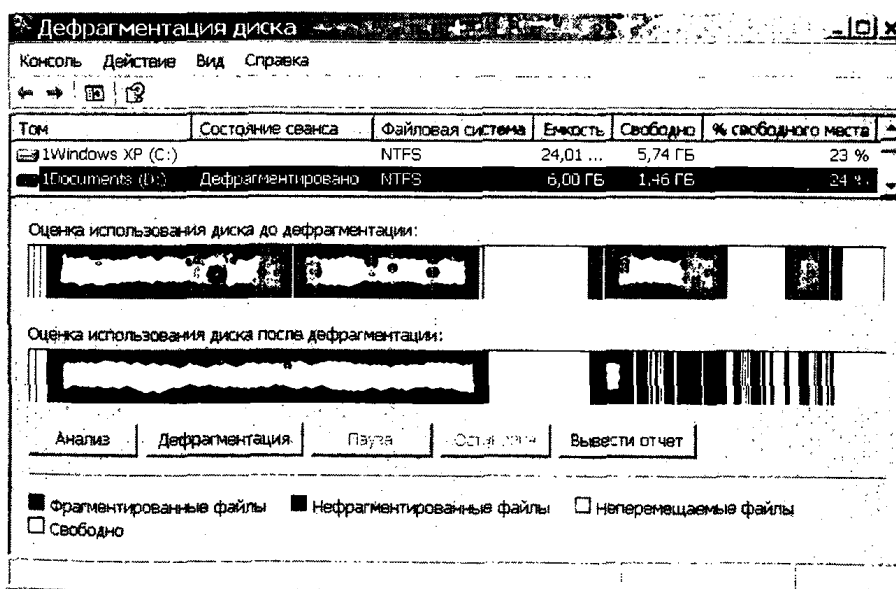


Рис. 1.28. Дефрагментация диска

Контрольные вопросы

1. В чем состоит различие между папками и каталогами?
2. Почему происходит фрагментация файлов?



- 1.2. Записать пути к файлам иерархической файловой системы, изображенной на рис. 1.25.



Windows-CD 

- 1.14. В операционной системе Windows выполнить практическое задание «Проверка файловой системы диска».
- 1.15. В операционной системе Windows выполнить практическое задание «Дефрагментация диска».



1.5. Операционная система

1.5.1. Назначение и состав операционной системы

Операционная система является базовой и необходимой составляющей программного обеспечения компьютера.



Операционная система обеспечивает совместное функционирование всех устройств компьютера и предоставляет пользователю доступ к его ресурсам.

Современные операционные системы имеют сложную структуру, каждый элемент которой выполняет определенные функции по управлению компьютером.

Управление файловой системой. В операционной системе имеются программные модули, управляющие файловой системой.

В процессе работы на компьютере наиболее часто над файлами производятся следующие операции:

- копирование (копия файла помещается в другую папку);
- перемещение (сам файл перемещается в другую папку);
- удаление (запись о файле удаляется из папки);
- переименование (изменяется имя файла).

Командный процессор. В состав операционной системы входит специальная программа — командный процессор, которая запрашивает у пользователя команды и выполняет их.

Пользователь может дать команду запуска программы, выполнения какой-либо операции над файлами (копирование, удаление, переименование), вывода документа на печать и т. д. Операционная система должна эту команду выполнить.

Практическое задание «Копирование файлов». В операционной системе Windows осуществить копирование файлов с использованием графического интерфейса и с помощью командной строки.



Копирование файлов

1. Скопировать файл с использованием графического интерфейса (копированием из папки в папку или с помощью Проводника).
2. Ввести команду [*Программы-Стандартные-Командная строка*]. Появится окно командной строки.
3. В ответ на приглашение системы перейти в каталог C:\Windows с помощью команды CD:
C:\Documents and Setting\НДУ>CD
C:\Windows
4. Получить справку о формате команды copy:
C:\Windows>copy /?
5. Ввести команду (рис. 1.29):
C:\Windows>copy A:\проба.doc D:

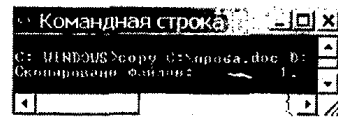


Рис. 1.29. Копирование файла в командной строке

Драйверы устройств. К магистрали компьютера подключаются различные устройства (дискетоды, монитор, клавиатура, мышь, принтер и др.). Каждое устройство выполняет определенную функцию (ввод информации, хранение информации, вывод информации), при этом техническая реализация устройств существенно различается.

Драйверы устройств — специальные программы, которые обеспечивают управление работой устройств и согласование информационного обмена с другими устройствами, а также позволяют производить настройку некоторых их параметров. Каждому типу устройств соответствует свой драйвер.

В процессе установки операционная система определяет тип и конкретную модель установленного устройства и подключает необходимые для их функционирования драйверы. При включении компьютера производится загрузка драйверов в оперативную память. Пользователь имеет возможность вручную установить или переустановить драйверы.

Графический интерфейс. Для упрощения работы пользователя в состав современных операционных систем входят программные модули, создающие графический пользовательский интерфейс. В операционных системах с графическим интерфейсом пользователь может вводить команды с помощью диалоговых окон, тогда как в режиме командной строки необходимо вводить команды с помощью клавиатуры.

Диалоговые окна могут включать в себя разнообразные элементы управления:

- вкладки, которые позволяют использовать «страницы» внутри диалогового окна;
- кнопки, которые обеспечивают выполнение тех или иных действий;
- текстовые поля, которые позволяют ввести информацию;
- списки, которые представляют собой наборы предлагаемых на выбор значений;
- переключатели, которые служат для выбора одного из взаимоисключающих вариантов;
- флажки, которые обеспечивают присваивание каким-либо параметрам определенных значений;
- счетчики, которые представляют собой пару стрелок и позволяют увеличивать или уменьшать значение в связанном с ними поле;
- ползунки, которые позволяют плавно изменять значения каких-либо параметров;
- контекстные меню, которые позволяют ознакомиться со свойствами объектов, а также выполнить над ними разрешенные операции.

Такие же элементы управления используются в системах объектно-ориентированного программирования, что позволяет создавать в них приложения с использованием графического интерфейса операционной системы.

Глава 4. Алгоритмизация и объектно-ориентированное программирование

Сервисные программы. В состав операционной системы входят также сервисные программы, или утилиты. Такие программы позволяют обслуживать диски (проверять, сжимать, дефрагментировать и т. д.), выполнять операции с файлами (архивировать и т. д.), работать в компьютерных сетях и т. д.

Системный реестр Windows. Системный реестр операционной системы Windows является иерархической базой данных, в которой хранится информация о конфигурации Windows. В реестре содержатся сведения об оборудовании системы, установленных программах и параметрах настройки, к которым операционная система постоянно обращается во время работы. В операционную систему Windows входит редактор реестра — программа `regedit.exe`.

Практическое задание «Ознакомление с системным реестром Windows». Ознакомиться с реестром на примере установки в редакторе реестра цвета надписей на кнопках графического интерфейса.



Ознакомление с системным реестром Windows

1. Ввести команду [*Выполнить*].

В появившемся диалоговом окне *Запуск программы* (рис. 1.30) ввести в список имя программы — редактора реестра и щелкнуть по кнопке *OK*.

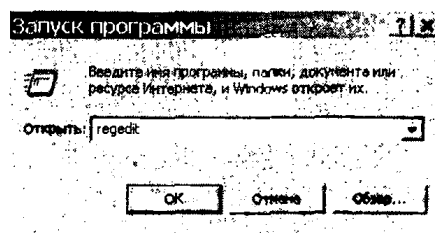


Рис. 1.30. Запуск редактора реестра

2. В появившемся диалоговом окне *Редактор реестра* (рис. 1.31) в иерархическом дереве реестра открыть пункт

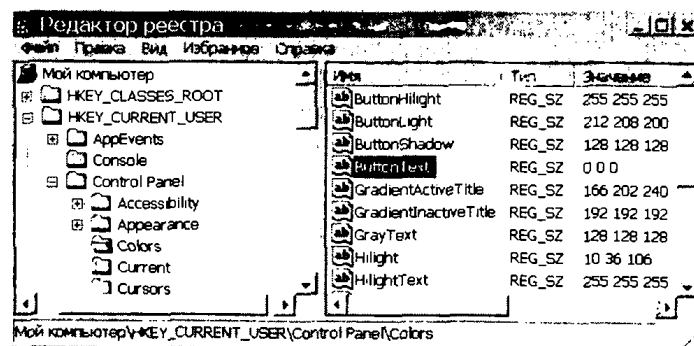


Рис. 1.31. Редактор реестра

HKEY_CURRENT_USER, а затем открыть вложенный пункт *Control Panel* и в нем активизировать пункт *Colors*. В правой части окна редактора реестра в списке параметров выделить параметр *ButtonText*.

3. В диалоговом окне *Изменение строкового параметра* (рис. 1.32) ввести новое значение параметра *ButtonText* (например, *255 0 0* — красный цвет надписей на кнопках).

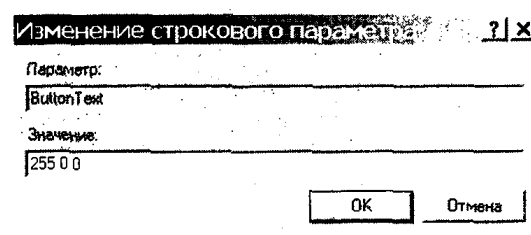


Рис. 1.32. Изменение значения параметра

4. Для того чтобы изменение вступило в силу, выйти из редактора реестра командой [*Файл-Выход*] и перезагрузить компьютер.

Категорически не рекомендуется изменять параметры реестра без четкого понимания производимых действий, так как ошибка при изменении реестра может серьезно повредить системе.

Если повреждена информация об устройствах, то соответствующий раздел реестра можно исправить или восстановить в том виде, который он имел в момент последнего удачного запуска компьютера. Для этого необходимо перезагрузить компьютер, в процессе загрузки нажать клавишу {F8} и выбрать вариант загрузки *Загрузка последней удачной конфигурации*. Современные версии операционной системы Windows содержат средство восстановления системы, которое может восстановить системный реестр, существовавший на определенную дату, если ранее была сохранена резервная копия этих данных.


Справочная система. Для удобства пользователя в состав операционной системы обычно входит также справочная система. Справочная система позволяет оперативно получить необходимую информацию как о функционировании операционной системы в целом, так и о работе ее отдельных модулей.

Контрольные вопросы

1. Каковы основные компоненты операционной системы и каково их назначение?

Компьютерный практикум



Windows-CD 

- 1.16. В операционной системе Windows выполнить практическое задание «Копирование файлов».
- 1.17. В операционной системе Windows выполнить практическое задание «Ознакомление с системным реестром Windows».



1.5.2. Загрузка операционной системы

Самотестирование компьютера. После включения компьютера или перезагрузки операционной системы процессор начинает считывать и выполнять микропрограммы тестирования компьютера и первого этапа загрузки операционной системы, которые хранятся в микросхеме **BIOS**.



Включение компьютера осуществляется включением питания (кнопка *Power* на системном блоке). Перезагрузка операционной системы производится нажатием кнопки *Reset* на системном блоке компьютера или одновременным нажатием комбинации клавиш $\{Ctrl\} + \{Alt\} + \{Del\}$ на клавиатуре.

Прежде всего начинает выполняться программа тестирования **POST**, которая проверяет работоспособность основных устройств компьютера: процессора, видеоадаптера, оперативной памяти, дисководов, контроллеров жестких дисков и клавиатуры. В случае обнаружения неисправностей выдаются диагностические сообщения в виде различных последовательностей коротких и длинных звуковых сигналов (например, в AWARD BIOS: 1 длинный и 3 коротких сигнала — не подключен монитор, 5 коротких — ошибка процессора и т. д.) или в виде текстовых сообщений. После успешной инициализации видеоплаты краткие диагностические сообщения выводятся на экран монитора.

BIOS Setup. Пользователь может установить новые параметры конфигурации компьютера и запомнить их в специальной микросхеме памяти, которая при выключенном компьютере питается от батарейки, установленной на системной плате. Для этого в процессе выполнения самотестирования обычно необходимо нажать клавишу *{Del}*. Загрузится системная утилита BIOS Setup, имеющая интерфейс в виде системы иерархических меню. В случае выхода из строя батарейки конфигурационные параметры теряются, и компьютер перестает нормально загружаться.

Загрузка операционной системы (рис. 1.33). Файлы операционной системы хранятся во внешней, долговременной памяти (на магнитных, оптических или флэш-дисках). Однако программы могут выполняться, только если они находятся в оперативной памяти, поэтому файлы операционной системы необходимо загрузить в оперативную память. Разрешение этого противоречия состоит в последовательной, поэтапной загрузке операционной системы.



Диск, на котором находятся файлы операционной системы и с которого производится ее загрузка, называется **системным**.

После проведения самотестирования специальная программа, содержащаяся в BIOS, начинает поиск загрузчика операционной системы. Современные версии BIOS позволяют загружать операционную систему не только с магнитных и оптических дисков, но и с USB флэш-дисков.

Если диск системный и программа-загрузчик оказывается на месте, то она загружается в оперативную память и ей передается управление работой компьютера. Программа ищет файлы операционной системы на системном диске и загружает их в оперативную память в качестве программных модулей.

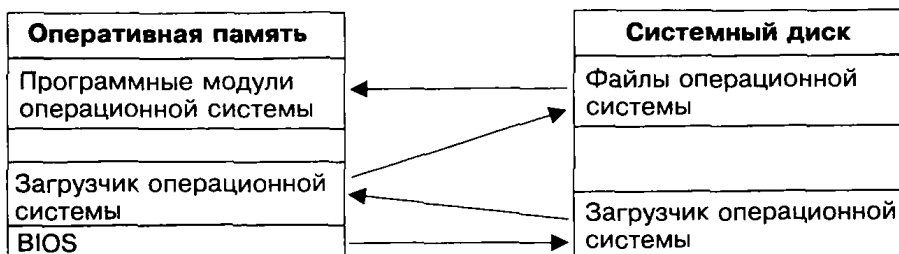


Рис. 1.33. Процесс загрузки операционной системы

Если системные диски в компьютере отсутствуют, на экране монитора появляется сообщение «Non system disk», и компьютер «зависает», т. е. загрузка операционной системы прекращается и компьютер остается неработоспособным.

В процессе загрузки можно выбрать в том числе вариант загрузки без графического интерфейса (для вывода меню вариантов загрузки нужно нажать клавишу {F8}). В случае использования интерфейса командной строки на экране появляется приглашение системы к вводу команд. Приглашение представляет собой последовательность символов, сообщающую о текущем диске и папке. Например, если загрузка операционной системы была произведена с диска C:, а операционная система была установлена в папку C:\WINDOWS, то появится приглашение:

```
C:\WINDOWS>
```

В случае загрузки графического интерфейса операционной системы команды могут вводиться с использованием элементов управления.

Контрольные вопросы


1. Каковы основные этапы загрузки операционной системы?

1.6. Защита информации от вредоносных программ

1.6.1. Вредоносные программы и антивирусные программы

Типы вредоносных программ. Вредоносными программами являются программы, наносящие вред данным и программам, хранящимся на компьютере. Основными типами вредоносных программ являются:

- компьютерные вирусы;
- сетевые черви;
- троянские программы;
- программы показа рекламы (от англ. adware) и программы-шпионы, занимающиеся сбором персональной информации о компьютере и пользователе (от англ. spyware);
- хакерские утилиты.

 Прообразом вредоносных программ была компьютерная игра «Дарвин», созданная в 1961 году в научно-исследовательских целях. Несколько компьютерных программ, названных «организмами», загружались в память ЭВМ, причем организмы, созданные одним игроком (т. е. принадлежащие к одному виду), должны были уничтожать представителей другого вида и захватывать оперативную память.

За создание, использование и распространение вредоносных программ в России и большинстве стран предусмотрена уголовная ответственность.

Антивирусные программы. Принцип работы антивирусных программ основан на проверке файлов, загрузочных секторов дисков и оперативной памяти и поиске в них известных и новых вирусов.

Для поиска известных вирусов используются сигнатуры, т. е. некоторые постоянные последовательности двоичного кода, специфичные для этого конкретного вируса. Если антивирусная программа обнаружит такую последовательность в каком-либо файле, то файл считается зараженным вирусом и подлежит лечению.

Для поиска новых вирусов используются алгоритмы эвристического сканирования, т. е. анализ последовательности команд в проверяемом объекте. Если «подозрительная» последовательность команд обнаруживается, то антивирусная программа выдает сообщение о возможном заражении объекта.

Большинство антивирусных программ сочетает в себе функции постоянной защиты (антивирусный монитор) и функции защиты по требованию пользователя (антивирусный сканер).

Антивирусный монитор запускается автоматически при старте операционной системы и работает в качестве фонового системного процесса, проверяя на вредоносность совершаемые другими программами действия. Основная задача антивирусного монитора состоит в обеспечении максимальной защиты от вредоносных программ при минимальном замедлении работы компьютера.

Антивирусный сканер запускается по заранее выбранному расписанию или в произвольный момент пользователем. Антивирусный сканер производит поиск вредоносных программ в оперативной памяти, а также на жестких и сетевых дисках компьютера.

К недостаткам антивирусных программ можно отнести большие размеры используемых ими антивирусных баз данных, которые должны содержать информацию о максимально возможном количестве вирусов (в настоящее время десятках тысяч), что, в свою очередь, приводит к относительно небольшой скорости поиска вирусов.

Признаки заражения компьютера. Есть ряд признаков, свидетельствующих о проникновении на компьютер вредоносных программ:

- вывод на экран непредусмотренных сообщений или изображений;
- подача непредусмотренных звуковых сигналов;
- неожиданное открытие и закрытие лотка CD/DVD дисковода;
- произвольный запуск на компьютере каких-либо программ;
- частые «зависания» и сбои в работе компьютера;
- медленная работа компьютера при запуске программ;
- исчезновение или изменение файлов и папок;
- частое обращение к жесткому диску (часто мигает лампочка на системном блоке);
- «зависание» или неожиданное поведение браузера (например, окно программы невозможно закрыть).

Кроме того, есть некоторые характерные признаки поражения сетевым вирусом через электронную почту:

- друзья или знакомые говорят о полученных от вас сообщениях, которые вы не отправляли;
- в вашем почтовом ящике находится большое количество сообщений без обратного адреса и заголовка.

Действия при наличии признаков заражения компьютера. Прежде чем предпринимать какие-либо действия, необходимо сохранить результаты работы на внешнем носителе (дискете, CD- или DVD-диске, флэш-карте и пр.). Далее необходимо:

- отключить компьютер от локальной сети и Интернета, если он к ним был подключен;
- если симптом заражения состоит в том, что невозможно загрузиться с жесткого диска компьютера (компьютер выдает ошибку, когда вы его включаете), попробовать загрузиться в режиме защиты от сбоев или с диска аварийной загрузки Windows;
- запустить антивирусную программу.

Контрольные вопросы

1. К каким последствиям может привести заражение компьютерными вирусами?
2. Какие существуют признаки заражения компьютера вирусом?
3. Что необходимо сделать в первую очередь в случае заражения компьютера вирусом?

1.6.2. Компьютерные вирусы и защита от них

Обязательным свойством компьютерного вируса является способность к размножению (самокопированию). Вирусы могут также незаметно для пользователя внедряться в исполняемые файлы, загрузочные секторы дисков и документы. Название «вирус» по отношению к компьютерным программам пришло из биологии именно по признаку способности к саморазмножению.

После заражения компьютера вирус может начать выполнение вредоносных действий и распространение своих копий, а также заставить компьютер выполнять какие-либо действия. Активизация вируса может быть связана с различными событиями (наступлением определенной даты или дня недели, запуском программы, открытием документа и т. д.).



Компьютерные вирусы являются вредоносными программами, которые могут «размножаться» и скрытно внедрять свои копии в исполнимые файлы, загрузочные секторы дисков и документы. Активизация компьютерного вируса может вызывать уничтожение программ и данных.

Разнообразны последствия действия вирусов. По величине вредных воздействий вирусы можно разделить на:

- неопасные, влияние которых ограничивается уменьшением свободной памяти на диске, графическими, звуковыми и другими внешними эффектами;
- опасные, которые могут привести к сбоям и «зависаниям» при работе компьютера;
- очень опасные, активизация которых может привести к потере программ и данных (изменению или удалению файлов и каталогов), форматированию винчестера и т. д.

В настоящее время известно несколько десятков тысяч вирусов, заражающих компьютеры различных операционных

систем. По способу сохранения и исполнения своего кода вирусы можно разделить на **загрузочные, файловые, макро-вирусы и скрипт-вирусы**.

Загрузочные вирусы. Загрузочные вирусы заражают загрузочный сектор гибкого или жесткого диска. Принцип действия загрузочных вирусов основан на алгоритмах запуска операционной системы при включении или перезагрузке компьютера. После необходимых тестов установленного оборудования программа системной загрузки считывает первый физический сектор загрузочного диска (гибкого, жесткого, оптического или флэш-диска в зависимости от параметров, установленных в BIOS Setup) и передает на него управление.

При заражении дисков загрузочные вирусы «подставляют» свой код вместо программы, получающей управление при загрузке системы, и отдают управление не оригинальному коду загрузчика, а коду вируса. При инфицировании диска вирус в большинстве случаев переносит оригинальный загрузочный сектор в какой-либо другой сектор диска (например, в первый свободный).



Первая эпидемия загрузочного компьютерного вируса произошла в 1986 году, когда вирус «Brain» «заражал» загрузочный сектор дискет для персональных компьютеров. Вирус «Brain» являлся также и первым вирусом-невидимкой, так как при попытке обнаружения зараженного загрузочного сектора вирус незаметно «подставлял» его незараженный оригинал.

Профилактическая защита от таких вирусов состоит в отказе от загрузки операционной системы с гибких дисков и установке в BIOS вашего компьютера защиты загрузочного сектора от изменений. С помощью программы BIOS Setup можно провести настройку BIOS таким образом, что будет запрещена (заблокирована) любая запись в загрузочный сектор диска и компьютер будет защищен от заражения загрузочными вирусами.

Файловые вирусы. Файловые вирусы различными способами внедряются в исполнимые файлы (командные файлы *.bat, программы *.exe, системные файлы *.com и *.sys, программные библиотеки *.dll и др.) и обычно активизируются при их запуске. После запуска зараженного файла вирус находится в оперативной памяти компьютера и является активным (т. е. может заражать другие файлы) вплоть до момента выключения компьютера или перезагрузки операционной системы.

По способу заражения файловые вирусы разделяют на:

- перезаписывающие вирусы, которые записывают свой код вместо кода программы, не изменяя названия исполнимого файла. При запуске программы выполняется код вируса, а не сама программа;
- вирусы-компаньоны, которые, как и перезаписывающие вирусы, создают свою копию на месте заражаемой программы, но в отличие от перезаписываемых не уничтожают оригинальный файл, а переименовывают или перемещают его. При запуске программы вначале выполняется код вируса, а затем управление передается оригинальной программе;
- паразитические вирусы — это файловые вирусы, изменяющие содержимое файла, добавляя в него свой код. Код может внедряться в начало, середину или конец программы и выполняется перед, вместе или после программы. При этом зараженная программа сохраняет полную или частичную работоспособность.



В 1999 году произошла эпидемия очень опасного файлового вируса Win95.CIH, названного «Чернобыль» из-за даты активации 26 апреля. Вирус уничтожил данные на жестком диске, а на системных платах стирал содержимое BIOS, что приводило к необходимости их замены.

Практически все загрузочные и файловые вирусы резидентны, т. е. они находятся в оперативной памяти компьютера, и в процессе работы пользователя могут осуществлять опасные действия (стирать данные на дисках, изменять названия и другие атрибуты файлов и т. д.). Лечение от резидентных вирусов затруднено, так как даже после удаления зараженных файлов с дисков вирус остается в оперативной памяти и возможно повторное заражение файлов.

Профилактическая защита от файловых вирусов состоит в том, что не рекомендуется запускать на исполнение файлы, полученные из сомнительного источника и предварительно не проверенные антивирусными программами.

Макро-вирусы. Наибольшее распространение получили макро-вирусы для интегрированного офисного приложения Microsoft Office (Word, Excel, PowerPoint и Access). Макро-вирусы фактически являются макрокомандами (макросами) на встроенном языке программирования Visual Basic for Applications (VBA), которые помещаются в документ.

При работе с документом пользователь выполняет различные действия: открывает документ, сохраняет, печатает, за-

крывает и т. д. При этом приложение ищет и выполняет соответствующие стандартные макросы. Макро-вирусы содержат стандартные макросы, вызываются вместо них и заражают каждый открываемый или сохраняемый документ. Вредные действия макро-вирусов реализуются с помощью встроенных макросов (вставки текстов, запрета выполнения команд меню приложения и т. д.).

Макро-вирусы являются **ограниченно резидентными**, т. е. они находятся в оперативной памяти и заражают документы, пока открыто приложение. Кроме того, макро-вирусы заражают шаблоны документов и поэтому активизируются уже при запуске зараженного приложения.



В августе 1995 года началась эпидемия первого макро-вируса «Concept» для текстового процессора Microsoft Word. Макро-вирус «Concept» до сих пор имеет широкое распространение, и на сегодняшний момент известно около 100 его модификаций.

Профилактическая защита от макро-вирусов состоит в предотвращении запуска вируса. При открытии документа в приложениях Microsoft Office сообщается о присутствии в них макросов (потенциальных вирусов) и предлагается запретить их загрузку. Выбор запрета на загрузку макросов надежно защитит ваш компьютер от заражения макро-вирусами, однако отключит и полезные макросы, содержащиеся в документе.

Скрипт-вирусы. Особой разновидностью вирусов являются активные элементы (программы) на языках JavaScript или VBScript, которые могут содержаться в файлах Web-страниц. Заражение локального компьютера происходит при их передаче по Всемирной паутине с серверов Интернета в браузер локального компьютера.



В ноябре 1998 года появился первый скрипт-вирус VBScript.Rabbit, заражающий скрипты Web-страниц, а через полтора года, в мае 2000 года грянула глобальная эпидемия скрипт-вируса «LoveLetter». Сейчас этот тип вирусов прочно удерживает первое место в списке наиболее распространенных и опасных вирусов.

Профилактическая защита от скрипт-вирусов состоит в том, что в браузере можно запретить получение активных элементов на локальный компьютер.

Практическое задание «Защита от компьютерных вирусов». С помощью антивирусной программы (например, AntiVir Personal Edition) проверить компьютер на наличие вирусов и при их обнаружении вылечить или удалить зараженные файлы.



Защита от компьютерных вирусов

Прежде всего, необходимо через Интернет обновить саму антивирусную программу и вирусную базу данных.

1. Соединиться с Интернетом.

На панели задач в контекстном меню значка антивирусной программы AntiVir Personal Edition выбрать пункт *Start Update*.

В появившемся диалоговом окне (рис. 1.34) будет отображаться процесс обновления антивирусной программы и вирусной базы данных.

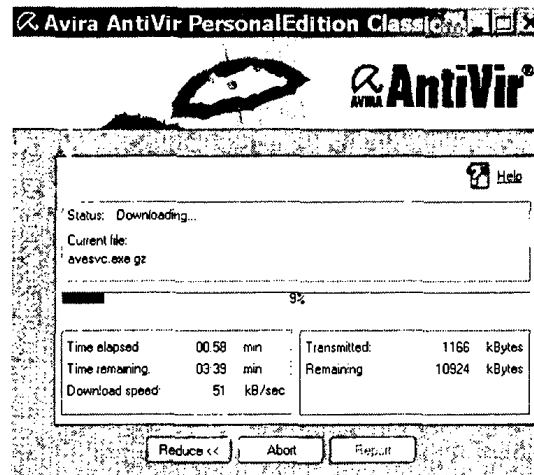


Рис. 1.34. Обновление антивирусной программы

Настроим параметры антивирусного монитора (Guard) и сканера (Scanner).

2. В контекстном меню значка антивирусной программы выбрать пункт *Configure AntiVir*.

В появившемся диалоговом окне (рис. 1.35) выбрать экспертный режим, установив флажок *Expert mode*.

В левой части окна в иерархическом меню выбрать пункт *Guard*, а в правой установить необходимые параметры антивирусного монитора.

Для выбора типов проверяемых файлов щелкнуть по кнопке *File extensions* и в появившемся раскрывающемся списке добавить или удалить типы файлов.

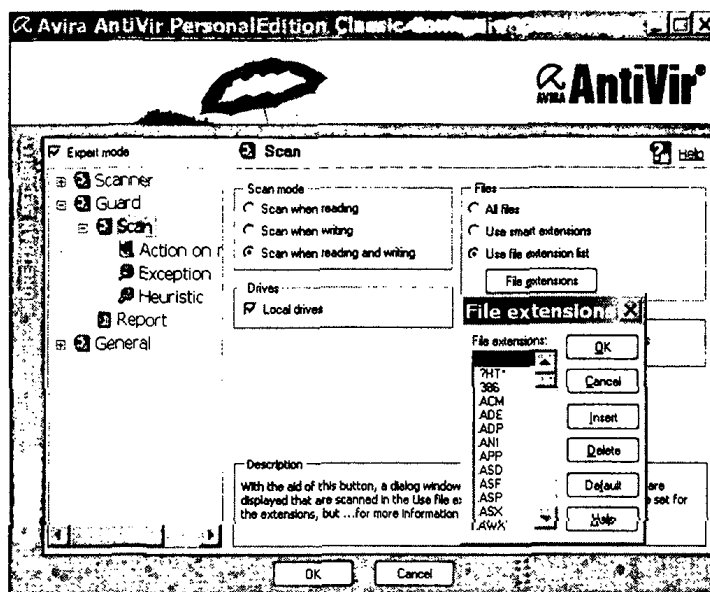


Рис. 1.35. Настройка параметров антивирусного монитора (Guard)

3. В иерархическом меню выбрать пункт *Scan* (рис. 1.36), а в правой части окна установить необходимые параметры антивирусного сканера.

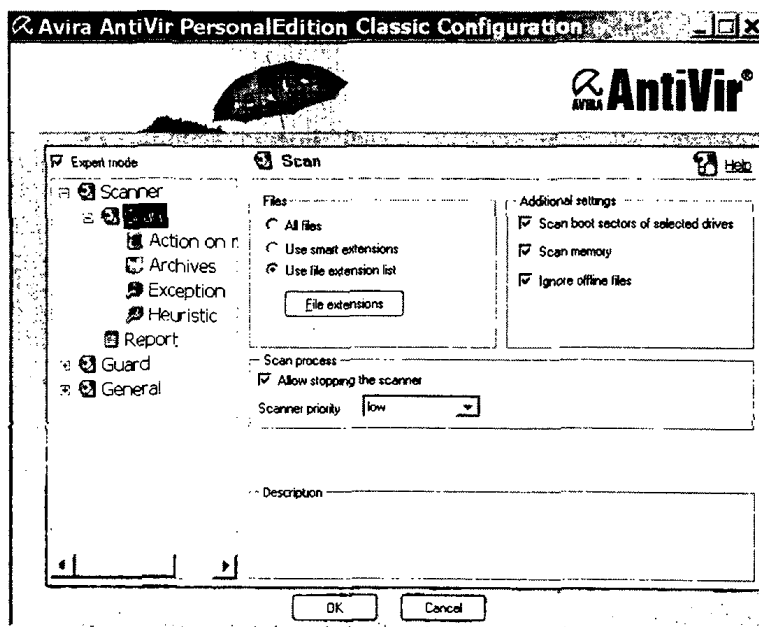


Рис. 1.36. Настройка параметров антивирусного сканера (Scanner)

Проведем проверку на вирусы выбранных дисков.

4. В контекстном меню значка антивирусной программы выбрать пункт *Start AntiVir*.

В появившемся диалоговом окне (рис. 1.37) выбрать вкладку *Scanner*.

В иерархической файловой системе отметить флажком диски, выбранные для проверки.

Щелкнуть по кнопке *Start scan*.

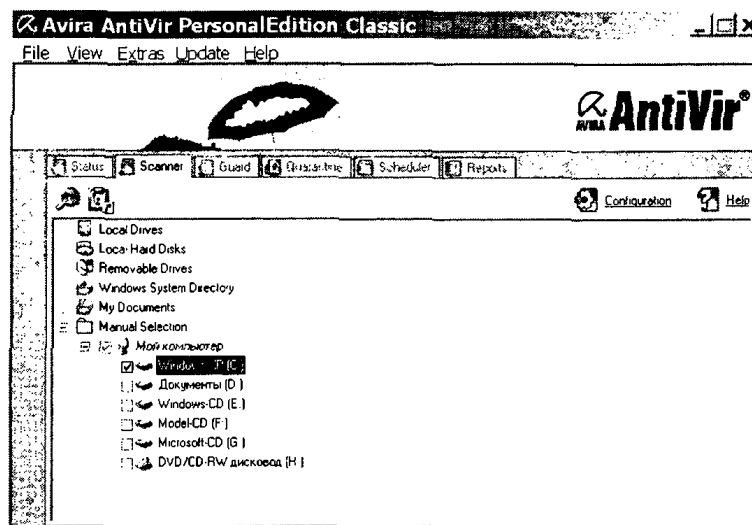


Рис. 1.37. Проверка на вирусы выбранных дисков


5. Появится диалоговое окно, в котором будет отображаться процесс антивирусной проверки файлов на выбранном диске.


Для получения результатов проверки щелкнуть по кнопке *Report*.

Контрольные вопросы

1. Какие типы компьютерных вирусов существуют, чем они отличаются друг от друга и какова должна быть профилактика заражения?
2. Почему даже чистая отформатированная дискета может стать источником заражения вирусом?

Компьютерный практикум

Windows-CD 

- 1.18. С помощью антивирусной программы (например, AntiVir Personal Edition) выполнить практическое задание «Защита от компьютерных вирусов». 
- 1.19. С использованием Вирусной энциклопедии ознакомиться с классификацией вирусов и методами антивирусной защиты.

<http://www.viruslist.com>

1.6.3. Сетевые черви и защита от них

К сетевым червям (от англ. worm) относятся вредоносные программы, распространяющие свои копии по локальным и/или глобальным сетям. Для своего распространения сетевые черви используют разнообразные сервисы глобальных и локальных компьютерных сетей: Всемирную паутину, электронную почту, интерактивное общение, файлообменные сети и т. д.

Основным признаком, по которому типы червей различаются между собой, является способ распространения червя — каким способом он передает свою копию на удаленные компьютеры. Однако многие сетевые черви используют более одного способа распространения своих копий по компьютерам локальных и глобальных сетей.



Сетевые черви являются вредоносными программами, которые проникают на компьютер, используя сервисы компьютерных сетей. Активизация сетевого червя может вызывать уничтожение программ и данных, а также похищение персональных данных пользователя.

Почтовые черви. Почтовые черви для своего распространения используют электронную почту. Червь отправляет либо свою копию в виде вложения в электронное письмо, либо ссылку на свой файл, расположенный на каком-либо сетевом ресурсе. В первом случае код червя активизируется при открытии (запуске) зараженного вложения, во втором — при открытии ссылки на зараженный файл. В обоих случаях эффект одинаков — активизируется код червя.

Лавинообразная цепная реакция распространения почтового червя базируется на том, что червь после заражения компьютера начинает рассылать себя по всем адресам электронной почты, которые имеются в адресной книге пользователя.

Профилактическая защита от почтовых червей состоит в том, что не рекомендуется открывать вложенные в почтовые сообщения файлы, полученные из сомнительных источников.

Черви, использующие «уязвимости» программного обеспечения. Червь ищет в сети компьютеры, на которых используются операционная система и приложения, содержащие критические уязвимости. Для заражения уязвимых компьютеров червь посылает специально оформленный сетевой пакет или запрос, в результате чего код (или часть кода) червя проникает на компьютер-жертву. Если сетевой пакет содержит только часть кода червя, он затем скачивает основной файл и запускает его на исполнение на зараженном компьютере.

Профилактическая защита от таких червей состоит в том, что рекомендуется своевременно скачивать из Интернета и устанавливать обновления системы безопасности операционной системы и приложений.

Черви, использующие файлообменные сети. Механизм работы подобных червей достаточно прост — для внедрения в файлообменную сеть червя достаточно скопировать себя в папку обмена файлами на одном из компьютеров. Всю остальную работу по распространению червя файлообменная сеть берет на себя — при поиске файлов в сети она сообщит удаленным пользователям о данном файле-черве и предоставит его для скачивания.



В сентябре 2001 года началась стремительное «расползание» сетевого червя «Nimda», который мог атаковать компьютеры сразу несколькими способами: через сообщения электронной почты, через открытые ресурсы локальных сетей, а также используя уязвимости в системе безопасности операционной системы серверов Интернета.

Сетевые черви кроме вредоносных действий, которыми обладают и классические компьютерные вирусы, могут выполнять шпионскую функцию троянских программ.



1.6.4. Троянские программы и защита от них

Практическое задание «Защита от сетевых червей». С помощью антивирусной программы (например, avast!) проверить компьютер на заражение сетевыми червями и при их обнаружении вылечить или удалить зараженные файлы.



Защита от сетевых червей

Прежде всего необходимо через Интернет обновить саму антивирусную программу и вирусную базу данных.

1. Соединиться с Интернетом.

На панели задач в контекстном меню значка антивирусной программы avast! выбрать сначала пункт [Обновление-Обновление программы], а затем пункт [Обновление-Обновление базы данных iAVS].

В появляющихся диалоговых окнах будет отображаться процесс обновления антивирусной программы и вирусной базы данных.

Настроим параметры резидентного антивирусного монитора (авторы программы называют его сканером доступа). Можно отдельно настроить параметры сканера доступа на наличие сетевых червей в электронной почте, Web-страницах, мгновенных сообщениях интерактивного общения, локальных и файлообменных сетях (авторы программы называют эти сервисы компьютерных сетей провайдерами).

2. В контекстном меню значка антивирусной программы выбрать пункт *Настройка сканера доступа*.

В левой части появившегося диалогового окна (рис. 1.38) выбрать провайдера, например *Электронная почта*.

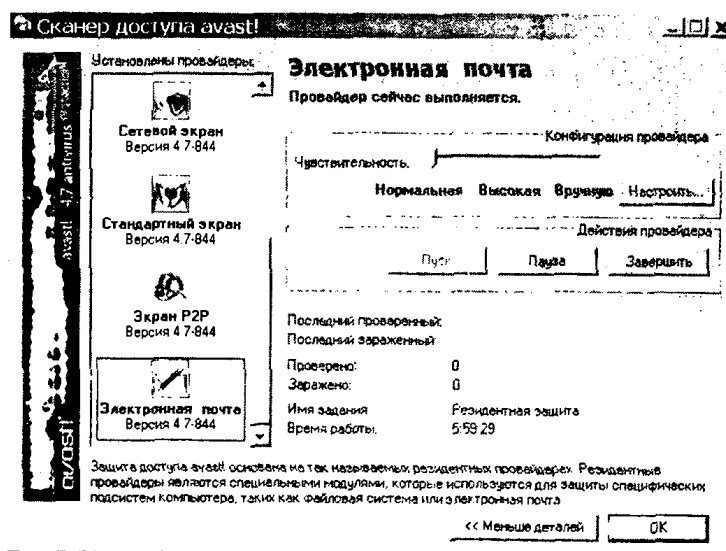


Рис. 1.38. Настройка параметров антивирусного монитора (сканера доступа)

В правой части окна установить необходимые параметры проверки входящей почты.

Проверим входящую почту на наличие почтовых червей.

3. Запустить почтовую программу и начать получение входящих сообщений.

В случае обнаружения в сообщении почтового червя раздастся голосовое предупреждение и будет выведено диалоговое окно *Обнаружен вирус!* (рис. 1.39).

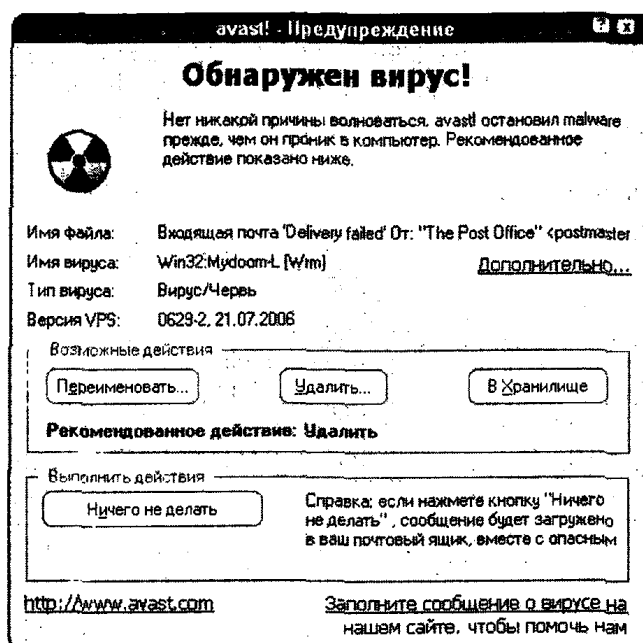


Рис. 1.39. Обнаружение почтового червя

В окне будет указано, какое сообщение инфицировано, имя вируса (например, *Win32.Mydoom-L*) и тип вируса (например, *Вирус/Червь*).

Пользователь может выбрать одно из следующих действий щелчком по соответствующей кнопке: *Переименовать*, *Удалить*, *В хранилище*, *Ничего не делать*.

Проведем проверку на вирусы выбранных дисков.

4. В контекстном меню значка антивирусной программы выбрать пункт *Запустить антивирус avast!*.

В появившемся диалоговом окне с помощью кнопок справа выбрать диски для сканирования. Щелкнуть по кнопке *Запустить*.

В диалоговом окне будет отображаться процесс антивирусной проверки файлов на выбранном диске (рис. 1.40).

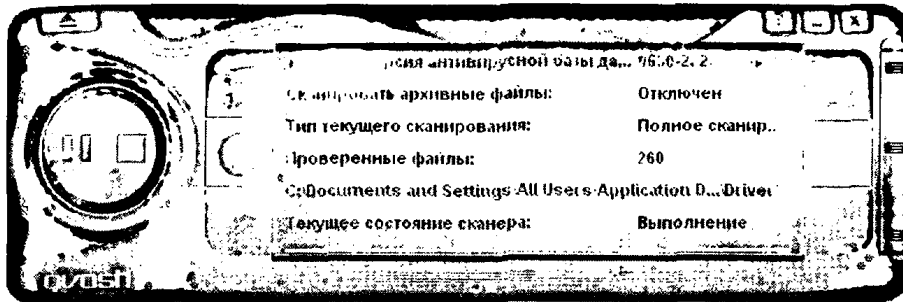



Рис. 1.40. Проверка на вирусы выбранных дисков

- Для ознакомления с результатами в контекстном меню значка антивирусной программы выбрать пункт *Просмотр журнала avast!*.

Контрольные вопросы

- Какие типы сетевых червей существуют, чем они отличаются друг от друга и какова должна быть профилактика заражения?

Компьютерный практикум

Windows-CD 

- С помощью антивирусной программы (например, avast!) выполнить практическое задание «Защита от сетевых червей».



1.6.4. Троянские программы и защита от них

Троянские программы осуществляют не санкционированные пользователем действия по сбору и передаче информации злоумышленнику, а также ее разрушение или злонамеренную модификацию. Кроме того, троянские программы могут вызывать нарушение работоспособности компьютера или незаметно для пользователя использовать ресурсы компьютера в целях злоумышленника.

Троянские программы обычно проникают на компьютер как сетевые черви, а различаются между собой по тем действиям, которые они производят на зараженном компьютере.



Троянская программа, троянец (от англ. trojan) — вредоносная программа, которая выполняет несанкционированную пользователем передачу управления компьютером удаленному пользователю, а также действия по удалению, модификации, сбору и пересылке информации третьим лицам.

Троянские утилиты удаленного администрирования. Троянские программы этого класса являются утилитами удаленного администрирования компьютеров в сети. Утилиты скрытого управления позволяют принимать или отсылать файлы, запускать и уничтожать их, выводить сообщения, стирать информацию, перезагружать компьютер и т. д.

При запуске троянец устанавливает себя в системе и затем следит за ней, при этом пользователю не выдается никаких сообщений о действиях троянской программы в системе. В результате «пользователь» этой троянской программы может и не знать о ее присутствии в системе, в то время как его компьютер открыт для удаленного управления.

Троянские программы данного типа являются одним из самых опасных видов вредоносного программного обеспечения, поскольку в них заложена возможность самых разнообразных злоумышленных действий, в том числе они могут быть использованы для обнаружения и передачи конфиденциальной информации.



В 2003 году широкое распространение получила троянская программа Backdoor.Win32.BO, которая осуществляет следующие действия:

- высылает имена компьютера, пользователя и информацию о системе: тип процессора, размер памяти, версию системы, информацию об установленных устройствах;
- посылает/принимает, уничтожает, копирует, переименовывает, исполняет любой файл;
- отключает пользователя от сети;
- «завешивает» компьютер;
- читает или модифицирует системный реестр.

Троянские программы, ворующие информацию. Такие троянские программы воруют различную информацию с зараженного компьютера. При запуске они ищут файлы, хранящие конфиденциальную информацию о пользователе (банковские реквизиты, пароли доступа к Интернету и др.) и отправляют ее по указанному в коде троянца электронному адресу или адресам.

Троянцы данного типа также сообщают информацию о зараженном компьютере (размер памяти и дискового пространства, версию операционной системы, IP-адрес и т. п.). Некоторые троянцы воруют регистрационную информацию к программному обеспечению.

Троянские программы — инсталляторы вредоносных программ. Троянские программы этого класса скрытно инсталлируют другие вредоносные программы и используются для «подсовывания» на компьютер-жертву вирусов или других троянских программ. Загруженные без ведома пользователя из Интернета программы затем либо запускаются на выполнение, либо включаются троянцем в автозагрузку операционной системы.

Троянские программы — шпионы. Данные троянцы осуществляют электронный шпионаж за пользователем зараженного компьютера: вводимая с клавиатуры информация, снимки экрана, список активных приложений и действия пользователя с ними сохраняются в каком-либо файле на диске и периодически отправляются злоумышленнику.

Троянские программы этого типа часто используются для кражи информации пользователей различных систем онлайн-новых платежей и банковских систем.

Троянские программы часто изменяют записи системного реестра операционной системы, поэтому для их удаления необходимо в том числе восстановление системного реестра.

Практическое задание «Защита от троянских программ». С помощью программы восстановления системы (например, CCleaner) исправить ошибки системного реестра.



Защита от троянских программ

Исправим записи системного реестра, в которые троянские программы могли внести изменения.

1. Запустить программу восстановления системы CCleaner. В левой части окна приложения (рис. 1.41) щелкнуть по кнопке *Неполадки*.

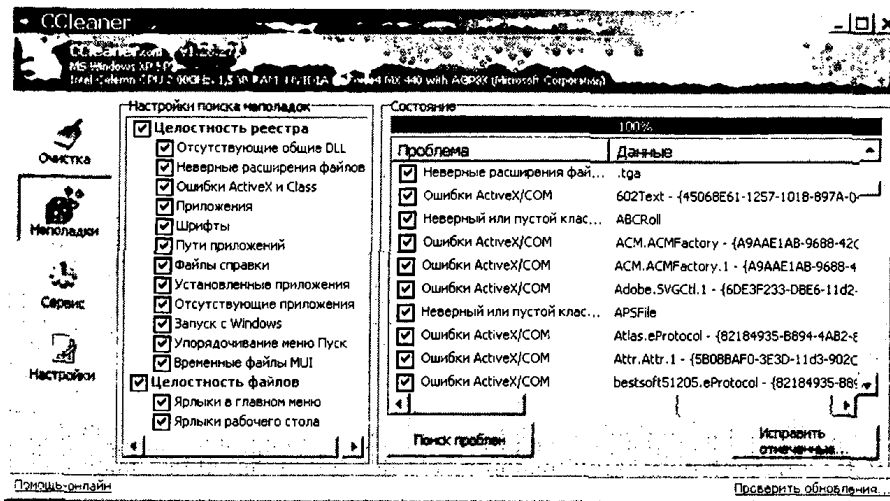


Рис. 1.41. Исправление записей системного реестра

В списке *Настройка поиска неполадок* отметить флажками нужные пункты.

Щелкнуть по кнопке *Поиск проблем*, в правой части окна будут показаны проблемы реестра и соответствующие записи реестра.

Отметить флажками нужные записи и щелкнуть по кнопке *Исправить отмеченные...*


Произойдет исправление выбранных записей системного реестра.

Контрольные вопросы

1. Какие типы троянских программ существуют и каковы их вредоносные действия?

Компьютерный практикум



Windows-CD 

- 1.21. С помощью программы восстановления системы (например, CCleaner) выполнить практическое задание «Защита от троянских программ».



1.6.5. Рекламные и шпионские программы и защита от них

Рекламные программы. Рекламные программы (от англ. **Adware: Advertisement** — реклама и **Software** — программное обеспечение) встраивают рекламу в основную полезную программу. Часто рекламные программы входят в состав официально поставляемых условно бесплатных версий программного обеспечения.

Реклама демонстрируется пользователю в процессе работы основной программы в виде графических баннеров или бегущей строки. Обычно после покупки и/или регистрации основной программы рекламная вставка удаляется и показ рекламы прекращается.

Шпионские программы. Шпионские программы (от англ. **Spyware: Spy** — шпион и **Software** — программное обеспечение) скрытно собирают различную информацию о пользователе компьютера и затем отправляют ее злоумышленнику.

Эти программы иногда проникают на компьютер под видом рекламных программ и не имеют возможности деинсталляции пользователем без нарушения функционирования использующей их программы. Иногда шпионские программы обнаруживаются в распространенных программных продуктах известных на рынке производителей.



В марте 2005 года под видом поисковой панели для браузера Internet Explorer начала распространяться рекламно-шпионская программа «mwsbar». Программа регистрирует себя в системном реестре и добавляет в автозагрузку, что приводит к изменению настроек браузера и перенаправлению результатов поиска в Интернете на сайт злоумышленника.

Практическое задание «Защита от рекламных и шпионских программ». С помощью программы удаления рекламных и шпионских программ (например, Ad-Adware) очистить компьютер от adware и spyware программ.



Защита от рекламных и шпионских программ

1. Запустить программу Ad-Adware, которая удаляет рекламные и шпионские программы.

В окне приложения (рис. 1.42) щелкнуть по кнопке *Start*.

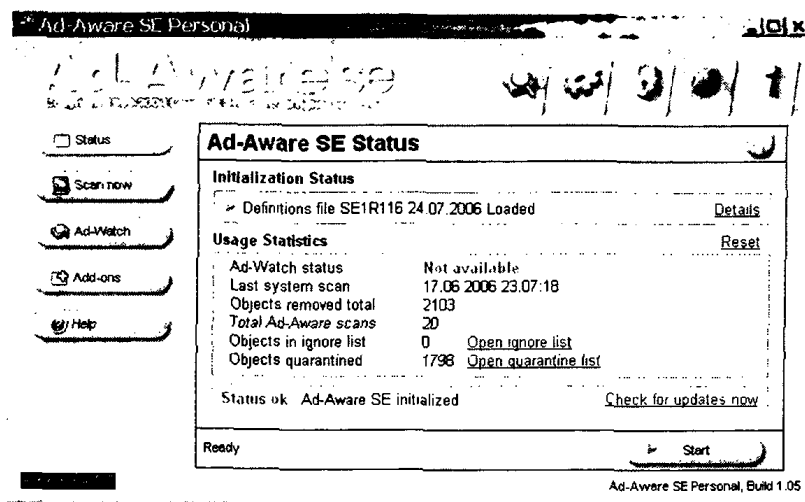


Рис. 1.42. Окно программы Ad-Aware

- В появившемся диалоговом окне (рис. 1.43) выбрать тип сканирования (например, *Perform smart system scan* — *Интеллектуальное сканирование*). Щелкнуть по кнопке *Next*.

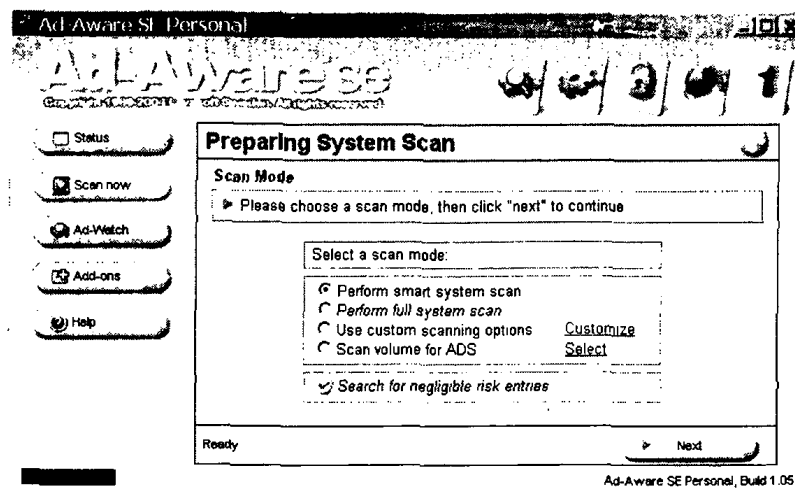


Рис. 1.43. Выбор типа сканирования

- В течение некоторого времени будет происходить сканирование, процесс которого будет отображаться в диалоговом окне.

4. С результатами сканирования можно ознакомиться в следующем диалоговом окне с использованием вкладок. Активизировать вкладку *Critical Objects* (рис. 1.44) и в контекстном меню выбрать пункт *Select all objects* — *Выделить все объекты*. В данном случае будут выделены файлы cookies (см. ниже).

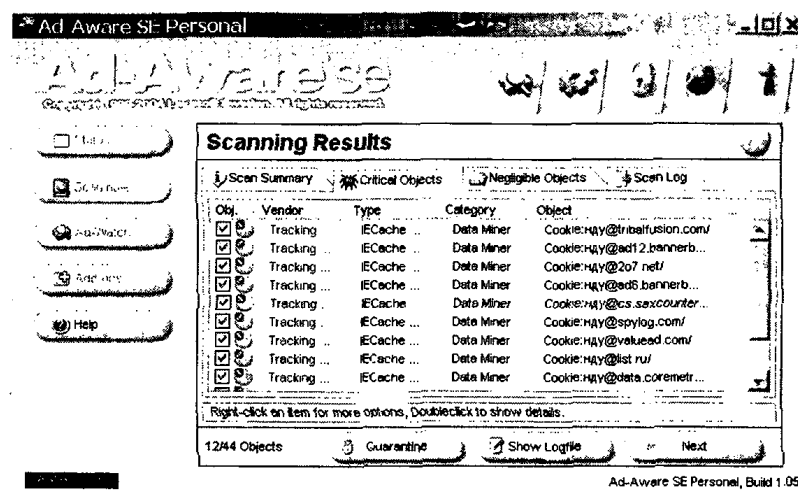


Рис. 1.44. Список найденных критических объектов

5. Активизировать вкладку *Negligible Objects* (рис. 1.45) и в контекстном меню выбрать пункт *Select all objects* — *Выделить все объекты*.

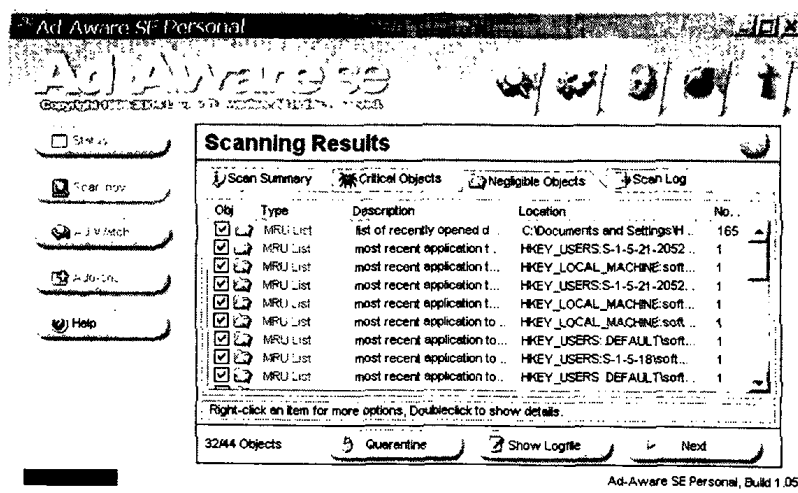


Рис. 1.45. Список найденных подозрительных объектов

В данном случае будут выделены файлы с подозрением на adware и spyware.

Щелкнуть по кнопке *Next*.

6. В появившемся диалоговом окне можно подтвердить удаление найденных критических и подозрительных объектов (adware, spyware и cookies).

Куки. Куки (от англ. cookies — домашнее печенье) — небольшой текстовый файл, помещаемый Web-сервером на локальный компьютер пользователя. Файлы cookies могут храниться в оперативной памяти (сеансовые файлы cookies) или записываться на жесткий диск (постоянные файлы cookies). Файлы cookies не могут быть использованы для запуска программного кода (запуска программ) или для заражения компьютера вирусами.

Cookies применяются для сохранения данных, специфичных для данного пользователя. При вводе регистрационных данных файлы cookies помогают серверу упростить процесс сохранения персональных данных, связанных с текущим пользователем. Если пользователь Интернет-магазина ранее указывал адрес для доставки счетов или товара, вместо повторного ввода этих данных можно указать пароль, позволяющий автоматически заполнить соответствующие поля в форме заказа.

Браузеры позволяют включать и отключать использование файлов cookies, а также выполнять прием файлов cookies только после подтверждения со стороны пользователя.

Практическое задание «Упорядочение использования файлов cookies». В браузере установить уровень защиты локального компьютера от файлов cookies.



Упорядочение использования файлов cookies

1. Запустить браузер Internet Explorer и ввести команду [*Сервис-Свойства обозревателя...*].
В появившемся диалоговом окне *Свойства обозревателя* выбрать вкладку *Конфиденциальность* (рис. 1.46).
С помощью вертикального ползунка установить уровень конфиденциальности от значения *Принимать все cookies* до уровня *Блокировать все cookies*.
2. Запустить браузер SeaMonkey и ввести команду [*Инструменты-Менеджер Cookies ...*].
С помощью пунктов меню установить уровень защиты от cookies.

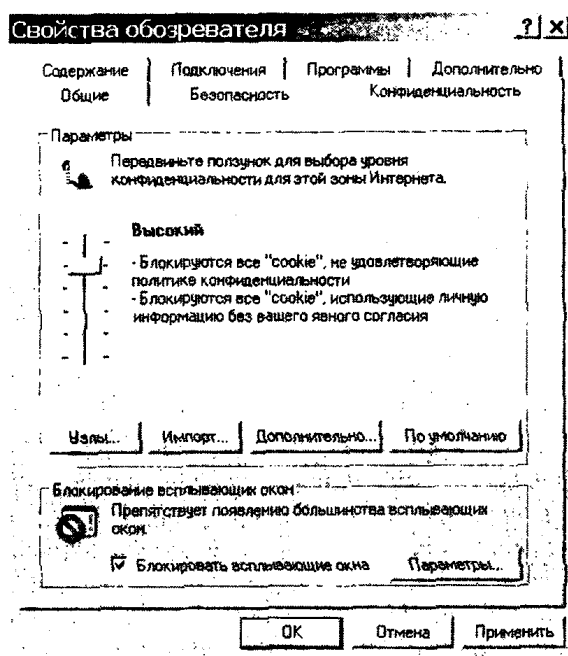



Рис. 1.46. Защита от cookies в браузере Internet Explorer

Контрольные вопросы

1. Каковы вредоносные действия рекламных и шпионских программ?
2. Что такое куки (cookies)? В чем состоит их польза и вред?

Компьютерный практикум

Windows-CD 

- 1.22. С помощью программы удаления рекламных и шпионских программ (например, Ad-Adware) выполнить практическое задание «Защита от рекламных и шпионских программ».
- 1.23. В браузерах выполнить практическое задание «Упорядочение использования файлов cookies».



1.6.6. Спам и защита от него

Спам (от англ. spam) — это массовая автоматическая рассылка рекламных электронных сообщений, со скрытым или фальсифицированным обратным адресом. Спам распространяется по компьютерным сетям с использованием электронной почты и систем интерактивного общения (типа ICQ), а также по мобильным сетям с использованием службы SMS-сообщений.



Спам — массово рассылаемая корреспонденция рекламного или иного характера, отправляемая людям, не выразившим желание ее получать. В первую очередь термин «спам» относится к рекламным электронным письмам.

Спам приходит потому, что электронный адрес получателя стал известен спамерам (рассыльщикам спама). Чаще всего владелец почтового ящика сам указывает электронный почтовый адрес при регистрации на каком-либо сайте и его обнаруживает специальный робот, «бродящий» по сайтам наподобие индексирующего робота поисковых систем.

Спамеры стремятся получить подтверждение, что почтовый адрес действительно используется (в этом случае поток спама может увеличиться многократно). Чтобы убедиться, что спамовое сообщение получено и прочитано, спамеры применяют различные уловки:

- требуется подтверждение о получении сообщения;
- предлагается активизировать ссылку на Web-страницу, на которой предлагается получить дополнительную информацию;
- предлагается отменить подписку на эту рассылку, по-слав письмо по указанному адресу.

Рекламный спам. Некоторые компании, занимающиеся легальным бизнесом, рекламируют свои товары или услуги с помощью спама. Они могут осуществлять его рассылку самостоятельно, но чаще заказывают ее тем компаниям (или лицам), которые на этом специализируются. Привлекательность такой рекламы заключается в ее сравнительно низкой стоимости и большом охвате потенциальных клиентов.

С помощью спама часто рекламируют продукцию, о которой нельзя сообщить другими способами, например оружие, порнографию, лекарственные средства с ограничениями по обороту, ворованную информацию (базы данных), контрафактное программное обеспечение и т. п.

«Нигерийские письма». Иногда спам используется для того, чтобы выманить деньги у получателя письма. Наиболее распространенный способ получил название «нигерийские письма», потому что большое количество таких писем приходило из Нигерии. Такое письмо содержит сообщение о том, что получатель письма может получить большую сумму денег, а отправитель может ему в этом помочь. Затем отправитель письма просит перевести ему немного денег под предлогом, например, оформления документов или открытия счета. Выманивание этой суммы и является целью мошенников.

Фишинг. Фишинг (от англ. fishing — рыбалка) — еще один способ мошенничества путем обмана пользователей. Он представляет собой попытку выманить у получателя письма данные, которые можно использовать для получения выгоды: номера его кредитных карточек или пароли доступа к системам онлайн-платежей. Такое письмо обычно маскируется под официальное сообщение от администрации банка. В нем говорится, что получатель должен подтвердить сведения о себе, иначе его счет будет заблокирован, и приводится адрес сайта (принадлежащего спамерам) с формой, которую надо заполнить. Среди данных, которые требуется сообщить, присутствуют и те, которые нужны мошенникам. Для того чтобы жертва не догадалась об обмане, оформление этого сайта имитирует оформление официального сайта банка.

Защита от спама. В силу массового характера спамовые почтовые рассылки затрудняют работу информационных систем и ресурсов, создавая для них бесполезную нагрузку. Пользователи сети вынуждены ежедневно тратить время на обработку бесполезных рекламных сообщений, а провайдерам спам приносит неудобства вследствие повышения нагрузки на почтовые серверы и каналы связи.

Для борьбы со спамом используются антиспамовые фильтры, которые могут быть установлены как на локальных компьютерах пользователей, так и на почтовых серверах провайдеров. Антиспамовые фильтры анализируют содержание письма или пытаются опознать спамера по электронному адресу. Если письмо классифицировано как спам, оно может быть помечено, перемещено в другую папку или даже удалено.

Для затруднения автоматической фильтрации спамовые сообщения часто искажаются, вместо букв используются похожие по начертанию цифры, русские буквы заменяются на латинские, а в случайных местах добавляются пробелы.

Практическое задание «Защита от спама». В почтовых программах (например, Outlook Express и SeaMonkey) создать антиспамовый фильтр (правила для работы с почтой), который будет защищать от получения рекламных сообщений:

- от конкретных отправителей;
- при наличии определенных слов в сообщении.



Защита от спама

Создадим антиспамовый фильтр в почтовой программе Outlook Express.

1. Запустить почтовую программу Outlook Express. Ввести команду [*Сервис-Правила для сообщений-Почта...*].
2. В появившемся диалоговом окне *Правила для сообщений* выбрать вкладку *Правила для почты* и щелкнуть по кнопке *Создать*.
3. Появившееся диалоговое окно *Создать правило для почты* (рис. 1.47) содержит три списка и текстовое поле. В первом списке необходимо выбрать условия применения правила, в данном случае *Искать сообщения, содержащие адресатов в поле «От»* и *Искать сообщения, содержащие заданные слова*.

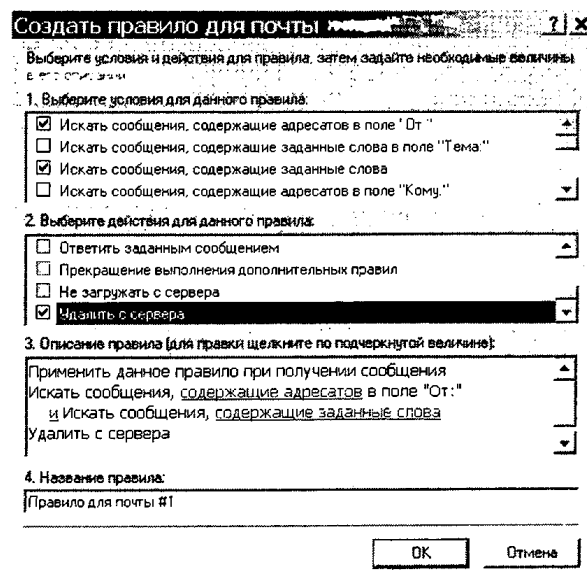


Рис. 1.47. Создание антиспамового фильтра в почтовой программе Outlook Express

4. Во втором списке необходимо выбрать действие над сообщениями для данного правила, в данном случае *Удалить с сервера*.
5. В третьем списке необходимо уточнить порядок действия данного правила.
Щелчком по ссылке *содержащие адресатов* вызвать диалоговое окно *Выбор получателей* и ввести почтовые адреса спамеров, от которых почтовые сообщения не будут получаться.
Щелчком по ссылке *содержащие заданные слова* вызвать диалоговое окно *Ввод ключевых слов* и ввести ключевые слова, при наличии которых в почтовых сообщениях эти сообщения будут получаться.
6. Текстовое поле содержит название и номер созданного правила. После щелчка по кнопке *ОК* правило вступает в действие.

Создать антиспамовый фильтр в почтовой программе SeaMonkey.

1. Запустить почтовую программу SeaMonkey. Ввести команду [*Инструменты-Фильтры сообщений...*].
2. В появившемся диалоговом окне *Правила фильтрации* (рис. 1.48) в текстовое поле ввести имя фильтра (например, *Антиспамовый фильтр*).

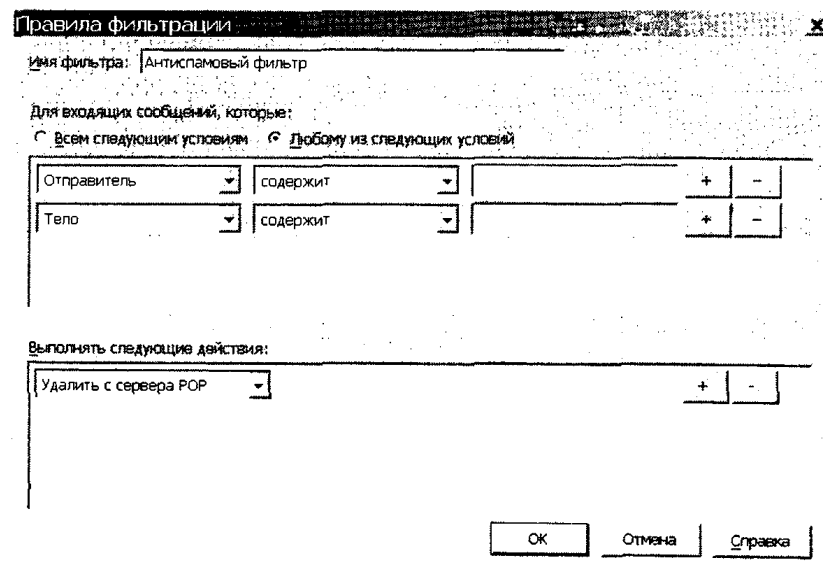


Рис. 1.48. Создание антиспамового фильтра в почтовой программе SeaMonkey


3. Во втором поле добавить условия применения фильтра щелчком по кнопке +.
В первом условии ввести почтовые адреса спамеров, от которых почтовые сообщения не будут получаться.
Во втором условии ввести ключевые слова, при наличии которых в почтовых сообщениях эти сообщения не будут получаться.
4. В третьем поле необходимо выбрать действие над сообщениями для данного фильтра, в данном случае *Удалить с сервера POP*.

Контрольные вопросы

1. Каковы вредные последствия спама?
2. Какие существуют основные типы спамовых сообщений?

Компьютерный практикум



Windows-CD 

- 1.24. В почтовых программах (например, Outlook Express и SeaMonkey) выполнить практическое задание «Защита от спама».



1.6.7. Хакерские утилиты и защита от них

В тех случаях, когда затруднения в работе или утере данных возникает в результате направленных действий, говорят о сетевых атаках.

Сетевые атаки. Сетевые атаки на удаленные серверы реализуются с помощью специальных программ, которые посылают на них специфические запросы. Это приводит к отказу в обслуживании («зависанию» сервера), если ресурсы атакуемого сервера недостаточны для обработки всех поступающих запросов.

DoS-программы (от англ. Denial of Service — отказ в обслуживании) реализуют атаку с одного компьютера с ведома пользователя. DoS-программы обычно наносят ущерб удаленным компьютерам и сетям, не нарушая работоспособность зараженного компьютера.

DDoS-программы (от англ. Distributed DoS — распределенный DoS) реализуют распределенные атаки с разных компьютеров, причем без ведома пользователей зараженных компьютеров. Для этого DDoS-программа засылается на

компьютеры «жертв-посредников» и после запуска в зависимости от текущей даты или по команде от хакера начинает сетевую атаку на указанный сервер в сети.



Некоторые сетевые черви содержат в себе DoS-процедуры, атакующие сайты, которые по каким-либо причинам «невзлюбил» автор червя. Так, червь «Codedred» 20 августа 2001 года организовал успешную атаку на официальный сайт президента США, а червь «Mydoom» 1 февраля 2004 года «выключил» сайт компании — производителя дистрибутивов UNIX.

Некоторые хакерские утилиты реализуют фатальные сетевые атаки. Такие утилиты используют уязвимости в операционных системах и приложениях и отправляют специально оформленные запросы на атакуемые компьютеры в сети. В результате сетевой запрос специального вида вызывает критическую ошибку в атакуемом приложении, и система прекращает работу.

Утилиты «взлома» удаленных компьютеров. Утилиты «взлома» удаленных компьютеров предназначены для проникновения в удаленные компьютеры с целью дальнейшего управления ими (используя методы троянских программ типа утилит удаленного администрирования) или для внедрения во «взломанную» систему других вредоносных программ.

Утилиты «взлома» удаленных компьютеров обычно используют уязвимости в операционных системах или приложениях, установленных на атакуемом компьютере. Профилактическая защита от таких хакерских утилит состоит в своевременной загрузке из Интернета обновлений системы безопасности операционной системы и приложений.

Руткиты. Руткит (от англ. root kit — «набор для получения прав root») — программа или набор программ для скрытого взятия под контроль «взломанной» системы.

Термин «rootkit» исторически пришел из мира операционной системы UNIX, и под этим термином понимается набор утилит, которые хакер устанавливает на «взломанном» им компьютере после получения первоначального доступа.

В операционной системе Windows под rootkit принято подразумевать программу, которая внедряется в систему и перехватывает системные функции. Перехват и модификация низкоуровневых функций, в первую очередь, позволяет такой программе достаточно качественно маскировать свое присутствие в системе. Кроме того, как правило, rootkit может маскировать присутствие в системе любых описанных в его кон-

фигурации процессов, каталогов и файлов на диске, ключей в реестре. Многие rootkit устанавливают в систему свои драйверы и службы (они, естественно, также являются «невидимыми»).

Защита от хакерских атак и сетевых червей. Защита компьютерных сетей или отдельных компьютеров от несанкционированного доступа может осуществляться с помощью межсетевого экрана, или брандмауэра (от англ. firewall). Межсетевой экран может быть реализован как аппаратно, так и программно.

Межсетевой экран позволяет:

- блокировать хакерские DoS-атаки, не пропуская на защищаемый компьютер сетевые пакеты с определенных серверов (определенных IP-адресов или доменных имен);
- не допускать проникновение на защищаемый компьютер сетевых червей (почтовых, Web и др.);
- препятствовать троянским программам отправлять конфиденциальную информацию о пользователе и компьютере.

Практическое задание «Настройка межсетевого экрана». Установить и настроить межсетевой экран (например, Agnitum Outpost).



Настройка межсетевого экрана

Установим политику защиты сетевого экрана, которая определяет, какие сетевые соединения разрешены или запрещены:

- *Блокировать все* — запрет любых сетевых взаимодействий;
- *Режим блокировки* — запрет всех сетевых взаимодействий, кроме явно разрешенных;
- *Режим обучения* — пользователь может принять решение о возможности или запрете выхода в сеть для приложения. Межсетевой экран выдает диалоговое окно предупреждения о первом сетевом взаимодействии для данного приложения, которым можно настроить правила доступа данного приложения в сеть (эта политика задается по умолчанию);
- *Режим бездействия* — разрешение любых сетевых взаимодействий;
- *Режим разрешения* — разрешение всех сетевых взаимодействий, кроме явно запрещенных.

1. Запустить межсетевой экран Agnitum Outpost.
Ввести команду [*Параметры-Политики...*].

2. В появившемся диалоговом окне *Параметры* активизировать вкладку *Политики* (рис. 1.49) и выбрать политику защиты сетевого экрана. Щелкнуть по кнопке *ОК*.

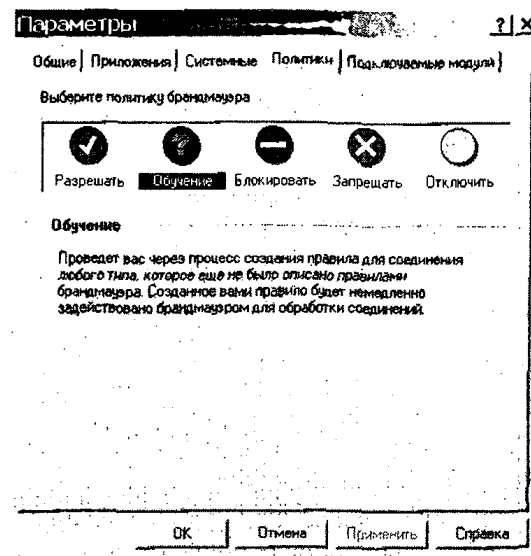


Рис. 1.49. Установка политики межсетевого экрана

При установленной политике *Режим обучения* каждое приложение будет запрашивать установление индивидуальной политики доступа к Интернету.

3. При попытке получения электронной почты появится диалоговое окно, запрашивающее разрешение данному приложению на доступ к Интернету (рис. 1.50).

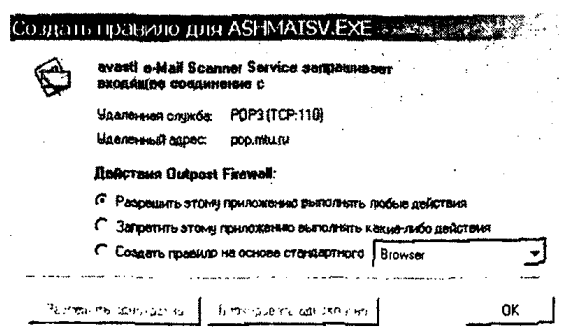


Рис. 1.50. Запрос на разрешение доступа к Интернету

Можно получить информацию о разрешенных и заблокированных приложениях.

4. Открыть окно межсетевое экрана (рис. 1.51) и в левой его части в иерархическом меню выбрать интересующий тип приложений. В правой части окна межсетевого экрана будет выведен список приложений с дополнительной информацией о типе соединения.

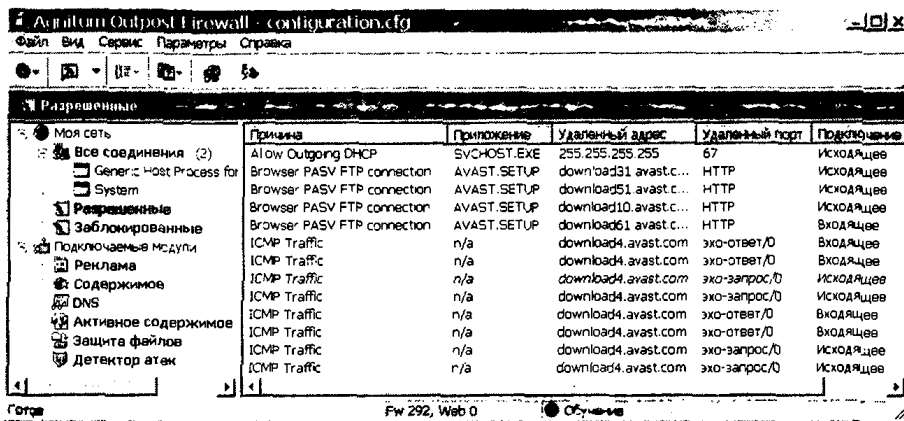


Рис. 1.51. Разрешенные приложения

С помощью межсетевого экрана можно установить уровень тревоги и заблокировать хакерские DoS-атаки.

5. Ввести команду [Параметры-Подключаемые модули...]. В появившемся диалоговом окне выбрать подключаемый модуль *Детектор атак* и щелкнуть по кнопке *Параметры...* В появившемся диалоговом окне *Параметры* (рис. 1.52) с

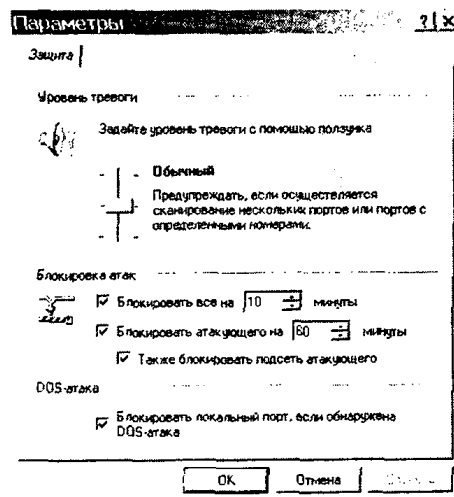


Рис. 1.52. Установка параметров защиты от DoS-атак

помощью вертикального ползунка выбрать уровень тревоги. Установить флажки в разделе *Блокировка атак* и с помощью счетчиков установить время блокировки.

- Открыть окно межсетевое экрана и в левой его части в иерархическом меню выбрать пункт *Детектор атак* (рис 1.53).

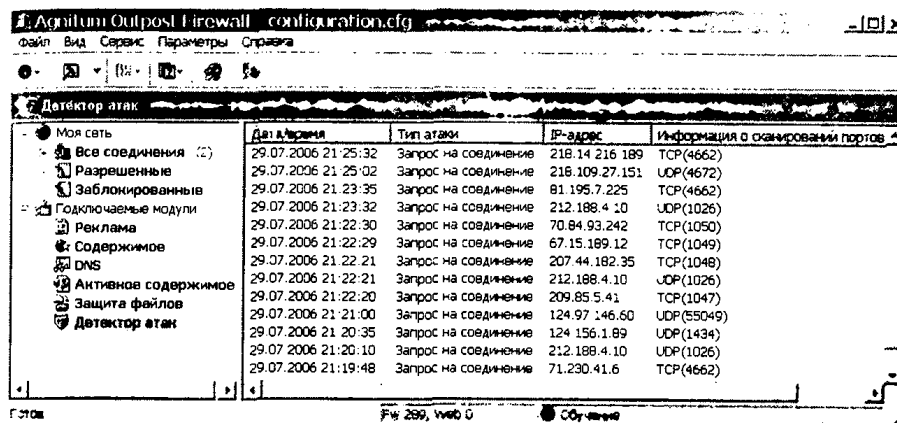



Рис. 1.53. Детектор DoS-атак

В правой части окна межсетевое экрана будет выведен список атак с дополнительной информацией о каждой атаке.

Контрольные вопросы

- Каковы основные хакерские утилиты и их злонамеренное использование?
- В каких целях используются межсетевые экраны?

Компьютерный практикум


Windows-CD 

- С помощью межсетевое экрана (например, Agnitum Outpost) выполнить практическое задание «Настройка межсетевое экрана».



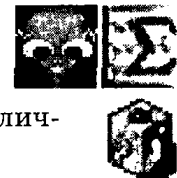
Глава 2

Информация. Системы счисления

Windows-CD 

В процессе изучения данной темы рекомендуется установить программное обеспечение в операционной системе Windows:

- компьютерные калькуляторы *Wise Calculator* и *NumLock Calculator*;
- программа перевода единиц измерения различных величин, в том числе информации.



2.1. Понятие «информация» в науках о неживой и живой природе, обществе и технике

Слово «информация» происходит от латинского слова «informatio», что в переводе означает сведение, разъяснение, ознакомление. Понятие «информация» является базовым в курсе информатики, невозможно дать его определение через другие, более «простые» понятия. В геометрии, например, невозможно выразить содержание базовых понятий «точка», «луч», «плоскость» через более простые понятия. Содержание основных, базовых понятий в любой науке должно быть пояснено на примерах или выявлено путем их сопоставления с содержанием других понятий.

В случае с понятием «информация» проблема его определения еще более сложная, так как оно является общенаучным понятием. Понятие «информация» используется в различных науках (информатике, кибернетике, биологии, физике и др.), при этом в каждой науке понятие «информация» связано с различными системами понятий.

2.1.1. Информация в физике

В физике информация рассматривается как антиэнтропия, или энтропия с обратным знаком. Энтропия системы яв-

Информация в открытых системах. Современная физика рассматривает открытые системы, которые обмениваются энергией или веществом с окружающей средой и увеличивают свою организованность. По мере увеличения организованности системы величина энтропии уменьшается и величина информации увеличивается.

В открытых системах в результате самоорганизации могут возникнуть пространственные или временные структуры, хранящие информацию. Примерами могут служить лазер, создающий когерентное излучение, жидкости, образующие пространственные или временные структуры при нагревании, или химические реакции, в которых наблюдаются периодические пространственные спирали или концентрические волны. Процессы самоорганизации в открытых системах изучаются синергетикой.

Информация в микро- и мегамире. В конце XIX века в классической физике рассматривали нашу Вселенную как замкнутую систему и предсказывали, что ее ждет «тепловая смерть», когда молекулы и атомы равномерно распределятся в пространстве и какие-либо изменения и развитие прекратятся.

Однако современная наука установила, что некоторые законы классической физики, справедливые для макротел, нельзя применять для микро- и мегамира. Согласно современным научным представлениям, наша Вселенная является динамически развивающейся системой, в которой постоянно происходят процессы усложнения структуры.

Астрономия-11

Согласно теории Большого взрыва, Вселенная образовалась около 15 миллиардов лет назад в результате взрыва «первоматерии» (рис. 2.2). В первые мгновения материя существовала фактически в форме энергии, а затем на протяжении долей секунды начало образовываться вещество в форме элементарных частиц (электронов, протонов, нейтронов и др.).

В следующий миллион лет основные события развивались в микромире. Из разлетающихся во все стороны элементарных частиц образовывались атомы, то есть из хаоса возникали системы с более сложной структурой. Сначала образовывались атомы самых легких химических элементов (водорода и гелия), а затем и более тяжелых элементов (рис. 2.3).

В мегамире в течение последующих миллиардов лет под действием сил гравитационного притяжения из хаоса гигант-

ских облаков пыли и газа формировались сложные структуры — галактики (рис. 2.4).

Краткая история вселенной

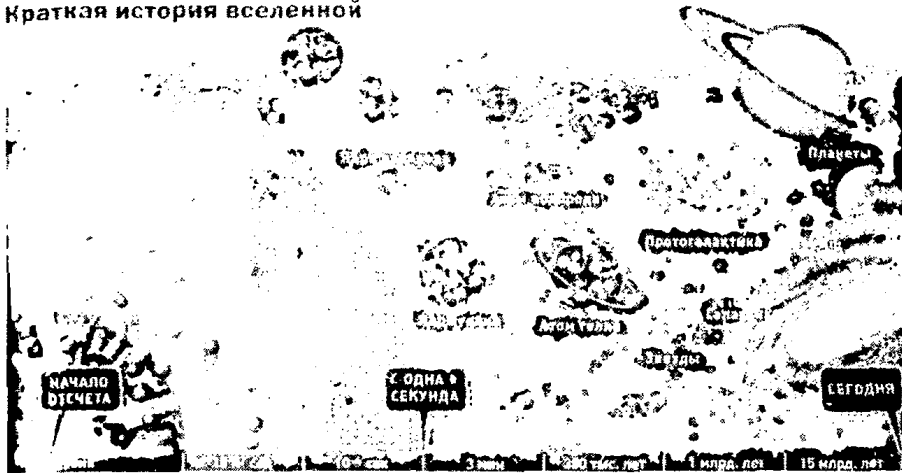


Рис. 2.2. Развитие Вселенной



Рис. 2.3. Переход от хаоса к порядку в микромире

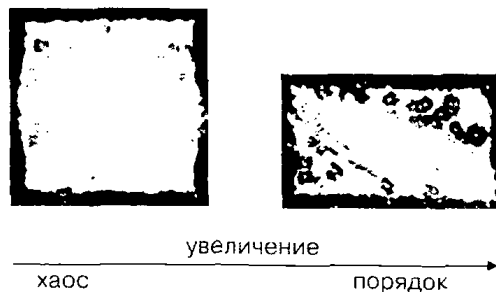


Рис. 2.4. Переход от хаоса к порядку в макромире

Наша солнечная система, в которую входит планета Земля, образовалась около 5 миллиардов лет назад и вместе с сотнями миллионов других звезд образует нашу галактику Млечный Путь.

На поверхности планет стали происходить химические реакции, в результате которых из атомов образовывались более сложные системы — молекулы веществ. В том числе молекула воды, которая состоит из двух атомов водорода и одного атома кислорода.

Химия 9

Таким образом, с одной стороны, в неживой природе в замкнутых системах идут процессы в направлении от порядка к хаосу (в них информация уменьшается). С другой стороны, в открытых системах может происходить усложнение структуры и, следовательно, информация, являющаяся мерой упорядоченности элементов системы, возрастает.



Мы живем в **макромире**, т. е. в мире, который состоит из объектов, по своим размерам сравнимых с человеком (рис. 2.5). Обычно макрообъекты разделяют на неживые (камень, льдина и т. д.), живые (растения, животные, сам человек) и искусственные (здания, средства транспорта, станки и механизмы, компьютеры и т. д.).

Макрообъекты состоят из молекул и атомов, которые, в свою очередь, состоят из элементарных частиц, размеры которых чрезвычайно малы. Этот мир называется **микромиром** (рис. 2.6).



Рис. 2.5. Макромир. Гулливер в стране лилипутов

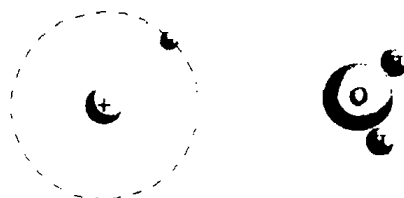


Рис. 2.6. Микромир. Атом водорода и молекула воды

Мы живем на планете Земля, которая входит в солнечную систему, Солнце вместе с сотнями миллионов других звезд образует нашу галактику Млечный Путь, а миллиарды галактик образуют Вселенную. Все эти объекты имеют громадные размеры и образуют **мегамир** (рис. 2.7).



Рис. 2.7. Мегамир. Солнечная система

Контрольные вопросы

1. Приведите примеры перехода от хаоса к порядку (увеличения информации) в окружающем мире.
2. Приведите примеры перехода от порядка к хаосу (уменьшения информации) в окружающем мире.

2.1.2. Информация в биологии

Процесс увеличения информации характерен для открытых, обменивающихся веществом и энергией с окружающей средой, саморазвивающихся систем живой природы (белковых молекул, организмов, популяций животных и т. д.). В процессе эволюции живой природы идет саморазвитие, т. е. повышение сложности и разнообразия живых организмов, а следовательно, увеличение информации, понимаемой как мера упорядоченности элементов системы.

Информационные сигналы. В биологии, которая изучает живую природу, понятие «информация» связывается с целесообразным поведением живых организмов. Такое поведение строится на основе получения и использования организмом информации об окружающей среде в форме информационных сигналов. Информационные сигналы могут иметь различную физическую или химическую природу: звук, свет, запах и другие.

Простейшие (например, амеба) могут получать информацию лишь о химическом составе и температуре окружающей среды. Причем информация может быть получена только о ближайших областях окружающей среды путем непосредственного контакта простейшего со средой.

Человек воспринимает информацию об окружающем мире с помощью органов чувств (зрения, слуха, обоняния, осязания и вкуса), причем может это делать на большом расстоянии. Чувствительные нервные окончания органов чувств (рецепторы) воспринимают воздействие (например, на глазном дне колбочки и палочки реагируют на воздействие световых лучей) и передают его нейронам (нервным клеткам), цепи которых составляют нервную систему.

Нейрон может находиться в двух состояниях: невозбужденном или возбужденном. Возбужденный нейрон генерирует электрический импульс, который передается по нервной системе. В нервной системе происходит кодирование и передача информации с помощью двух состояний нейрона: нет импульса, есть импульс.

В этом случае сами состояния нейрона можно рассматривать как знаки некоторого алфавита нервной системы, с помощью которого происходит передача информации.

Генетическая информация. Понятие «информация» в биологии используется также в связи с исследованиями механизмов наследственности. Генетическая информация передается по наследству и хранится во всех клетках живых организмов. Гены представляют собой сложные молекулярные структуры, содержащие информацию о строении живых организмов. Последнее обстоятельство позволило проводить научные эксперименты по клонированию, т. е. созданию точных копий организмов из одной клетки.

Общая биология 10–11

Генетическая информация определяет строение и развитие живых организмов и передается по наследству.

Хранится генетическая информация в клетках организмов в структуре молекул ДНК (дезоксирибонуклеиновой кислоты). Молекула ДНК (рис. 2.8) состоит из двух скрученных друг с другом в спираль цепей, построенных из четырех нуклеотидов: А, G, Т и С, которые образуют генетический код.

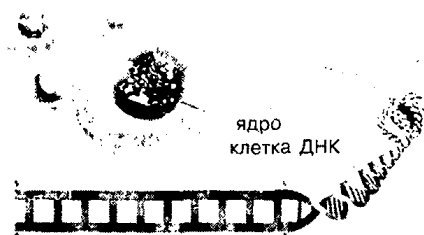


Рис. 2.8. Молекула ДНК

Молекула ДНК человека включает в себя около 3 миллиардов пар нуклеотидов. В ней закодирована вся информация об организме человека: его внешность, предрасположенность к болезням, способности и т. д.



В живых организмах информация передается и хранится с помощью объектов различной физической природы (состояние нейрона, нуклеотиды в молекуле ДНК), которые могут рассматриваться как знаки биологических алфавитов.

Контрольные вопросы

1. Какова физическая природа знака при представлении информации в нервной системе? В генетическом коде?

2.1.3. Информация в общественных науках

Социально-значимые свойства информации. Человек — существо социальное, для общения с другими людьми он должен обмениваться с ними информацией, причем обмен информацией всегда производится на определенном языке: русском, английском и т. д. Участники дискуссии должны владеть тем языком, на котором ведется общение, тогда информация будет понятной всем участникам обмена информацией.

Информация должна быть **полезной**, тогда дискуссия приобретает практическую ценность. Бесплезная информация создает информационный шум, который затрудняет восприятие полезной информации. Примерами передачи и получения бесполезной информации могут служить некоторые конференции и чаты в Интернете.

Широко известен термин «средства массовой информации» (газеты, радио, телевидение), которые доводят информацию до каждого члена общества. Такая информация должна быть **достоверной** и **актуальной**. Недостоверная информация вводит членов общества в заблуждение и может быть причиной возникновения социальных потрясений. Неактуальная информация бесполезна и поэтому никто, кроме историков, не читает прошлогодних газет.

Для того чтобы человек мог правильно ориентироваться в окружающем мире, информация должна быть **полной** и **точной**. Задача получения полной и точной информации стоит перед наукой. Владение научными знаниями в процессе обучения позволяет человеку получать полную и точную информацию о природе, обществе и технике.

Информация и знания. Человек получает информацию из окружающего мира с помощью органов чувств, анализирует ее и выявляет существенные закономерности с помощью мышления, хранит полученную информацию в памяти. Процесс систематического научного познания окружающего мира приводит к накоплению информации в форме знаний (фактов, научных теорий и т. д.). Таким образом, с точки зрения процесса познания информация может рассматриваться как **знания**.

Процесс познания можно наглядно изобразить в виде расширяющегося круга знания (такой способ придумали еще древние греки). Вне этого круга лежит область незнания, а окружность является границей между знанием и незнанием (рис. 2.9). Парадокс состоит в том, что чем большим объемом знаний обладает человек, чем шире круг знаний, тем больше



Рис. 2.9. Знание и незнание

он ощущает недостаток знаний, и тем больше граница нашего незнания, мерой которого в этой модели является длина окружности.

Так, объем знаний выпускника школы гораздо больше, чем объем знаний первоклассника, однако и граница его незнания существенно больше. Действительно, первоклассник ничего не знает о законах физики и его это не очень волнует, тогда как выпускник школы при подготовке к экзаменам по физике может обнаружить, что существуют физические законы, которые он не знает или не понимает.

Информацию, которую получает человек, можно считать мерой уменьшения неопределенности знания. Если некоторое сообщение приводит к уменьшению неопределенности нашего знания, то можно говорить, что такое сообщение содержит информацию.

Например, после сдачи экзамена по информатике вы мучаетесь неопределенностью, вы не знаете, какую оценку получили. Наконец, экзаменационная комиссия объявляет результаты экзамена и вы получаете сообщение, которое приносит полную определенность, теперь вы знаете свою оценку. Происходит переход от незнания к полному знанию, значит сообщение экзаменационной комиссии содержит информацию.

Контрольные вопросы

1. Какие социально-значимые свойства информации можно выделить?
2. Как можно представить процесс познания?

2.1.4. Информация в кибернетике

В кибернетике (науке об управлении) понятие «информация» используется для описания процессов управления в сложных системах (живых организмах или технических устройствах). Жизнедеятельность любого организма или нормальное функционирование технического устройства связано с процессами управления, благодаря которым поддерживаются в необходимых пределах значения его параметров. Так для поддержания своей жизнедеятельности любой живой организм постоянно получает информацию из внешнего мира с помощью органов чувств, обрабатывает ее и управляет своим поведением (например, перемещаясь в пространстве, избегает опасности).



Процессы управления включают в себя получение, хранение, преобразование и передачу информации.

В процессе управления полетом самолета в режиме автопилота бортовой компьютер получает информацию от датчиков (скорости, высоты и т. д.), обрабатывает ее и передает команды на исполнительные механизмы, изменяющие режим полета (закрылки, клапаны, регулирующие работу двигателей, и т. д.).

В любом процессе управления всегда происходит взаимодействие двух объектов — управляющего и управляемого, которые соединены каналами прямой и обратной связи. По каналу прямой связи передаются управляющие сигналы, а по каналу обратной связи — информация о состоянии управляемого объекта.

Разомкнутые системы управления. Если в процессе управления не учитывается состояние управляемого объекта и обеспечивается управление только по прямому каналу (от управляющего объекта к управляемому), то такие системы управления называются разомкнутыми. Информационную модель разомкнутой системы управления можно наглядно представить с помощью следующей схемы (рис. 2.10).

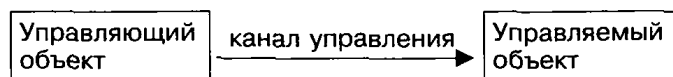


Рис. 2.10. Разомкнутая система управления

В качестве примера разомкнутой системы управления рассмотрим процесс записи информации на гибкий диск, в котором управляющий объект *Контроллер дисковод* изменяет состояние управляемого объекта *Магнитной головки*.

Для того чтобы информация могла быть записана, необходимо установить магнитную головку дисковода над определенной концентрической дорожкой диска. При записи информации на гибкие диски не требуется особой точности установки (имеется всего 80 дорожек) и можно не учитывать возможные (например, от нагревания) механические деформации носителя, поэтому управляющий объект (*Контроллер дисковода*) просто перемещает управляемый объект (*Магнитную головку*) на определенное расстояние вдоль радиуса дискеты.

Замкнутые системы управления. В замкнутых системах управления управляющий объект по прямому каналу управления производит необходимые действия над объектом управления, а по каналу обратной связи получает информацию о его реальных параметрах (рис. 2.11). Это позволяет осуществлять управление с гораздо большей точностью.

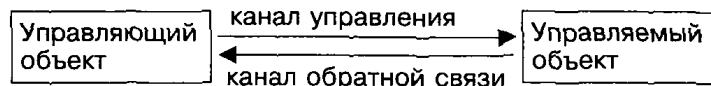


Рис. 2.11. Замкнутая система управления

Информационную модель замкнутой системы управления можно наглядно представить с помощью следующей схемы:

Примером использования замкнутой системы управления являются жесткие диски. При записи информации на жесткие диски требуется особая точность установки магнитных головок, так как на рабочей поверхности носителя имеются десятки тысяч дорожек и необходимо учитывать механические деформации магнитного носителя (например, в результате изменения температуры). Управляющий объект (*Контроллер жесткого диска*) постоянно получает информацию о реальном положении управляемого объекта (*Магнитных головок*) по каналу обратной связи, а по прямому каналу выставляет головки над поверхностью носителя с большой точностью.

Контрольные вопросы

1. В чем состоит различие между разомкнутыми и замкнутыми системами управления? Приведите примеры.

2.2. Количество информации как мера уменьшения неопределенности знания

1.3.1. Количество информации как мера уменьшения неопределенности знания

Информатика-8



Уменьшение неопределенности знания. Получение информации можно связать с уменьшением неопределенности

знания. Это позволяет количественно измерять информацию, что чрезвычайно важно для информатики. Рассмотрим вопрос об определении количества информации более подробно на конкретных примерах.

Пусть у нас имеется монета, которую мы бросаем на ровную поверхность. С равной вероятностью произойдет одно из двух возможных событий — монета окажется в одном из двух положений: «орел» или «решка».



Можно говорить, что события равновероятны, если при возрастающем числе опытов количества выпадений «орла» и «решки» постепенно сближаются. Например, если мы бросим монету 10 раз, то «орел» может выпасть 7 раз, а решка — 3 раза, если бросим монету 100 раз, то «орел» может выпасть 60 раз, а «решка» — 40 раз, если бросим монету 1000 раз, то «орел» выпадет 520 раз, а «решка» — 480 и т. д. В итоге при очень большой серии опытов количества выпадений «орла» и «решки» практически сравниваются.

Перед броском существует неопределенность нашего знания (возможны два события), и как упадет монета, предсказать невозможно. После броска наступает полная определенность, так как мы видим (получаем зрительное сообщение), что монета в данный момент находится в определенном положении (например, «орел»). Это сообщение приводит к уменьшению неопределенности нашего знания в два раза, так как из двух возможных равновероятных событий реализовалось одно (рис. 2.12).

Возможные события	Произошедшее событие
	
	

Рис. 2.12. Возможные и произошедшее события

В окружающей действительности достаточно часто встречаются ситуации, когда может произойти некоторое количество

равновероятных событий. Так, при бросании равносторонней четырехгранной пирамиды существуют 4 равновероятных события, а при бросании шестигранного игрального кубика — 6 равновероятных событий.

Чем больше количество возможных событий, тем больше начальная неопределенность нашего знания и соответственно тем большее количество информации будет содержать сообщение о результатах опыта.

Единицы измерения количества информации. Для количественного выражения любой величины необходимо определить единицу измерения. Так, для измерения длины в качестве единицы выбран метр, для измерения массы — килограмм и т. д. Аналогично, для определения количества информации необходимо ввести единицу измерения.

1.3.2. Определение количества информации

Информатика и ИКТ-8



За единицу количества информации принимается такое количество информации, которое содержит сообщение, уменьшающее неопределенность знания в два раза. Такая единица названа бит.

Если вернуться к опыту с бросанием монеты, то здесь неопределенность как раз уменьшается в два раза (из двух возможных событий реализуется одно) и, следовательно, количество полученной информации равно 1 биту.

Минимальной единицей измерения количества информации является бит, а следующей по величине единицей — байт, причем

$$1 \text{ байт} = 2^3 \text{ битов} = 8 \text{ битов.}$$

В информатике система образования кратных единиц измерения количества информации несколько отличается от принятых в большинстве наук. Традиционные метрические системы единиц, например Международная система единиц СИ, в качестве множителей кратных единиц использует коэффициент 10^n , где $n = 3, 6, 9$ и т. д., что соответствует десятичным приставкам «кило» (10^3), «мега» (10^6), «гига» (10^9) и т. д.

Компьютер оперирует числами не в десятичной, а в двоичной системе счисления, поэтому в кратных единицах измерения количества информации используется коэффициент 2^n .

Так, кратные байту единицы измерения количества информации вводятся следующим образом:

- 1 Кбайт = 2^{10} байт = 1024 байт;
- 1 Мбайт = 2^{10} Кбайт = 1024 Кбайт;
- 1 Гбайт = 2^{10} Мбайт = 1024 Мбайт.

Практическое задание «Перевод единиц измерения количества информации». Произвести перевод количества информации с помощью специализированной программы перевода единиц измерения (например, VersaVerter).



Перевод единиц измерения количества информации

1. Запустить программу VersaVerter (Перевод единиц измерения).
2. В открывшемся диалоговом окне программы выбрать вкладку *Data*.
3. В левом окне диалогового окна выбрать исходную единицу измерения (например, *gigabyte*) и в текстовом поле ввести числовое значение (например, 1).

После выбора в правом окне конечной единицы измерения (например, *byte*) в текстовом поле появится числовое значение (в данном случае, 1 073 741 824) (рис. 2.13).

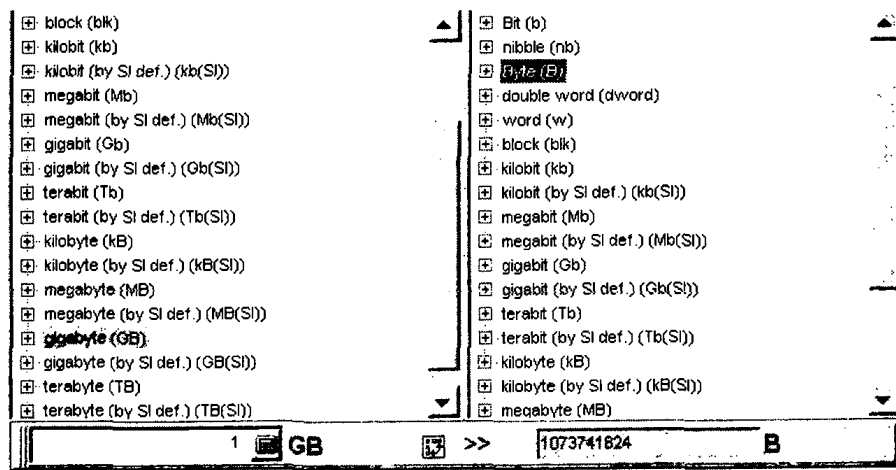


Рис. 2.13. Перевод единиц измерения количества информации

Количество возможных событий и количество информации. Существует формула, которая связывает между собой количество возможных событий N и количество информации I :

$$N = 2^I. \quad (2.1)$$

По этой формуле можно легко определить количество возможных событий, если известно количество информации. Например, если мы получили 4 бита информации, то количество возможных событий составляло

$$N = 2^4 = 16.$$

Наоборот, для определения количества информации, если известно количество событий, необходимо решить показательное уравнение относительно I .

Задание «Определение количества информации». Определить количество информации, полученной вторым игроком после первого хода первого игрока, в игре «крестики-нолики» на поле размером 8×8 клеток.

Перед первым ходом существует 64 возможных события (64 различных вариантов расположения «крестика»), тогда уравнение (2.1) принимает вид:

$$64 = 2^I.$$

Так как $64 = 2^6$, то

$$2^6 = 2^I.$$

Таким образом, $I = 6$ битов, т. е. количество информации, полученной вторым игроком после первого хода первого игрока, составляет 6 битов.

Контрольные вопросы


1. Приведите примеры уменьшения неопределенности знания после получения информации о произошедшем событии.
2. Как зависит количество информации от количества возможных событий?

Задания

- 2.1. В рулетке общее количество лунок равно 128. Какое количество информации мы получаем в зрительном сообщении об остановке шарика в одной из лунок?

Компьютерный практикум



Windows-CD 

- 2.1. С помощью программы перевода единиц измерения выполнить практическое задание «Перевод единиц измерения количества информации».



2.2. Заполнить пропуски числами и проверить правильность вычислений с помощью программы перевода единиц измерения:



- а) 5 Кбайт = ___ байт = ___ бит;
- б) ___ Кбайт = ___ байт = 12 288 бит;
- в) ___ Кбайт = ___ байт = 2^{13} бит;
- г) ___ Гбайт = 1536 Мбайт = ___ Кбайт;
- д) 512 Кбайт = 2___ байт = 2___ бит.

2.3. Алфавитный подход к определению количества информации

Определение количества информации на основе уменьшения неопределенности нашего знания рассматривает информацию с точки зрения содержания, ее понятности и новизны для человека. С этой точки зрения в опыте по бросанию монеты одинаковое количество информации содержится и в зрительном образе упавшей монеты, и в коротком сообщении «Орел», и в длинной фразе «Монета упала на поверхность земли той стороной вверх, на которой изображен орел».

Однако при хранении и передаче информации с помощью технических устройств целесообразно отвлечься от содержания информации и рассматривать ее как последовательность символов — знаков (букв, цифр, кодов цветов точек изображения и т. д.).



Набор символов знаковой системы (алфавит) можно рассматривать как различные возможные состояния (события).

Тогда, если считать, что появление символов в сообщении равновероятно, по формуле (2.1) можно рассчитать, какое количество информации несет каждый символ. Информационная емкость знаков зависит от их количества в алфавите (мощности алфавита): чем больше их количество, тем большее количество информации несет один знак.

Так, информационная емкость буквы в русском алфавите, если не использовать букву «ё», составляет:

$$32 = 2^I, \text{ т. е. } I = 5 \text{ битов.}$$

На основании алфавитного подхода можно подсчитать количество информации в сообщении I_c , для этого необходимо умножить количество информации, которое несет один символ I , на количество символов K в сообщении:

$$I_c = I \times K. \quad (2.2)$$

Задание «Определение количества информации в тексте». Система оптического распознавания символов позволяет преобразовывать отсканированные изображения страниц документа в текстовый формат со скоростью 4 страницы в минуту и использует алфавит мощностью 65 536 символов. Какое количество информации будет нести текстовый документ, каждая страница которого содержит 40 строк по 50 символов, после 10 минут работы приложения?

По формуле (2.1) определим информационную емкость символа алфавита:

$$65\,536 = 2^I \Rightarrow 2^{16} = 2^I \Rightarrow I = 16 \text{ битов.}$$

По формуле (2.2) определим количество информации на странице:

$$16 \text{ битов} \times 40 \times 50 = 32\,000 \text{ битов} = 4000 \text{ байтов.}$$

Определим количество информации, которое будет нести текстовый документ:

$$4000 \text{ байтов} \times 4 \times 10 = 160\,000 \text{ байтов} \approx 156 \text{ Кбайт.}$$

З а д а н и я

- 2.2. Пользователь компьютера может вводить в минуту 200 знаков. Мощность используемого алфавита равна 256. Какое количество информации может ввести пользователь в компьютер за 5 минут?

2.4. Формула Шеннона

Существует множество ситуаций, когда возможные события имеют различные вероятности реализации. Например, если монета несимметрична (одна сторона тяжелее другой), то при ее бросании вероятности выпадения «орла» и «решки» будут различаться.

Формулу для вычисления количества информации для событий с различными вероятностями предложил К. Шеннон в 1948 году. В этом случае количество информации определяется по формуле:

$$I = - \sum_{i=1}^N p_i \log_2 p_i, \quad (2.3)$$

- где I — количество информации,
 N — количество возможных событий,
 p_i — вероятности отдельных событий.

Для частного, но широко распространенного и рассмотренного выше случая, когда события равновероятны ($p_i = 1/N$), величину количества информации I можно рассчитать по формуле:

$$I = -\sum_{i=1}^N \frac{1}{N} \log_2 \frac{1}{N} = \log_2 N. \quad (2.4)$$

Задание «Бросание пирамидки». Определить количество информации, которую мы получаем в результате бросания несимметричной и симметричной пирамидок.

При бросании несимметричной четырехгранной пирамидки вероятности отдельных событий равны:

$$p_1 = 1/2, p_2 = 1/4, p_3 = 1/8, p_4 = 1/8.$$

Количество информации, которую мы получим после бросания несимметричной пирамидки, можно рассчитать по формуле (2.3):

$$\begin{aligned} I &= -(1/2 \cdot \log_2 1/2 + 1/4 \cdot \log_2 1/4 + 1/8 \cdot \log_2 1/8 + 1/8 \cdot \log_2 1/8) \text{ битов} = \\ &= (1/2 \cdot \log_2 2 + 1/4 \cdot \log_2 4 + 1/8 \cdot \log_2 8 + 1/8 \cdot \log_2 8) \text{ битов} = \\ &= (1/2 + 2/4 + 3/8 + 3/8) \text{ битов} = 14/8 \text{ битов} = 1,75 \text{ бита}. \end{aligned}$$

При бросании симметричной четырехгранной пирамидки вероятности отдельных событий равны между собой:

$$p_1 = p_2 = p_3 = p_4 = 1/4.$$

Количество информации, которую мы получим после бросания симметричной пирамидки, можно рассчитать по формуле (2.4):

$$I = \log_2 4 = 2 \text{ бита}.$$

Таким образом, при бросании симметричной пирамидки, когда события равновероятны, мы получим большее количество информации (2 бита), чем при бросании несимметричной пирамидки, когда события неравновероятны (1,75 бита).



Количество информации, которую мы получаем, достигает максимального значения, если события равновероятны.



В теории информации доказано, что максимальное количество информации несет сообщение, в котором вероятности появления всех знаков одинаковы.

Количество информации, которое несет знак, зависит от вероятности его получения. Если получатель заранее точно знает, какой знак придет, то полученное количество информации будет равно 0. Наоборот, чем менее вероятно получение знака, тем больше его информационная емкость.

В русской письменной речи частота использования букв в тексте различна, так в среднем на 1000 знаков осмысленного текста приходится 200 букв «а» и в сто раз меньшее количество буквы «ф» (всего 2). Таким образом, с точки зрения теории информации, информационная емкость знаков русского алфавита различна (у буквы «а» она наименьшая, а у буквы «ф» — наибольшая).

Выбор правильной стратегии в игре «Угадай число». На получении максимального количества информации строится выбор правильной стратегии в игре «Угадай число», в которой первый участник загадывает целое число (например, 3) из заданного интервала (например, от 1 до 16), а второй должен «угадать» задуманное число. Если рассмотреть эту игру с информационной точки зрения, то начальная неопределенность знания для второго участника составляет 16 возможных событий (вариантов загаданных чисел).

При правильной стратегии интервал чисел всегда должен делиться пополам, тогда количество возможных событий (чисел) в каждом из полученных интервалов будет одинаково и их отгадывание равновероятно. В этом случае на каждом шаге ответ первого игрока («Да» или «Нет») будет нести максимальное количество информации (1 бит).

Как видно из табл. 2.1, угадывание числа 3 произошло за четыре шага, на каждом из которых неопределенность знания второго участника уменьшалась в два раза за счет получения сообщения от первого участника, содержащего 1 бит информации. Таким образом, количество информации, необходимой для отгадывания одного из 16 чисел, составило 4 бита.

Таблица 2.1. Информационная модель игры «Угадай число»

Вопрос второго участника	Ответ первого участника	Неопределенность знания (количество возможных событий)	Полученное количество информации
		16	
Число больше 8?	Нет	8	1 бит
Число больше 4?	Нет	4	1 бит
Число больше 2?	Да	2	1 бит
Это число 3?	Да	1	1 бит

Практическое задание «Определение количества информации». В непрозрачном мешочке хранятся 10 белых, 20 красных, 30 синих и 40 зеленых шариков. Какое количество информации будет содержать зрительное сообщение о цвете вынутого шарика?

Так как количество шариков различных цветов неодинаково, то вероятности зрительных сообщений о цвете вынутого из мешочка шарика также различаются и равны количеству шариков данного цвета, деленному на общее количество шариков:

$$p_b = 0,1; \quad p_k = 0,2; \quad p_s = 0,3; \quad p_c = 0,4.$$

События неравновоятны, поэтому для определения количества информации, содержащейся в сообщении о цвете шарика, воспользуемся формулой (2.3):

$$I = -(0,1 \cdot \log_2 0,1 + 0,2 \cdot \log_2 0,2 + 0,3 \cdot \log_2 0,3 + 0,4 \cdot \log_2 0,4) \text{ битов.}$$

Для вычисления этого выражения, содержащего логарифмы, воспользуемся компьютерным калькулятором.



Определение количества информации

1. Запустить компьютерный калькулятор Wise Calculator.
2. Ввести выражение и получить результат (рис. 2.14).
Таким образом, $I \approx 1,85$ бита.

$$-(0,1 \cdot \log(2,0,1) + 0,2 \cdot \log(2,0,2) + 0,3 \cdot \log(2,0,3) + 0,4 \cdot \log(2,0,4))$$
 Result: 1.846439344671015

Рис. 2.14. Вычисление количества информации

Контрольные вопросы

1. В каком случае количество информации, полученной о событии, достигает максимального значения?

Задания

- 2.3. Какое количество вопросов достаточно задать вашему собеседнику, чтобы наверняка определить:

- день недели, в котором он родился;
- месяц, в котором он родился;
- число, в которое он родился?

Компьютерный практикум



Windows-CD 

2.3. Выполнить практическое задание «Определение количества информации» для следующих данных:



- 25 белых, 25 красных, 25 синих и 25 зеленых шариков;
- 30 белых, 30 красных, 30 синих и 10 зеленых шариков.

2.5. Кодирование текстовой, графической и звуковой информации

Двоичное кодирование текстовой информации в компьютере

Для представления текстовой информации (прописные и строчные буквы русского и латинского алфавитов, цифры, знаки и математические символы) достаточно 256 различных знаков. По формуле (2.1) можно вычислить, какое количество информации необходимо, чтобы закодировать каждый знак:

$$N = 2^I \Rightarrow 256 = 2^I \Rightarrow 2^8 = 2^I \Rightarrow I = 8 \text{ битов.}$$

Для обработки текстовой информации на компьютере необходимо представить ее в двоичной знаковой системе. Каждому знаку необходимо поставить в соответствие уникальный 8-битовый двоичный код, значения которого находятся в интервале от 00000000 до 11111111 (в десятичном коде от 0 до 255).

Присвоение знаку конкретного двоичного кода — это вопрос соглашения, которое фиксируется в кодовой таблице. К сожалению, в настоящее время существуют пять различных кодовых таблиц для русских букв (*Windows*, *MS-DOS*, *КОИ-8*, *Mac*, *ISO*), поэтому тексты, созданные в одной кодировке, не будут правильно отображаться в другой.

В последние годы широкое распространение получил новый международный стандарт кодирования текстовых символов *Unicode*, который отводит на каждый символ 2 байта (16 битов). По формуле (2.1) можно определить количество символов, которые можно закодировать:

$$N = 2^I = 2^{16} = 65\,536.$$

Такого количества символов оказалось достаточно, чтобы закодировать не только русский и латинский алфавиты, цифры, знаки и математические символы, но и греческий, арабский, иврит и другие алфавиты.

Двоичное кодирование графической информации в компьютере

Графические изображения из аналоговой (непрерывной) формы в цифровую (дискретную) преобразуются путем **пространственной дискретизации**. Изображение разбивается на отдельные маленькие фрагменты (точки, или пиксели), причем каждый элемент может иметь свой цвет (красный, зеленый, синий и т. д.).



Пиксель — минимальный участок изображения, которому независимым образом можно задать цвет.

В результате пространственной дискретизации графическая информация представляется в виде растрового изображения, которое формируется из определенного количества строк, которые, в свою очередь, содержат определенное количество точек.

Важнейшей характеристикой качества растрового изображения является разрешающая способность.



Разрешающая способность растрового изображения определяется количеством точек по горизонтали и вертикали на единицу длины изображения.

При одних и тех же размерах экрана чем меньше размер точки, тем больше разрешающая способность (больше количество строк раstra и точек в строке), и, соответственно, выше качество изображения. Величина разрешающей способности обычно выражается в dpi (dot per inch — точек на дюйм), т. е. в количестве точек в полоске изображения длиной один дюйм (1 дюйм = 2,54 см).

В процессе дискретизации могут использоваться различные палитры цветов, т. е. наборы цветов, которые могут принимать точки изображения. Каждый цвет можно рассматривать как возможное состояние точки. Количество цветов N в палитре и количество информации I , необходимое для кодирования цвета каждой точки, связаны между собой и могут быть вычислены по формуле (2.1).



Количество информации, которое используется при кодировании цвета точек изображения, называется глубиной цвета.

Наиболее распространенными значениями глубины цвета при кодировании цветных изображений являются 4, 8, 16 или 24 бита на точку. Можно определить количество цветов в 24-битовой палитре:

$$N = 2^I = 2^{24} = 16\,777\,216.$$

Двоичное кодирование звуковой информации в компьютере

Звук представляет собой распространяющуюся в воздухе, воде или другой среде волну с непрерывно меняющейся интенсивностью и частотой.

Человек воспринимает звуковые волны (колебания воздуха) с помощью слуха в форме звуков различной громкости и тона, чем больше интенсивность звуковой волны, тем громче звук, чем больше частота волны, тем выше тон звука.

Для того чтобы компьютер мог обрабатывать реальный (записанный) звук, непрерывный звуковой сигнал должен быть преобразован в цифровую дискретную форму с помощью **временной дискретизации**. Непрерывная звуковая волна разбивается на отдельные маленькие временные участки, причем для каждого такого участка устанавливается определенная величина интенсивности звука.

Для записи аналогового звука и его преобразования в цифровую форму используется микрофон, подключенный к звуковой плате. Качество полученного цифрового звука зависит от количества измерений уровня громкости звука в единицу времени, т. е. **частоты дискретизации**. Чем большее количество измерений производится за 1 секунду (чем больше частота дискретизации), тем точнее «лесенка» цифрового звукового сигнала повторяет кривую аналогового сигнала.



Частота дискретизации звука — это количество измерений громкости звука за одну секунду.

Частота дискретизации звука может лежать в диапазоне от 8000 до 48 000 измерений громкости звука за одну секунду.

Каждому уровню дискретизации присваивается определенное значение уровня громкости звука. Уровни громкости

звука можно рассматривать как набор возможных состояний N , для кодирования которых необходимо определенное количество информации I , которое называется **глубиной кодирования звука**.



Глубина кодирования звука — это количество информации, которое необходимо для кодирования дискретных уровней громкости цифрового звука.

Если известна глубина кодирования, то количество уровней громкости цифрового звука можно рассчитать по формуле (2.1). Пусть глубина кодирования звука составляет 16 битов, тогда количество уровней громкости звука равно

$$N = 2^I = 2^{16} = 65\,536.$$

Чем больше частота дискретизации и глубина кодирования звука, тем более качественным будет звучание оцифрованного звука. Самое низкое качество оцифрованного звука, соответствующее качеству телефонной связи, будет при частоте дискретизации 8000 раз в секунду, глубине кодирования 8 битов и записи одной звуковой дорожки (режим моно). Высокое качество оцифрованного звука, соответствующее качеству аудио-CD, обеспечивается при частоте дискретизации 48 000 раз в секунду, глубине кодирования 16 битов и записи двух звуковых дорожек (режим стерео).

Контрольные вопросы

1. Чем различаются между собой кодировки русского алфавита?
2. Объясните, как разрешение и глубина цвета влияют на качество изображения.
3. Объясните, как частота дискретизации и глубина кодирования влияют на качество цифрового звука.



- 2.4. Представьте в форме десятичного кода слово «бит» в кодировках *Windows*, *MS-DOS*, *КОИ-8*, *Mac*, *ISO*. Воспользуйтесь *Windows-CD* для получения кодировочных таблиц.
- 2.5. Сканируется цветное изображение стандартного размера А4 (21×29,7 см). Разрешающая способность сканера 1200 dpi, глубина цвета 24 бита. Какой информационный объем будет иметь полученный графический файл?

- 2.6. Определить длительность звукового файла, который уместится на гибкой дискете 3,5" (для хранения данных на дискете выделяется 2847 секторов объемом 512 байтов каждый):
- при низком качестве звука: моно, 8 битов, 8 кГц;
 - при высоком качестве звука: стерео, 16 битов, 48 кГц.

2.6. Хранение информации

Информация, закодированная с помощью естественных и формальных языков, а также информация в форме зрительных и звуковых образов хранится в памяти человека. Однако для долговременного хранения информации, ее накопления и передачи из поколения в поколение используются **носители информации**.

Материальная природа носителей информации может быть различной: молекулы ДНК, которые хранят генетическую информацию; бумага, на которой хранятся тексты и изображения; магнитная лента, на которой хранится звуковая информация; фото- и киноплёнки, на которых хранится графическая информация; микросхемы памяти, магнитные и оптические диски, на которых хранятся программы и данные в компьютере, и т. д.

По оценкам специалистов, объем информации, фиксируемой на различных носителях, превышает один эксабайт в год (10^{18} байт/год). Примерно 80% всей этой информации хранится в цифровой форме на магнитных и оптических носителях и только 20% — на аналоговых носителях (бумага, магнитные ленты, фото- и киноплёнки). Если всю записанную в 2005 году информацию распределить на всех жителей планеты, то на каждого человека придется по 300 Мбайт.

Информационная емкость носителей информации. Носители информации характеризуются информационной емкостью, т. е. количеством информации, которое они могут хранить. Наиболее информационно емкими являются молекулы ДНК, которые имеют очень малый размер и плотно упакованы. Это позволяет хранить огромное количество информации (до 10^{21} бит в 1 см^3), что дает возможность организму развиваться из одной-единственной клетки, содержащей всю необходимую генетическую информацию.

Современные микросхемы памяти позволяют хранить в 1 см^3 до 10^{10} битов информации, однако это в 100 миллиардов раз меньше, чем в ДНК. Можно сказать, что современные технологии пока существенно проигрывают биологической эволюции.

Однако если сравнивать информационную емкость традиционных носителей информации (книг) и современных компьютерных носителей, то прогресс очевиден. На каждом гибком магнитном диске может храниться книга объемом около 600 страниц, а на жестком магнитном диске или DVD — целая библиотека, включающая десятки тысяч книг.

Надежность и долговременность хранения информации. Большое значение имеет надежность и долговременность хранения информации. Большую устойчивость к возможным повреждениям имеют молекулы ДНК, так как существует механизм обнаружения повреждений ее структуры (мутаций) и самовосстановления.

Надежность (устойчивость к повреждениям) достаточно высока у аналоговых носителей, повреждение которых приводит к потере информации только на поврежденном участке. Поврежденная часть фотографии не лишает возможности видеть оставшуюся часть, повреждение участка магнитной ленты приводит лишь к временному пропаданию звука и т. д.

Цифровые носители гораздо более чувствительны к повреждениям, даже потеря одного бита данных на магнитном или оптическом дисках может привести к невозможности считать файл, т. е. к потере большого объема данных. Именно поэтому необходимо соблюдать правила эксплуатации и хранения цифровых носителей информации.

Наиболее долговременным носителем информации является молекула ДНК, которая в течение десятков тысяч лет (человек) и миллионов лет (некоторые живые организмы) сохраняет генетическую информацию данного вида.

Аналоговые носители способны сохранять информацию в течение тысяч лет (египетские папирусы и шумерские глиняные таблички), сотен лет (бумага) и десятков лет (магнитная лента, фото- и киноплёнка).

Цифровые носители появились сравнительно недавно, и поэтому об их долговременности можно судить только по оценкам специалистов. По экспертным оценкам, при правильном хранении оптические носители способны хранить информацию сотни лет, а магнитные — десятки лет.

Контрольные вопросы

1. Какие достоинства и недостатки имеют аналоговые и цифровые носители информации?

2.7. Кодирование числовой информации. Системы счисления

«Всё есть число», — говорили древнегреческие философы, ученики Пифагора, подчеркивая необычайно важную роль чисел в практической деятельности. Известно множество способов представления чисел. Числа записываются с использованием особых знаковых систем, которые называются **системами счисления**. Алфавит систем счисления состоит из символов, которые называются цифрами.



Система счисления — это знаковая система, в которой числа записываются по определенным правилам с помощью символов некоторого алфавита, называемых цифрами.

Все системы счисления делятся на две большие группы: **непозиционные** и **позиционные**.

2.7.1. Непозиционные системы счисления

В непозиционных системах счисления количественный эквивалент каждой цифры не зависит от ее положения (места, позиции) в записи числа.

Единичная система счисления. Как только люди начали считать, у них появилась потребность в записи чисел. Находки археологов на стоянках первобытных людей свидетельствуют о том, что первоначально количество предметов отображали равным количеством каких-либо значков (бирок): зарубок, черточек, точек.

Позже, для облегчения счета, эти значки стали группировать по три или по пять. Такая система записи чисел называется **единичной (унарной)**, так как любое число в ней образуется путем повторения одного знака, символизирующего единицу. Отголоски единичной системы счисления встречаются и сегодня. Так, чтобы узнать, на каком курсе учится курсант военного училища, нужно сосчитать, какое количество полосок нашито на его рукаве.

Древнеегипетская непозиционная система счисления. Примерно в третьем тысячелетии до нашей эры древние египтяне придумали свою числовую систему, в которой для

обозначения ключевых чисел 1, 10, 100 и т. д. использовались специальные значки — иероглифы (рис. 2.15).

Все остальные числа составлялись из этих ключевых при помощи операции сложения. Система счисления Древнего Египта является десятичной, но непозиционной.

Например, чтобы изобразить число 3252, рисовали три цветка лотоса (три тысячи), два свернутых пальмовых листа (две сотни), пять дуг (пять десятков) и два шеста (две единицы). Величина числа не зависела от того, в каком порядке располагались составляющие его знаки: их можно было записывать сверху вниз, справа налево или в произвольном порядке.

Римская система счисления. Примером непозиционной системы, которая сохранилась до наших дней, может служить система счисления, которая применялась более двух с половиной тысяч лет назад в Древнем Риме. В основе римской системы счисления лежали знаки I (один палец) для числа 1, V (раскрытая ладонь) для числа 5, X (две сложенные ладони) для 10, а для обозначения чисел 100, 500 и 1000 стали применять первые буквы соответствующих латинских слов (Centum — сто, Demimille — половина тысячи, Mille — тысяча).

Чтобы записать число, римляне разлагали его на сумму тысяч, полутысяч, сотен, полусотен, десятков, пятков, единиц. Например, десятичное число 28 представляется следующим образом:

$$\text{XXVIII} = 10 + 10 + 5 + 1 + 1 + 1$$

(два десятка, пяток, три единицы).

Для записи чисел римляне использовали не только сложение, но и вычитание. При этом применялось следующее правило: каждый меньший знак, поставленный справа от большего, прибавляется к его значению, а каждый меньший знак, поставленный слева от большего, вычитается из него.

Например, IX — обозначает 9, XI — обозначает 11.

Десятичное число 99 имеет следующее представление:

$$\text{XCIX} = -10 + 100 - 1 + 10.$$

Практическое задание «Римская система счисления». Перевести числа из римской системы счисления в десятичную и из десятичной в римскую с использованием компьютерного калькулятора NumLock Calculator.

I	-	единицы
X	-	десятки
C	-	сотни
I	-	тысячи

Рис. 2.15. Алфавит древнеегипетской системы счисления



Римская система счисления

1. Запустить компьютерный калькулятор NumLock Calculator.
2. Установить минималистский вид калькулятора командой [*Вид калькулятора-Универсальный*].
3. Установить формат исходных данных командой [*Формат чисел в выражении-Римский*] и формат результата командой [*Формат результата-Десятичный*].
4. В окне ввода данных ввести римское число, например XXVIII, и нажать клавишу со знаком «=». Появится результат, число 28, записанное в десятичной системе счисления (рис. 2.16).

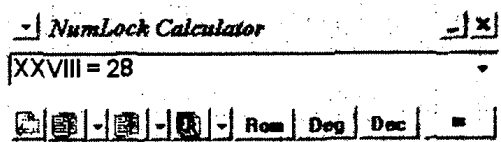


Рис. 2.16. Перевод из римской в десятичную систему счисления

5. Установить формат исходных данных командой [*Формат чисел в выражении-Десятичный*] и формат результата командой [*Формат результата-Римский*].
6. В окне ввода данных ввести десятичное число, например 99, и нажать клавишу со знаком «=». Появится результат, число XCIX, записанное в римской системе счисления (рис. 2.17).

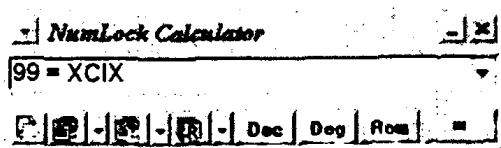


Рис. 2.17. Перевод из десятичной в римскую систему счисления

Римскими цифрами пользовались очень долго. Еще 200 лет назад в деловых бумагах числа должны были обозначаться римскими цифрами (считалось, что обычные арабские цифры легко подделать). Римская система счисления сегодня используется в основном для наименования знаменательных дат, томов, разделов и глав в книгах.

Алфавитные системы счисления. Более совершенными непозиционными системами счисления были алфавитные сис-

темы. К числу таких систем счисления относились греческая, славянская, финикийская и другие. В них числа от 1 до 9, десятки (от 10 до 90) и сотни (от 100 до 900) обозначались буквами алфавита.

В алфавитной системе счисления Древней Греции числа 1, 2, ..., 9 обозначались первыми девятью буквами греческого алфавита, например $\alpha = 1$, $\beta = 2$, $\gamma = 3$ и т. д. Для обозначения чисел 10, 20, ..., 90 применялись следующие 9 букв ($\iota = 10$, $\kappa = 20$, $\lambda = 30$, $\mu = 4$ и т. д.), а для обозначения чисел 100, 200, ..., 900 — последние 9 букв ($\rho = 100$, $\sigma = 200$, $\tau = 300$ и т. д.). Например, число 141 обозначалось $\rho\mu\alpha$.

У славянских народов числовые значения букв установились в порядке славянского алфавита, который использовал сначала глаголицу, а затем кириллицу (рис. 2.18).

Буквы кириллицы	Цифровое значение кириллицы	Буквы глаголицы	Цифровое значение глаголицы	Кириллическое название
А	1	Ѡ	1	Азь
Б		Ѣ	2	Буки
В	2	Ѥ	3	Вѣди
Г	3	Ѧ	4	Глаголь
Д	4	Ѩ	5	Добро
Е	5	Ѭ	6	Есть
Ж		Ѯ	7	Живѣте
З	6	Ѱ	8	Зѣло
И	7	Ѳ	9	Земля
Н	8	Ѵ	10	Иже
І	10	Ѷ	20	І

Рис. 2.18. Древнерусская алфавитная система счисления

В России славянская нумерация сохранилась до конца XVII века. При Петре I возобладала так называемая арабская нумерация, которой мы пользуемся и сейчас. Славянская нумерация сохранилась только в богослужебных книгах.

Непозиционные системы счисления имеют ряд существенных недостатков:

1. Существует постоянная потребность введения новых знаков для записи больших чисел.

2. Невозможно представлять дробные и отрицательные числа.
3. Сложно выполнять арифметические операции, так как не существует алгоритмов их выполнения.

Контрольные вопросы

1. Существуют ли алгоритмы проведения арифметических действий в непозиционных системах счисления?

Задания

- 2.7. Некоторые римские цифры легко изобразить, используя палочки или спички. Ниже написано несколько неверных равенств. Как можно получить из них верные равенства, если разрешается переложить с одного места на другое только одну спичку (палочку)?

$$VII - V = XI$$


$$VI - IX = III$$

$$IX - V = VI$$

$$VIII - III = X$$

Компьютерный практикум



Windows-CD 

- 2.4. С помощью компьютерного калькулятора (например, NumLock Calculator) выполнить практическое задание «Римская система счисления».



2.7.2. Позиционные системы счисления

В позиционных системах счисления количественный эквивалент (значение) цифры зависит от ее места (позиции) в записи числа. Позиция цифры в числе называется **разрядом**. Разряд числа возрастает справа налево, от младших разрядов к старшим.

Основанием позиционной системы счисления называется целое число, которое равно количеству цифр, используемых для изображения чисел в данной системе счисления. Основание показывает, во сколько раз изменяется количественное значение цифры при перемещении ее в младший или старший разряд.

Позиционные системы с произвольным основанием. Возможно использование множества позиционных систем счисления, основание которых равно или больше 2. В системах счисления с основанием q (q -ичная система счисления) числа в развернутой форме записываются в виде суммы ряда степеней основания q с коэффициентами, в качестве которых выступают цифры 0, 1, $q-1$. Для записи дробей используются разряды с отрицательными значениями степеней основания.

В развернутой форме число в системе счисления с основанием q (q -ичная система счисления) записывается следующим образом:

$$A_q = a_{n-1} \cdot q^{n-1} + a_{n-2} \cdot q^{n-2} + \dots + a_0 \cdot q^0 + a_{-1} \cdot q^{-1} + \dots + a_{-m} \cdot q^{-m} \quad (2.5)$$

или

$$A_q = \pm \sum_{i=-m}^{n-1} a_i q_i. \quad (2.6)$$

Здесь: A_q — число в q -ичной системе счисления,
 q — основание системы счисления,
 a_i — цифры, принадлежащие алфавиту данной системы счисления,
 n — число целых разрядов числа,
 m — число дробных разрядов числа.

Коэффициенты a_i в этой записи являются цифрами числа, записанного в q -ичной системе счисления. Свернутой формой записи числа называется запись в виде:

$$A = a_{n-1} a_{n-2} \dots a_1 a_0, a_{-1} \dots a_{-m}. \quad (2.7)$$

Свернутой формой записи чисел мы и пользуемся в повседневной жизни, ее называют естественной или цифровой.

Десятичная система счисления

Основание: $q = 10$.

Алфавит: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Число в десятичной системе счисления записывается в виде суммы числового ряда степеней основания (в данном случае 10), в качестве коэффициентов которых выступают цифры данного числа. В развернутой форме запись числа A_{10} , которое содержит n целых разрядов числа и m дробных разрядов числа, производится следующим образом:

$$A_{10} = a_{n-1} \cdot 10^{n-1} + a_{n-2} \cdot 10^{n-2} + \dots + a_0 \cdot 10^0 + a_{-1} \cdot 10^{-1} + \dots + a_{-m} \cdot 10^{-m}. \quad (2.8)$$

Коэффициенты a_i в этой записи являются цифрами десятичного числа, которое в свернутой форме записывается следующим образом:

$$A_{10} = a_{n-1}a_{n-2}\dots a_1a_0,a_{-1}\dots a_{-m}. \quad (2.9)$$

Например, десятичное число $555,55_{10}$ в развернутой форме будет записываться следующим образом:

$$555,55_{10} = 5 \cdot 10^2 + 5 \cdot 10^1 + 5 \cdot 10^0 + 5 \cdot 10^{-1} + 5 \cdot 10^{-2}.$$

Умножение или деление десятичного числа на 10 (величину основания) приводит к перемещению запятой, отделяющей целую часть от дробной на один разряд вправо или влево. Например:

$$\begin{aligned} 555,55_{10} \cdot 10 &= 5555,5_{10}; \\ 555,55_{10} : 10 &= 55,555_{10}. \end{aligned}$$

Двоичная система счисления

Основание: $q = 2$.

Алфавит: 0, 1.

Число в двоичной системе счисления записывается в виде суммы числового ряда степеней основания (в данном случае 2), в качестве коэффициентов которых выступают цифры данного числа. В развернутой форме запись числа A_2 , которое содержит n целых разрядов числа и m дробных разрядов числа, производится следующим образом:

$$A_2 = a_{n-1} \cdot 2^{n-1} + a_{n-2} \cdot 2^{n-2} + \dots + a_0 \cdot 2^0 + a_{-1} \cdot 2^{-1} + \dots + a_{-m} \cdot 2^{-m}. \quad (2.10)$$

Коэффициенты a_i в этой записи являются цифрами двоичного числа (0 или 1), которое в свернутой форме записывается следующим образом:

$$A_2 = a_{n-1}a_{n-2}\dots a_1a_0,a_{-1}\dots a_{-m}. \quad (2.11)$$

Умножение или деление двоичного числа на 2 (величину основания) приводит к перемещению запятой, отделяющей целую часть от дробной на один разряд вправо или влево. Например:

$$\begin{aligned} 101,01_2 \cdot 2 &= 1010,1_2; \\ 101,01_2 : 2 &= 10,101_2. \end{aligned}$$

Восьмеричная система счисления

Основание: $q = 8$.

Алфавит: 0, 1, 2, 3, 4, 5, 6, 7.

Число в восьмеричной системе счисления записывается в виде суммы числового ряда степеней основания (в данном

случае 8), в качестве коэффициентов которых выступают цифры данного числа. В развернутой форме запись числа A_8 , которое содержит n целых разрядов числа и m дробных разрядов числа, производится следующим образом:

$$A_8 = a_{n-1} \cdot 8^{n-1} + a_{n-2} \cdot 8^{n-2} + \dots + a_0 \cdot 8^0 + a_{-1} \cdot 8^{-1} + \dots + a_{-m} \cdot 8^{-m}. \quad (2.12)$$

Коэффициенты a_i в этой записи являются цифрами восьмеричного числа, которое в свернутой форме записывается следующим образом:

$$A_8 = a_{n-1}a_{n-2}\dots a_1a_0,a_{-1}\dots a_{-m}. \quad (2.13)$$

Умножение или деление восьмеричного числа на 8 (величину основания) приводит к перемещению запятой, отделяющей целую часть от дробной на один разряд вправо или влево. Например:

$$567,12_8 \cdot 8 = 5671,2_8;$$

$$567,12_8 : 8 = 56,712_8.$$

Шестнадцатеричная система счисления

Основание: $q = 16$.

Алфавит: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Здесь только десять цифр из шестнадцати имеют общепринятое обозначение 0, 1, ..., 9. Для записи остальных цифр (10, 11, 12, 13, 14 и 15) обычно используются первые шесть букв латинского алфавита.

Число в шестнадцатеричной системе счисления записывается в виде суммы числового ряда степеней основания (в данном случае 16), в качестве коэффициентов которых выступают цифры данного числа. В развернутой форме запись числа A_{16} , которое содержит n целых разрядов числа и m дробных разрядов числа, производится следующим образом:

$$A_{16} = a_{n-1} \cdot 16^{n-1} + a_{n-2} \cdot 16^{n-2} + \dots + a_0 \cdot 16^0 + a_{-1} \cdot 16^{-1} + \dots + a_{-m} \cdot 16^{-m}. \quad (2.14)$$

Коэффициенты a_i в этой записи являются цифрами восьмеричного числа, которое в свернутой форме записывается следующим образом:

$$A_{16} = a_{n-1}a_{n-2}\dots a_1a_0,a_{-1}\dots a_{-m}. \quad (2.15)$$

Умножение или деление шестнадцатеричного числа на 16 (величину основания) приводит к перемещению запятой, отделяющей целую часть от дробной на один разряд вправо или влево. Например:

$$ABC,DE_{16} \cdot 16 = ABCD,E_{16};$$

$$ABC,DE_{16} : 16 = AB,CDE_{16}.$$

Контрольные вопросы

1. Чем отличаются позиционные системы счисления от непозиционных?
2. Какое количество цифр используется в q -ичной системе счисления?

З а д а н и я

- 2.8. Во сколько раз увеличатся числа $10,1_{10}$, $10,1_2$, $64,5_8$, $39,F_{16}$ при переносе запятой на один знак вправо?
- 2.9. При переносе запятой на два знака вправо число $11,11_x$ увеличилось в 4 раза. Чему равен x ?

2.8. Перевод чисел из одной системы счисления в другую

2.8.1. Перевод целых чисел из десятичной системы в двоичную, восьмеричную и шестнадцатеричную

Перевод чисел из десятичной системы в двоичную, восьмеричную и шестнадцатеричную может осуществляться различными способами. Рассмотрим один из алгоритмов перевода на примере перевода чисел из десятичной системы в двоичную, при этом необходимо учитывать, что алгоритмы перевода целых чисел и правильных дробей будут различаться.

Вывод алгоритма перевода целых десятичных чисел в двоичную систему счисления

Пусть $A_{\text{цд}}$ — целое десятичное число, разложим его в ряд по основанию 2. Получим запись в развернутой форме двоичного целого числа, так как в этой записи будут отсутствовать отрицательные степени основания, т. е. числа 2:

$$A_{\text{цд}} = a_{n-1} \cdot 2^{n-1} + a_{n-2} \cdot 2^{n-2} + \dots + a_1 \cdot 2^1 + a_0 \cdot 2^0.$$

На первом шаге разделим число $A_{\text{цд}}$ на основание двоичной системы, т. е. на 2. Частное от деления будет равно:

$$a_{n-1} \cdot 2^{n-2} + a_{n-2} \cdot 2^{n-3} + \dots + a_1,$$

а остаток равен a_0 .

На втором шаге целое частное опять разделим на 2, остаток от деления будет теперь равен a_1 .

Если продолжать этот процесс деления, то после n -го шага получим последовательность остатков:

$$a_0, a_1, \dots, a_{n-1}.$$

Легко заметить, что их последовательность совпадает с обратной последовательностью цифр целого двоичного числа, записанного в свернутой форме:

$$A_{\text{цд}} = a_{n-1} \dots a_1 a_0.$$

Таким образом, достаточно записать остатки в обратной последовательности, чтобы получить искомое двоичное число.

Аналогичные рассуждения могут быть проведены и для перевода целых десятичных чисел в восьмеричную и шестнадцатеричную системы счисления.

Алгоритм перевода целых десятичных чисел в двоичную, восьмеричную и шестнадцатеричную системы счисления. Алгоритм перевода целого десятичного числа в целое двоичное, восьмеричное или шестнадцатеричное число будет следующим:

1. Последовательно выполнять деление исходного целого десятичного числа и получаемых целых частных на основание системы (на 2, 8 или 16) до тех пор, пока не получим частное, равное нулю.
2. Получить искомое двоичное, восьмеричное или шестнадцатеричное число, для чего записать полученные остатки в обратной последовательности.

Практическое задание «Перевод целого десятичного числа в целое двоичное, восьмеричное и шестнадцатеричное числа». Осуществить перевод десятичного числа (например, числа 19_{10}) в двоичную, восьмеричную и шестнадцатеричную системы счисления с помощью алгоритма перевода и с помощью компьютерного калькулятора NumLock Calculator.



**Перевод целого десятичного числа
в целые двоичное, восьмеричное
и шестнадцатеричное числа**

Применить алгоритм перевода целого десятичного числа в целое двоичное и записать результаты в таблицу.

Десятичное число/целое частное	Делитель (основание системы)	Остаток	Цифры двоичного числа
19	2	1	a_0 ↑
9	2	1	a_1
4	2	0	a_2
2	2	0	a_3
1	2	1	a_4
0			

В результате получаем двоичное число:

$$A_2 = a_4 a_3 a_2 a_1 a_0 = 10011_2.$$

Применить алгоритм перевода целого десятичного числа в целое восьмеричное и записать результаты в таблицу.

Десятичное число/целое частное	Делитель (основание системы)	Остаток	Цифры восьмеричного числа
19	8	3	a_0 ↑
2	8	2	a_1
0			

В результате получаем восьмеричное число:

$$A_8 = a_1 a_0 = 23_8.$$

Применить алгоритм перевода целого десятичного числа в целое шестнадцатеричное и записать результаты в таблицу.

Десятичное число/целое частное	Делитель (основание системы)	Остаток	Цифры шестнадцатеричного числа
19	16	3	a_0 ↑
1	16	1	a_1
0			

В результате получаем шестнадцатеричное число:

$$A_{16} = a_1 a_0 = 13_{16}.$$



Перевод целого десятичного числа в целые двоичное, восьмеричное и шестнадцатеричное числа

1. Запустить компьютерный калькулятор NumLock Calculator.
2. Установить минималистский вид калькулятора командой [Вид калькулятора-Универсальный].
3. Установить формат исходных данных командой [Формат чисел в выражении-Десятичный] и формат результата командой [Формат результата-Двоичный].
4. В окне ввода данных ввести десятичное число, например 19, и нажать клавишу со знаком «=». Появится результат — двоичное число 10011 (рис. 2.19).

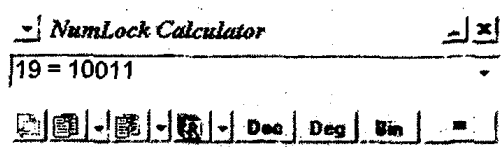


Рис. 2.19. Перевод из десятичной системы счисления в двоичную

5. Установить формат исходных данных командой [Формат чисел в выражении-Десятичный] и формат результата командой [Формат результата-Восьмеричный].
6. В окне ввода данных ввести десятичное число, например 19, и нажать клавишу со знаком «=». Появится результат — восьмеричное число 23 (рис. 2.20).

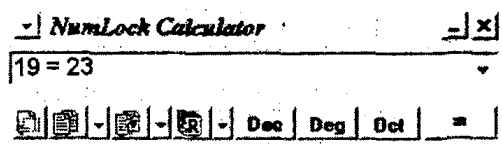


Рис. 2.20. Перевод из десятичной системы счисления в восьмеричную

7. Установить формат исходных данных командой [Формат чисел в выражении-Десятичный] и формат результата командой [Формат результата-Шестнадцатеричный].
8. В окне ввода данных ввести десятичное число, например 19, и нажать клавишу со знаком «=». Появится результат — шестнадцатеричное число 13 (рис. 2.21).

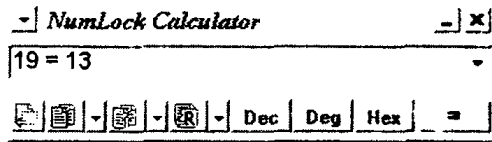


Рис. 2.21. Перевод из десятичной системы счисления в шестнадцатеричную

Контрольные вопросы

1. Сформулируйте алгоритм перевода целых десятичных чисел в двоичную, восьмеричную и шестнадцатеричную системы счисления.

Задания

- 2.10. Заполните таблицу, в каждой строке которой одно и то же целое число должно быть записано в различных системах счисления:

Двоичная	Восьмеричная	Десятичная	Шестнадцатеричная
101010			
	127		
		269	
			9B

Компьютерный практикум

Windows-CD 

- 2.5. Выполнить практическое задание «Перевод целого десятичного числа в целые двоичное, восьмеричное и шестнадцатеричное числа» (для нескольких чисел).



2.8.2. Перевод дробей из десятичной системы в двоичную, восьмеричную и шестнадцатеричную

Вывод алгоритма перевода десятичных дробей в двоичную систему счисления

Пусть $A_{дл}$ — десятичная дробь, разложим ее в ряд по основанию 2. Получим запись в развернутой форме двоичной

дроби, так как в этой записи будут отсутствовать положительные степени основания, т. е. числа 2:

$$A_{\text{дд}} = a_{-1} \cdot 2^{-1} + a_{-2} \cdot 2^{-2} + \dots$$

На первом шаге умножим число $A_{\text{дд}}$ на основание двоичной системы, т. е. на 2. Произведение будет равно:

$$a_{-1} + a_{-2} \cdot 2^{-1} + \dots$$

Целая часть равна a_{-1} , именно это число и является значением первого дробного разряда двоичного числа.

На втором шаге оставшуюся дробную часть опять умножим на 2, получим целую часть, равную a_{-2} , именно это число и является значением второго дробного разряда двоичного числа.

Описанный процесс необходимо продолжать до тех пор, пока в результате умножения мы не получим нулевую дробную часть или не будет достигнута требуемая точность вычислений.

Легко заметить, что последовательность полученных чисел совпадает с последовательностью цифр дробного двоичного числа, записанного в свернутой форме:

$$A_{\text{дд}} = a_{-1} a_{-2} \dots$$

Аналогичные рассуждения могут быть проведены и для перевода дробных десятичных чисел в восьмеричную и шестнадцатеричную системы счисления.

Алгоритм перевода дробных десятичных чисел в двоичную, восьмеричную и шестнадцатеричную системы счисления. Алгоритм перевода десятичной дроби в двоичную, восьмеричную или шестнадцатеричную дробь будет следующим:

1. Последовательно выполнять умножение исходной десятичной дроби и получаемых дробей на основание системы счисления (на 2, 8 или 16) до тех пор, пока не получим нулевую дробную часть или не будет достигнута требуемая точность вычислений.
2. Получить искомую двоичную, восьмеричную или шестнадцатеричную дробь, записав полученные целые части произведения в прямой последовательности.

Задание «Перевод десятичной дроби в двоичную, восьмеричную и шестнадцатеричную дроби». Осуществить перевод десятичной дроби (например, числа $0,65625_{10}$) в двоичную, восьмеричную и шестнадцатеричную системы счисления с помощью алгоритма перевода.



Перевод десятичной дроби в двоичную, восьмеричную и шестнадцатеричную дроби

Применить алгоритм перевода дробного десятичного числа в дробное двоичное и записать результаты в таблицу.

Десятичная дробь/ дробная часть произведения	Множитель (основание системы)	Целая часть произведения	Цифры двоичной дроби
0,65625	2	1	a_{-1}
0,3125	2	0	a_{-2}
0,625	2	1	a_{-3}
0,25	2	0	a_{-4}
0,5	2	1	a_{-5} ▼
0,0			

В результате получаем двоичную дробь:

$$A_2 = 0, a_{-1} a_{-2} a_{-3} a_{-4} a_{-5} = 0,10101_2.$$

Применить алгоритм перевода дробного десятичного числа в дробное восьмеричное и записать результаты в таблицу.

Десятичная дробь/ дробная часть произведения	Множитель (основание системы)	Целая часть произведения	Цифры восьмерич- ной дроби
0,65625	8	5	a_{-1}
0,25	8	2	a_{-2} ▼
0,0			

В результате получаем восьмеричную дробь:

$$A_8 = 0, a_{-1} a_{-2} = 0,52_8.$$

Применить алгоритм перевода дробного десятичного числа в дробное шестнадцатеричное и записать результаты в таблицу.

Десятичная дробь/ дробная часть произведения	Множитель (основание системы)	Целая часть произ- ведения	Цифры шест- надцатерич- ной дроби
0,65625	16	10 (A)	a_{-1}
0,5	16	8	a_{-2} ▼
0,0			

В результате получаем шестнадцатеричную дробь:

$$A_{16} = 0, a_{-1} a_{-2} = 0, A8_{16}.$$

Перевод чисел, содержащих и целую, и дробную части, производится в два этапа. Отдельно переводится по соответствующему алгоритму целая часть и отдельно — дробная. В итоговой записи полученного числа целая часть от дробной отделяется запятой.

Контрольные вопросы

1. Сформулируйте алгоритм перевода целых десятичных чисел в двоичную, восьмеричную и шестнадцатеричную системы счисления.

З а д а н и я

- 2.11. Выполнить задание «Перевод десятичной дроби в двоичную, восьмеричную и шестнадцатеричную дроби» (для нескольких дробей).
 2.12. Заполнить таблицу, в каждой строке которой одно и то же дробное число должно быть записано в различных системах счисления:

Двоичная	Восьмеричная	Десятичная	Шестнадцатеричная
0,101			
	0,6		
		0,125	
			0,4

2.8.3. Перевод чисел из двоичной системы в восьмеричную и шестнадцатеричную и обратно

Перевод чисел между системами счисления, основания которых являются степенями числа 2 ($q = 2^n$), может производиться по более простым алгоритмам. Такие алгоритмы могут применяться для перевода чисел между двоичной ($q = 2^1$), восьмеричной ($q = 2^3$) и шестнадцатеричной ($q = 2^4$) системами счисления.

Перевод чисел из двоичной системы счисления в восьмеричную

Для записи двоичных чисел используются две цифры, т. е. в каждом разряде числа возможны два варианта записи.

Для определения количества информации, которое содержит один двоичный разряд, воспользуемся формулой (2.2). Решаем показательное уравнение:

$$2 = 2^I, \text{ так как } 2 = 2^1, \text{ то } I = 1 \text{ бит.}$$

Каждый разряд двоичного числа содержит 1 бит.

Для записи восьмеричных чисел используются восемь цифр, т. е. в каждом разряде числа возможны восемь вариантов записи. Решаем показательное уравнение:

$$8 = 2^I, \text{ так как } 8 = 2^3, \text{ то } I = 3 \text{ бита.}$$

Каждый разряд восьмеричного числа содержит 3 бита. Таким образом, для перевода двоичного числа в восьмеричное двоичное число нужно разбить на группы по три цифры. Для упрощения перевода можно заранее подготовить таблицу преобразования двоичных триад (групп по 3 цифры) в восьмеричные цифры (табл. 2.1).

Таблица 2.2. Двоичные триады

Двоичные триады	000	001	010	110	100	101	110	111
Шестнадцатеричные цифры	0	1	2	3	4	5	6	7

Перевод целых чисел. Для перевода целого двоичного числа в восьмеричное двоичное число нужно разбить на группы по три цифры, справа налево; если в последней левой группе окажется меньше чем три разряда, то необходимо ее дополнить слева нулями. Затем надо преобразовать каждую группу в восьмеричную цифру.

Переведем таким способом двоичное число 101001_2 в восьмеричное число. Получаем:

$$101\ 001_2 = 51_8.$$

Перевод дробей. Для перевода дробного двоичного числа в восьмеричное необходимо разбить двоичное число на триады слева направо; если в последней правой группе окажется меньше разрядов, надо дополнить ее справа нулями. Далее следует триады заменить на восьмеричные числа.

Например, преобразуем дробное двоичное число $A_2 = 0,110101_2$ в восьмеричную систему счисления. Получаем:

$$0,110101_2 = 0,35_8.$$

Например, преобразуем дробное двоичное число $A_2 = 0,110101_2$ в восьмеричную систему счисления. Получаем:

$$0,110101_2 = 0,35_8.$$

Перевод чисел из двоичной системы счисления в шестнадцатеричную

Для записи шестнадцатеричных чисел используются шестнадцать цифр, т. е. в каждом разряде числа возможны шестнадцать вариантов записи. Решаем показательное уравнение:

$$16 = 2^I, \text{ так как } 16 = 2^4, \text{ то } I = 4 \text{ бита.}$$

Каждый разряд шестнадцатеричного числа содержит 4 бита. Таким образом, для перевода двоичного числа в шестнадцатеричное двоичное число нужно разбить на группы по четыре цифры. Для упрощения перевода можно заранее подготовить таблицу преобразования двоичных тетрад (групп по 4 цифры) в шестнадцатеричные цифры (табл. 2.3).

Таблица 2.3. Двоичные тетрады

Двоичные тетрады	0000	0001	0010	0110	0100	0101	0110	0111
Шестнадцатеричные цифры	0	1	2	3	4	5	6	7
Двоичные тетрады	1000	1001	1010	1011	1100	1101	1110	1111
Шестнадцатеричные цифры	8	9	A	B	C	D	E	F

Перевод целых чисел. Для перевода целого двоичного числа в шестнадцатеричное его нужно разбить на группы по четыре цифры (тетрады), начиная справа; если в последней левой группе окажется меньше разрядов, надо дополнить ее слева нулями.

Переведем целое двоичное число $A_2 = 101001_2$ в шестнадцатеричное:

$$0010 \ 1001_2 = 29_{16}.$$

Перевод дробей. Для перевода дробного двоичного числа в шестнадцатеричное необходимо разбить его на тетрады слева направо; если в последней правой группе окажется меньше чем четыре разряда, необходимо ее дополнить справа нулями.

Переведем дробное двоичное число $A_2 = 0,110101_2$ в шестнадцатеричную систему счисления:

$$0,1101 \ 0100_2 = 0,D4_{16}.$$

Для того чтобы преобразовать любое двоичное число в восьмеричную или шестнадцатеричную системы счисления, не-

обходимо произвести преобразования по рассмотренным выше алгоритмам отдельно для его целой и дробной частей.

Перевод чисел из восьмеричной и шестнадцатеричной систем счисления в двоичную

Для перевода чисел из восьмеричной и шестнадцатеричной систем счисления в двоичную необходимо цифры числа преобразовать в группы двоичных чисел. Для перевода из восьмеричной системы в двоичную каждую цифру числа надо преобразовать в группу из трех двоичных разрядов (триаду), а при преобразовании шестнадцатеричного числа — в группу из четырех разрядов (тетраду).

Например, преобразуем дробное восьмеричное число $A_8 = 0,47_8$ в двоичную систему счисления:

$$0,47_8 = 0,100111_2.$$

Переведем целое шестнадцатеричное число $A_{16} = AB_{16}$ в двоичную систему счисления:

$$AB_{16} = 10101011_2.$$

Контрольные вопросы

1. Сформулируйте алгоритм перевода целых двоичных чисел в восьмеричную и шестнадцатеричную системы счисления.
2. Сформулируйте алгоритм перевода дробных двоичных чисел в восьмеричную и шестнадцатеричную системы счисления.

Задания

- 2.13. Заполнить таблицу, в каждой строке которой одно и то же произвольное число (число может содержать как целую, так и дробную часть) должно быть записано в различных системах счисления:

Двоичная	Восьмеричная	Шестнадцатеричная
111101,1		
	233,5	
		59,B

2.9. Арифметические операции в позиционных системах счисления

Арифметические операции во всех позиционных системах счисления выполняются по одним и тем же хорошо известным правилам.

Сложение. Рассмотрим сложение чисел в двоичной системе счисления. В его основе лежит таблица сложения одноразрядных двоичных чисел:

$$\begin{aligned} 0 + 0 &= 0 \\ 0 + 1 &= 1 \\ 1 + 0 &= 1 \\ 1 + 1 &= 10 \end{aligned}$$

Важно обратить внимание на то, что при сложении двух единиц происходит переполнение разряда и производится перенос в старший разряд. Переполнение разряда наступает тогда, когда величина числа в нем становится равной или большей основания системы счисления. Для двоичной системы счисления эта величина равна двум.

Сложение многоразрядных двоичных чисел происходит в соответствии с вышеприведенной таблицей сложения с учетом возможных переносов из младших разрядов в старшие. В качестве примера сложим в столбик двоичные числа 110_2 и 11_2 .

$$\begin{array}{r} 110_2 \\ + 11_2 \\ \hline 1001_2 \end{array}$$

Проверим правильность вычислений сложением в десятичной системе счисления. Переведем двоичные числа в десятичную систему счисления и затем их сложим.

$$110_2 = 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 6_{10};$$

$$11_2 = 1 \cdot 2^1 + 1 \cdot 2^0 = 3_{10};$$

$$6_{10} + 3_{10} = 9_{10}.$$

Теперь переведем результат двоичного сложения в десятичное число.

$$1001_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 9_{10}.$$

Сравним результаты, сложение выполнено правильно.

Вычитание. Рассмотрим вычитание двоичных чисел. В его основе лежит таблица вычитания одноразрядных двоичных чисел. При вычитании из меньшего числа (0) большего (1) производится заем из старшего разряда. В таблице заем обозначен 1 с чертой.

$$\begin{aligned} 0 - 0 &= \underline{0} \\ 0 - 1 &= \overline{11} \\ 1 - 0 &= 1 \\ 1 - 1 &= 0 \end{aligned}$$

Вычитание многоразрядных двоичных чисел происходит в соответствии с вышеприведенной таблицей вычитания с учетом возможных заемов в старших разрядах. В качестве примера произведем вычитание двоичных чисел 110_2 и 11_2 .

$$\begin{array}{r} 110_2 \\ - 11_2 \\ \hline 11_2 \end{array}$$

Умножение. В основе умножения лежит таблица умножения одnorазрядных двоичных чисел:

$$\begin{aligned} 0 \cdot 0 &= 0 \\ 0 \cdot 1 &= 0 \\ 1 \cdot 0 &= 0 \\ 1 \cdot 1 &= 1 \end{aligned}$$

Умножение многоразрядных двоичных чисел происходит в соответствии с вышеприведенной таблицей умножения по обычной схеме, применяемой в десятичной системе счисления, с последовательным умножением множимого на очередную цифру множителя. В качестве примера произведем умножение двоичных чисел 110_2 и 11_2 .

$$\begin{array}{r} 110_2 \\ \times 11_2 \\ \hline 110 \\ 110 \\ \hline 10010_2 \end{array}$$

Деление. Операция деления выполняется по алгоритму, подобному алгоритму выполнения операции деления в десятичной системе счисления. В качестве примера произведем деление двоичного числа 110_2 на 11_2 .

$$\begin{array}{r} 110_2 : 11_2 = \\ \underline{11} \\ 0 \end{array}$$

Арифметические операции в восьмеричной и шестнадцатеричной системах счисления. Аналогично можно выполнять арифметические действия в восьмеричной и шестнадцатерич-

ной системах счисления. Необходимо только помнить, что перенос в следующий разряд при сложении и заем из старшего разряда при вычитании определяется величиной основания системы счисления:

$$\begin{array}{r} 37_8 \\ + 25_8 \\ \hline 64_8 \end{array} \qquad \begin{array}{r} 9C_{16} \\ - 78_{16} \\ \hline 24_{16} \end{array}$$

Для проведения арифметических операций над числами, выраженными в различных системах счисления, необходимо предварительно перевести их в одну и ту же систему.

Практическое задание «Арифметические операции в позиционных системах счисления». Выполнить сложение, вычитание, умножение и деление чисел, выраженных в различных системах счисления (например, 12_8 и 64_{16}), с помощью компьютерного калькулятора Wise Calculator.



Арифметические операции в позиционных системах счисления

1. Запустить Wise Calculator и ввести команду [*Tools-Multi-Base Calculator...*].

В режиме *Multi-Base Calculator* появляется многооконная панель, позволяющая ввести число в любой системе счисления и автоматически получить значения этого числа в других системах счисления. В режиме *Multi-Base Calculator* возможно проведение арифметических операций (сложение, вычитание, умножение и деление) в различных системах счисления. Для этого в окна операндов *A* и *B* необходимо ввести числа (можно в различных системах счисления) и выбрать в группе переключателей *Operation* арифметическую операцию.

2. На появившейся многооконной панели *Multi-Base Calculator* (рис. 2.22) ввести пару чисел в окна операндов *A* и *B*. Например, шестнадцатеричное число 64_{16} в окно *HEX* (операнд *A*) и восьмеричное число 12_8 в окно *OCT* (операнд *B*).
3. Выбрать в группе переключателей *Operation*, например операцию умножения $A*B$.
4. В окнах *Result* появится результат выполнения арифметической операции одновременно в четырех системах счисления: *DEC* (1000_{10}), *BIN* (1111101000_2), *OCT* (1750_8) и *HEX* ($3E8_{16}$).

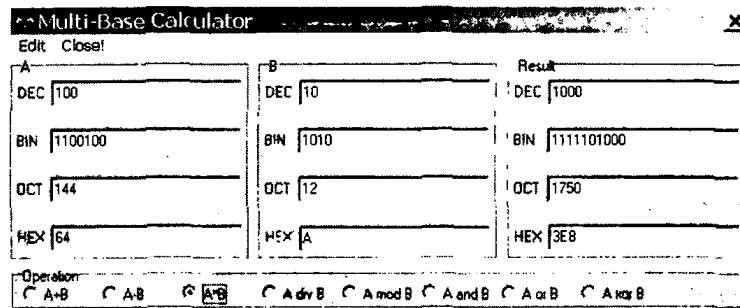



Рис. 2.22. Умножение чисел в различных системах счисления

5. Последовательно выбрать в группе переключателей *Operation* операции сложения $A+B$, вычитания $A-B$ и деления $A \div B$.

Компьютерный практикум

Windows-CD 

- 2.6. Выполнить с использованием компьютерного калькулятора *Wise Calculator* практическое задание «Арифметические операции в позиционных системах счисления».



2.10. Представление чисел в компьютере

2.10.1. Представление чисел в формате с фиксированной запятой

Целые числа в компьютере хранятся в памяти в формате с **фиксированной запятой**. В этом случае каждому разряду ячейки памяти соответствует всегда один и тот же разряд числа, а запятая находится справа после младшего разряда, т. е. вне разрядной сетки.

Достоинствами представления чисел в формате с фиксированной запятой являются простота и наглядность представления чисел, а также простота алгоритмов реализации арифметических операций (вычитание благодаря использованию дополнительного кода (см. ниже) для представления отрицательных чисел сводится к сложению).

Недостатком представления чисел в формате с фиксированной запятой является конечный диапазон представления

величин, недостаточный для решения математических, физических, экономических и других задач, в которых используются как очень малые, так и очень большие числа.

Целые неотрицательные числа. Для хранения целых неотрицательных чисел отводится одна ячейка памяти (8 битов). Например, число $A_2 = 10101010_2$ будет храниться в ячейке памяти следующим образом:

1	0	1	0	1	0	1	0
---	---	---	---	---	---	---	---

Минимальное число соответствует восьми нулям, хранящимся в восьми ячейках памяти, и равно нулю.

Максимальное значение целого неотрицательного числа достигается в случае, когда во всех ячейках хранятся единицы. Для n -разрядного представления оно будет равно

$$2^n - 1.$$

Задание «Диапазон хранения целых неотрицательных чисел». Определить диапазон чисел, которые могут храниться в оперативной памяти в формате *целого неотрицательного числа*.

Минимальное число равно 0.

Максимальное число соответствует восьми единицам, хранящимся в ячейках памяти, и равно:

$$A = 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 1 \cdot 2^8 - 1 = 255_{10}.$$

Диапазон изменения целых неотрицательных чисел — от 0 до 255.

Целые числа со знаком. Для хранения *целых чисел со знаком* отводится две ячейки памяти (16 битов), а для хранения *больших целых чисел со знаком* отводится четыре ячейки памяти (32 бита). Старший (левый) разряд отводится под знак числа (если число положительное, то в знаковый разряд записывается 0, если число отрицательное — записывается 1).

Представление в компьютере положительных чисел с использованием формата «знак-величина» называется **прямым кодом** числа. Например, число $2002_{10} = 11111010010_2$ будет представлено в 16-разрядном представлении следующим образом:

0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

При представлении целых чисел в n -разрядном представлении со знаком максимальное положительное число (с учетом выделения одного разряда на знак) равно

$$A = 2^{n-1} - 1.$$

Минимальное отрицательное число равно

$$A = -2^{n-1}.$$

Задание «Диапазон хранения целых чисел со знаком и больших целых чисел со знаком». Определить диапазон чисел, которые могут храниться в оперативной памяти в формате:

- *целых чисел со знаком;*
- *больших целых чисел со знаком.*

а) Максимальное положительное целое число в формате *целых чисел со знаком* равно:

$$A_{10} = 2^{15} - 1 = 32\,767_{10}.$$

Минимальное отрицательное целое число равно

$$A_{10} = -2^{15} = -32\,768_{10}.$$

б) Максимальное положительное целое число в формате *больших целых чисел со знаком* равно

$$A_{10} = 2^{31} - 1 = 2\,147\,483\,647_{10}.$$

Минимальное отрицательное целое число равно

$$A_{10} = -2^{31} = -2\,147\,483\,648_{10}.$$

Дополнительный код. Для представления отрицательных чисел используется дополнительный код. Дополнительный код позволяет заменить арифметическую операцию вычитания операцией сложения, что существенно упрощает работу процессора и увеличивает его быстродействие.



Дополнительный код отрицательного числа A , хранящегося в n ячейках, равен $2^n - |A|$.

Дополнительный код представляет собой дополнение модуля отрицательного числа A до 0, поэтому в n -разрядной компьютерной арифметике:

$$2^n - |A| + |A| = 0.$$

Это равенство тождественно справедливо, так как в компьютерной n -разрядной арифметике $2^n \equiv 0$. Действительно, двоичная запись такого числа состоит из одной единицы и n нулей, а в n -разрядную ячейку может поместиться только n младших разрядов, т. е. n нулей.

Задание «Дополнительный код числа». Записать дополнительный код отрицательного числа -2002 для 16-разрядного компьютерного представления.

Проведем вычисления в соответствии с определением дополнительного кода:

2^{16}	=	1000000000000000_2	65536_{10}
2002_{10}	=	0000011111010010_2	2002_{10}
$2^{16} - 2002_{10} $	=	1111100000101110_2	63534_{10}

Проведем проверку с использованием десятичной системы счисления. Дополнительный код $63\ 534_{10}$ в сумме с модулем отрицательного числа 2002_{10} равен $65\ 536_{10}$, т. е. дополнительный код дополняет модуль отрицательного числа до 2^{16} (до нуля 16-разрядной компьютерной арифметики).

Алгоритм получения дополнительного кода. Для получения дополнительного кода отрицательного числа можно использовать довольно простой алгоритм:

1. Модуль числа записать прямым кодом в n двоичных разрядах.
2. Получить **обратный код** числа, для этого значения всех битов инвертировать (все единицы заменить на нули и все нули заменить на единицы).
3. К полученному обратному коду прибавить единицу.

Задание «Вычисление дополнительного кода числа с использованием обратного кода». Вычислить дополнительный код отрицательного числа -2002 для 16-разрядного компьютерного представления с использованием алгоритма.

Прямой код	$ -2002_{10} $	0000011111010010_2
Обратный код	Инвертирование	1111100000101101_2
	Прибавление единицы	1111100000101101_2 + 0000000000000001_2
Дополнительный код		1111100000101110_2

Задание «Арифметическое действие». Выполнить арифметическое действие $3000_{10} - 5000_{10}$ в 16-разрядном компьютерном представлении.

Представим положительное число в прямом, а отрицательное число — в дополнительном коде:

Десятичное число	Прямой код	Обратный код	Дополнительный код
3000	0000101110111000		
-5000	0001001110001000	1110110001110111	+ 1110110001110111 0000000000000001 1110110001110000

Сложим прямой код положительного числа с дополнительным кодом отрицательного числа. Получим результат в дополнительном коде:

3000-5000			1111100000110000
-----------	--	--	------------------

Переведем полученный дополнительный код в десятичное число:

- 1) Инвертируем дополнительный код: 0000011111001111.
- 2) Прибавим к полученному коду 1 и получим модуль отрицательного числа:

$$\begin{array}{r}
 0000011111001111 \\
 + \quad 0000000000000001 \\
 \hline
 0000011111010000
 \end{array}$$

- 3) Переведем в десятичное число и припишем знак отрицательного числа: -2000.

Контрольные вопросы

1. Чем различаются обратный и дополнительный коды числа в компьютерном представлении отрицательных чисел?

Задачи

- 2.14. Заполнить таблицу, записав отрицательные десятичные числа в прямом, обратном и дополнительном кодах в 16-разрядном представлении:

Десятичные числа	Обратный код	Дополнительный код
-10		
-100		
-1000		
-10000		

2.15. Выполнить арифметическое действие $20_{10} - 60_{10}$ в 16-разрядном компьютерном представлении.

2.10.2. Представление чисел в формате с плавающей запятой

Вещественные числа (конечные и бесконечные десятичные дроби) хранятся и обрабатываются в компьютере в формате с плавающей запятой. В этом случае положение запятой в записи числа может изменяться.

Формат чисел с плавающей запятой базируется на экспоненциальной форме записи, в которой может быть представлено любое число. Так число A может быть представлено в виде:

$$A = m \times q^n, \quad (2.16)$$

где m — мантисса числа,

q — основание системы счисления,

n — порядок числа.

Для однозначности представления чисел с плавающей запятой используется нормализованная форма, при которой мантисса отвечает условию:

$$1/n \leq |m| < 1.$$

Это означает, что мантисса должна быть правильной дробью и иметь после запятой цифру, отличную от нуля.

Задание «Приведение числа с плавающей запятой к нормализованной форме». Преобразовать десятичное число 888,888, записанное в естественной форме, в экспоненциальную форму с нормализованной мантиссой.

$$888,888 = 0,888888 \times 10^3.$$

Нормализованная мантисса $m = 0,888888$, порядок $n = 3$.

Число в форме с плавающей запятой занимает в памяти компьютера четыре байта (*число обычной точности*) или восемь байтов (*число двойной точности*). При записи числа с плавающей запятой выделяются разряды для хранения знака мантиссы, знака порядка, порядка и мантиссы.

Диапазон изменения чисел определяется количеством разрядов, отведенных для хранения порядка числа, а точность (количество значащих цифр) определяется количеством разрядов, отведенных для хранения мантиссы.

Задание «Определение максимального числа и его точности». Определить максимальное число и его точность для формата чисел *обычной точности*, если для хранения порядка и

- сложение и вычитание;
- умножение и деление.

а) Произведем выравнивание порядков, сложение или вычитание мантисс, а также нормализацию (если это необходимо):

Сложение	Вычитание
$0,100 \times 2^5$	$0,100 \times 2^5$
$+ 0,001 \times 2^5$	$- 0,001 \times 2^5$
$0,101 \times 2^5$	$0,010 \times 2^5 = 0,10 \times 2^4$

б) Произведем умножение или деление мантисс, сложение или вычитание порядков, а также нормализацию (если это необходимо):

Умножение	Деление
$0,1 \times 2^5$	$0,1 \times 2^5$
$\times 0,1 \times 2^3$	$- 0,1 \times 2^5$
$0,01 \times 2^8 = 0,1 \times 2^7$	$1 \times 2^2 = 0,1 \times 2^3$

Контрольные вопросы


1. От чего зависит максимальное значение порядка числа и его точность в формате с плавающей запятой?

З а д а н и я

- 2.16. Определить максимальное число и его точность для формата чисел двойной точности, если для хранения порядка и его знака отводится 11 разрядов, а для хранения мантиссы и ее знака — 53 разряда.
- 2.17. Произвести сложение, вычитание, умножение и деление чисел $0,1 \times 2^2$ и $0,1 \times 2^{-2}$ в формате с плавающей запятой.

Глава 3


Основы логики и логические основы компьютера

Windows-CD 

В процессе изучения данной темы рекомендуется использовать программное обеспечение для операционной системы Windows:

- компьютерный калькулятор NumLock Calculator;
- электронные таблицы OpenOffice.org Calc;
- редактор электрических и логических схем sPlan;
- конструктор электрических схем Начала электроники;



Дистрибутив Microsoft Office 

- электронные таблицы Microsoft Excel.



3.1. Формы мышления

Первые учения о формах и способах рассуждений возникли в странах Древнего Востока (Китай, Индия), но в основе современной логики лежат учения, созданные древнегреческими мыслителями. Основы формальной логики заложил Аристотель, который впервые отделил логические формы мышления (речи) от его содержания.



Логика — это наука о формах и способах мышления.

Законы логики отражают в сознании человека свойства, связи и отношения объектов окружающего мира. Логика позволяет строить формальные модели окружающего мира, отвлекаясь от содержательной стороны.

Мышление всегда осуществляется в каких-то формах. Основными формами мышления являются **понятие, высказывание (суждение) и умозаключение**.

Понятие. Понятие — это форма мышления, отражающая наиболее существенные признаки предмета, отличающие его от других предметов. В структуре каждого понятия нужно различать две стороны: содержание и объем. Содержание понятия составляет совокупность существенных признаков предмета. Чтобы раскрыть содержание понятия, следует выделить признаки, необходимые и достаточные для выделения данного объекта по отношению к другим объектам.

Например, понятие «компьютер» объединяет множество электронных устройств, которые предназначены для обработки информации и обладают монитором и клавиатурой. Даже по этому короткому описанию компьютер трудно спутать с другими объектами, например с механизмами, служащими для перемещения по дорогам и хранящимися в гаражах, которые объединяются понятием «автомобиль».

Объем понятия определяется совокупностью предметов, на которую понятие распространяется. Объем понятия «компьютер» выражает всю совокупность существовавших, существующих и могущих существовать в будущем компьютеров.

Объем и содержание понятия связаны между собой, и эта связь выражается следующим законом: чем больше объем понятия, тем меньше его содержание, и наоборот, чем больше содержание понятия, тем меньше его объем. Иначе говоря, чем меньшее количество вещей мыслится в данном понятии, тем больше оно сообщает об этих вещах. Например, понятие «карманный компьютер» охватывает меньший объем, чем понятие «компьютер», но обладает большим содержанием.



Понятие — это форма мышления, фиксирующая основные, существенные признаки объекта.



Алгебра множеств, одна из основополагающих современных математических теорий, позволяет исследовать отношения между множествами и, соответственно, объемами понятий.

Для наглядной геометрической иллюстрации объемов понятий и соотношений между ними используются **диаграммы Эйлера–Венна**. Если имеются какие-либо понятия A , B , C и т. д., то объем каждого понятия (множество) можно представить в виде круга, а отношения между этими объемами (множествами) — в виде пересекающихся кругов.

Отообразим с помощью диаграммы Эйлера–Венна соотношение между объемами понятий «натуральные числа» и «четные числа». Объем понятия «натуральные числа» включает в себя множество целых положительных чисел A , а объем понятия «четные числа» включает в себя множество отрицательных и положительных четных чисел B . Эти множества пересекаются, так как оба включают в себя множество положительных четных чисел C (рис. 3.1).

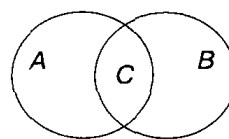


Рис. 3.1.
Представление объемов понятий с использованием диаграммы Эйлера–Венна

Высказывание. Высказывание (суждение) — это форма мышления, выраженная с помощью понятий, посредством которой что-либо утверждают или отрицают о предметах, их свойствах и отношениях между ними.

О предметах можно судить верно или неверно, т. е. высказывание может быть **истинным** или **ложным**. Истинным будет высказывание, в котором связь понятий правильно отражает свойства и отношения реальных вещей. Ложным высказывание будет в том случае, когда связь понятий искажает объективные отношения, не соответствует реальной действительности.

Обоснование истинности или ложности простых высказываний решается вне алгебры логики. Например, истинность или ложность высказывания «Сумма углов треугольника равна 180 градусам» устанавливается геометрией, причем в геометрии Евклида это высказывание является истинным, а в геометрии Лобачевского — ложным.

На естественном языке высказывания выражаются повествовательными предложениями. Высказывание не может быть выражено повелительным или вопросительным предложением, потому что невозможна оценка их истинности или ложности.

Высказывания могут быть выражены не только с помощью естественных языков, но и с помощью формальных языков. Из двух числовых выражений можно составить высказы-

вания, соединив их знаками равенства или неравенства. Например, высказывание на естественном языке имеет вид «Два умножить на два равно четырем», а на формальном, математическом языке оно записывается в виде « $2 \times 2 = 4$ ».



Высказывание — это форма мышления, в которой что-либо утверждается или отрицается о реальных предметах, их свойствах и отношениях между ними. Высказывание может быть либо истинно, либо ложно.

Высказывание называется **простым**, если никакая его часть сама не является высказыванием. Высказывание, состоящее из простых высказываний, называется **составным** (сложным).



Высказывание состоит из понятий, и его можно сравнить с арифметическим выражением, в котором все числа заданы. В математической логике рассматриваются **предикаты**, т. е. функциональные зависимости от неопределенных понятий (терминов), которые можно сравнить с переменными в уравнении.

В предикатах первого порядка один из терминов является неопределенным понятием (например, « x — человек»).

В предикатах второго порядка два термина не определены (например, « x любит y »).

В предикатах третьего порядка неопределены три термина (например, « z — сын x и y »).

Преобразуем рассмотренные выше предикаты в высказывания путем подстановки вместо переменных соответствующих понятий: x = «Сократ», y = «Ксантиппа», z = «Софрониск». Получим высказывания:

«Сократ — человек»;

«Ксантиппа любит Сократа»;

«Софрониск — сын Сократа и Ксантиппы».

Умозаключение. Умозаключение — это форма мышления, посредством которой из одного или нескольких высказываний, называемых **посылками**, по определенным правилам логического вывода получается новое знание о предметах реального мира (**вывод**).

Умозаключения бывают дедуктивные, индуктивные и по аналогии. В дедуктивных умозаключениях рассуждения ведутся от общего к частному. Например, из двух суждений: «Все металлы электропроводны» и «Ртуть является металлом» — путем умозаключения можно сделать вывод: «Ртуть электропроводна».

В индуктивных умозаклЮчениях рассуждения ведутся от частного к общему. Например, установив, что отдельные металлы — железо, медь, цинк, алюминий и т. д. — обладают свойством электропроводности, можно сделать вывод, что все металлы электропроводны.

УмозаклЮчение по аналогии представляет собой движение мысли от общности одних свойств и отношений у сравниваемых предметов или процессов к общности других свойств и отношений. Например, химический состав Солнца и Земли сходен по многим показателям, поэтому когда на Солнце обнаружили неизвестный еще на Земле химический элемент гелий, то по аналогии заключили: такой элемент есть и на Земле.



УмозаклЮчение — это форма мышления, с помощью которой из одного или нескольких высказываний (посылок) может быть получено новое высказывание (вывод).

Доказательство. Доказательство есть мыслительный процесс, направленный на подтверждение или опровержение какого-либо положения посредством других несомненных, ранее обоснованных доводов. Доказательство по своей логической форме не отличается от умозаклЮчения. Однако, если в умозаклЮчении заранее исходят из истинности посылок и следят только за правильностью логического вывода, в доказательстве подвергается логической проверке истинность самих посылок.

Примером умозаклЮчений могут быть геометрические доказательства. Например, если мы имеем суждение «Все углы треугольника равны», то мы можем путем умозаклЮчения доказать, что в этом случае справедливо суждение «Этот треугольник равносторонний».

Контрольные вопросы

1. В чем состоит разница между содержанием и объемом понятия? Связаны ли между собой содержание и объем понятия? Приведите примеры.
2. Как определяется истинность или ложность простого высказывания?

З а д а н и я

- 3.1. Привести примеры понятий, высказываний, умозаключений и доказательств из различных наук: математики; информатики; физики и химии.
- 3.2. Представить с использованием диаграммы Эйлера–Венна соотношение между объемами понятий «четные числа» и «нечетные числа».
- 3.3. Построить высказывания на основе предиката второго порядка « x состоит из y ».

3.2. Алгебра логики

3.2.1. Логическое умножение, сложение и отрицание

Алгебра в широком смысле этого слова — наука об общих операциях, аналогичных сложению и умножению, которые могут выполняться над различными математическими объектами (алгебра переменных и функций, алгебра векторов, алгебра множеств и т. д.). Объектами алгебры логики являются высказывания.

Алгебра логики отвлекается от смысловой содержательности высказываний. Ее интересует только один факт — истинно или ложно данное высказывание, что дает возможность определять истинность или ложность составных высказываний алгебраическими методами.

Логические переменные. Простые высказывания в алгебре логики обозначаются прописными латинскими буквами. Высказывания, как уже говорилось ранее, могут быть истинными или ложными. Истинному высказыванию соответствует значение логической переменной 1, а ложному — значение 0.



В алгебре логики высказывания обозначаются именами логических переменных, которые могут принимать лишь два значения: «истина» (1) и «ложь» (0).

Рассмотрим два простых высказывания:

A — «Два умножить на два равно четырем».

B — «Два умножить на два равно пяти».

Первое высказывание истинно ($A = 1$), а второе ложно ($B = 0$).

Составные высказывания на естественном языке образуются с помощью связок «и», «или», «не», которые в алгебре логики заменяются на логические операции. Логические операции задаются таблицами истинности.

Логическое умножение (конъюнкция). Объединение двух (или нескольких) высказываний в одно с помощью союза «и» называется операцией логического умножения или конъюнкцией.



Составное высказывание, образованное в результате операции логического умножения (конъюнкции), истинно тогда и только тогда, когда истинны все входящие в него простые высказывания.

Из приведенных ниже четырех составных высказываний, образованных с помощью операции логического умножения, истинно только четвертое, так как в первых трех составных высказываниях хотя бы одно из простых высказываний ложно:

- 1) « $2 \times 2 = 5$ и $3 \times 3 = 10$ »;
- 2) « $2 \times 2 = 5$ и $3 \times 3 = 9$ »;
- 3) « $2 \times 2 = 4$ и $3 \times 3 = 10$ »;
- 4) « $2 \times 2 = 4$ и $3 \times 3 = 9$ ».

Перейдем теперь от записи высказываний на естественном языке к их записи на формальном языке алгебры логики. Операцию логического умножения (конъюнкции) принято обозначать значком «&» (амперсанд). Операция логического умножения, аргументами которой являются логические переменные A и B , записывается следующей формулой:

$$A \ \& \ B. \tag{3.1}$$

Значение логической операции логического умножения задается с помощью таблицы истинности. Таблица истинности показывает, какие значения дает логическая операция при всех возможных наборах ее аргументов (табл. 3.1). Результатом операции логического умножения является «истина» (1) тогда и только тогда, когда оба аргумента принимают значения «истина» (1).

Таблица 3.1. Таблица истинности конъюнкции (логического умножения)

A	B	$A \ \& \ B$
0	0	0
0	1	0
1	0	0
1	1	1

По таблице истинности легко определить истинность составного высказывания, образованного с помощью операции логического умножения. Рассмотрим, например, составное высказывание « $2 \times 2 = 4$ и $3 \times 3 = 10$ ». Первое простое высказывание истинно ($A = 1$), а второе высказывание ложно ($B = 0$), с использованием таблицы истинности логического умножения определяем, что данное составное высказывание ложно.

Логическое сложение (дизъюнкция). Объединение двух (или нескольких) высказываний с помощью союза «или» называется операцией логического сложения или дизъюнкцией.



Составное высказывание, образованное в результате логического сложения (дизъюнкции), истинно тогда и только тогда, когда истинно хотя бы одно из входящих в него простых высказываний.

Так, из приведенных ниже четырех составных высказываний, образованных с помощью операции логического сложения, ложно только первое, так как в последних трех составных высказываниях хотя бы одно из простых высказываний истинно:

- 1) « $2 \times 2 = 5$ или $3 \times 3 = 10$ »;
- 2) « $2 \times 2 = 5$ или $3 \times 3 = 9$ »;
- 3) « $2 \times 2 = 4$ или $3 \times 3 = 10$ »;
- 4) « $2 \times 2 = 4$ или $3 \times 3 = 9$ ».

Запишем теперь операцию логического сложения на формальном языке алгебры логики. Операцию логического сложения (дизъюнкцию) принято обозначать значком « \vee ». Операция логического сложения, аргументами которой являются логические переменные A и B , записывается следующей формулой:

$$A \vee B. \quad (3.2)$$

Значение логической операции логического сложения задается с помощью таблицы истинности (табл. 3.2). Результатом операции логического сложения является «ложь» (0) тогда и только тогда, когда оба аргумента принимают значения «ложь» (0).

Таблица 3.2. Таблица истинности дизъюнкции (логического сложения)

A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

По таблице истинности легко определить истинность составного высказывания, образованного с помощью операции логического сложения. Рассмотрим, например, составное высказывание « $2 \times 2 = 4$ или $3 \times 3 = 10$ ». Первое простое высказывание истинно ($A = 1$), а второе высказывание ложно ($B = 0$), с помощью таблицы истинности логического сложения определяем, что данное составное высказывание истинно.

Логическое отрицание (инверсия). Присоединение частицы «не» к высказыванию называется операцией логического отрицания или инверсией.



Логическое отрицание (инверсия) получает из истинного высказывания ложное и, наоборот, из ложного — истинное.

Высказывание «Два умножить на два равно четырем» истинно, а высказывание, образованное с помощью операции логического отрицания, «Два умножить на два не равно четырем» — ложно.

Запишем теперь операцию логического отрицания на формальном языке алгебры логики. Операцию логического отрицания (инверсию) над логическим высказыванием A принято обозначать \bar{A} . Операция логического отрицания, аргументом которой является логическая переменная A , записывается следующей формулой:

$$\bar{A} \tag{3.3}$$

Значение логической операции отрицания задается с помощью таблицы истинности (табл. 3.3). Результатом операции логического отрицания является «истина» (1), когда аргумент принимает значение «ложь» (0), и значение «ложь» (0), когда аргумент принимает значение «истина» (1).

Таблица 3.3. Таблица истинности инверсии (логического отрицания)

A	\bar{A}
0	1
1	0

Истинность высказывания, образованного с помощью операции логического отрицания, можно легко определить с помощью таблицы истинности. Например, высказывание «Два

умножить на два не равно четырем» ложно ($A = 0$), а полученное из него в результате логического отрицания высказывание «Два умножить на два равно четырем» истинно (1).

Практическое задание «Таблицы истинности». Получить таблицы истинности операций логического умножения, логического сложения и логического отрицания с использованием электронных таблиц.



Получение таблиц истинности операций логического умножения, логического сложения и логического отрицания с использованием электронных таблиц

1. На листе *Лист1* создать заготовку таблиц истинности базовых логических операций. Создать заголовки и ввести в столбцы А и В, Е и F, I значения логических аргументов, а в столбцы С, G, J соответствующие логические операции.

Электронные таблицы обладают встроенными логическими функциями. Функция логического умножения И(логическое значение1;логическое значение2;...) дает значение TRUE (1) тогда и только тогда, когда все логические аргументы имеют значение TRUE (1).

Функция логического сложения ИЛИ(логическое значение1;логическое значение2;...) дает значение TRUE (1) тогда и только тогда, когда хотя бы один логический аргумент имеет значение TRUE (1).

Функция логического отрицания НЕ(логическое значение) дает значение TRUE (1), когда логический аргумент имеет значение FALSE (0) и, наоборот, значение FALSE (0), когда логический аргумент имеет значение TRUE (1).



Ввод логических функций с использованием электронных таблиц Microsoft Excel

2. Для ввода логических функций воспользоваться командой [Вставка- Функция...]. В появившемся диалоговом окне *Мастер функций* (рис. 3.2) в раскрывающемся списке *Категории* выбрать *Логические*, а в окне *Выберите функцию:* — функцию. Щелкнуть по кнопке *Далее*.
3. В диалоговом окне *Аргументы функции* (рис. 3.3) в текстовых полях *Логическое значение 1* и *Логическое значение 2* выбрать имена ячеек, в которых хранятся аргументы логической функции. Щелкнуть по кнопке *OK*.

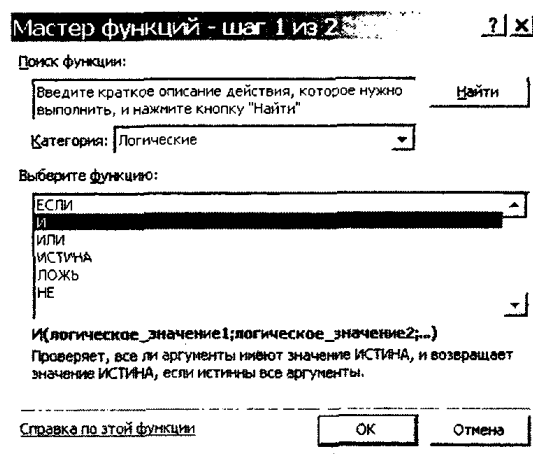


Рис. 3.2. Выбор логической функции

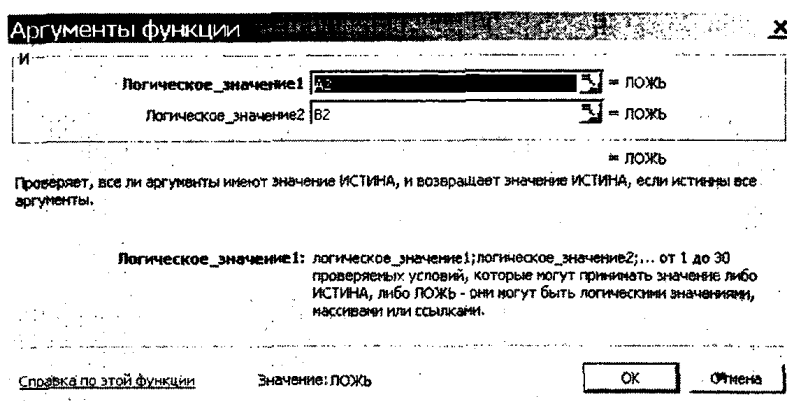


Рис. 3.3. Ввод аргументов логической функции

В электронных таблицах OpenOffice.org Calc функция логического умножения обозначается AND(), функция логического сложения OR() и функция логического отрицания NOT().



Ввод логических функций с использованием электронных таблиц OpenOffice.org Calc

2. Для ввода логических функций воспользоваться командой [Вставить Функция...].

В появившемся диалоговом окне *Мастер:Функции* в раскрывающемся списке *Категория* выбрать *Логические*, а в окне *Функция* — нужную функцию (рис. 3.4). Щелкнуть по кнопке *Далее*.

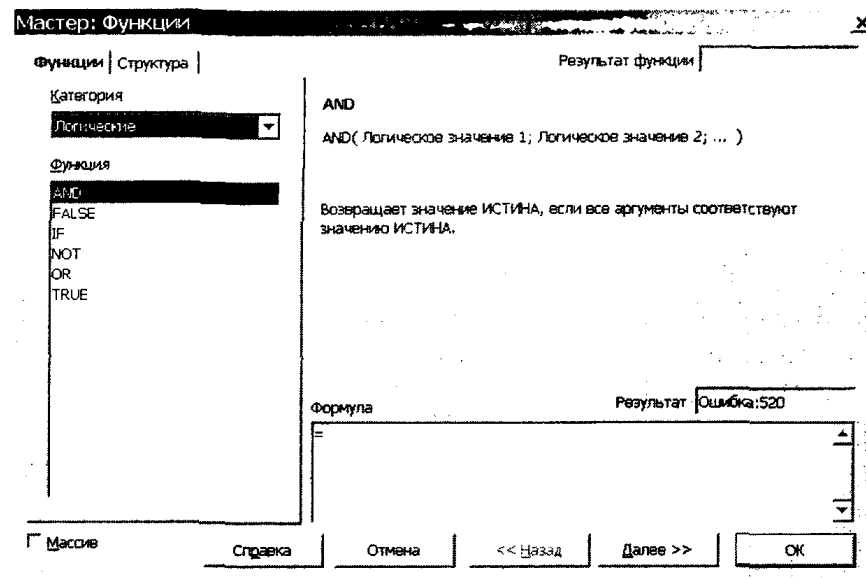


Рис. 3.4. Выбор логической функции

3. В диалоговом окне *Мастер:Функции* в текстовых полях *Логическое значение 1* и *Логическое значение 2* выбрать имена ячеек, в которых хранятся аргументы логической функции (рис. 3.5). Щелкнуть по кнопке *OK*.

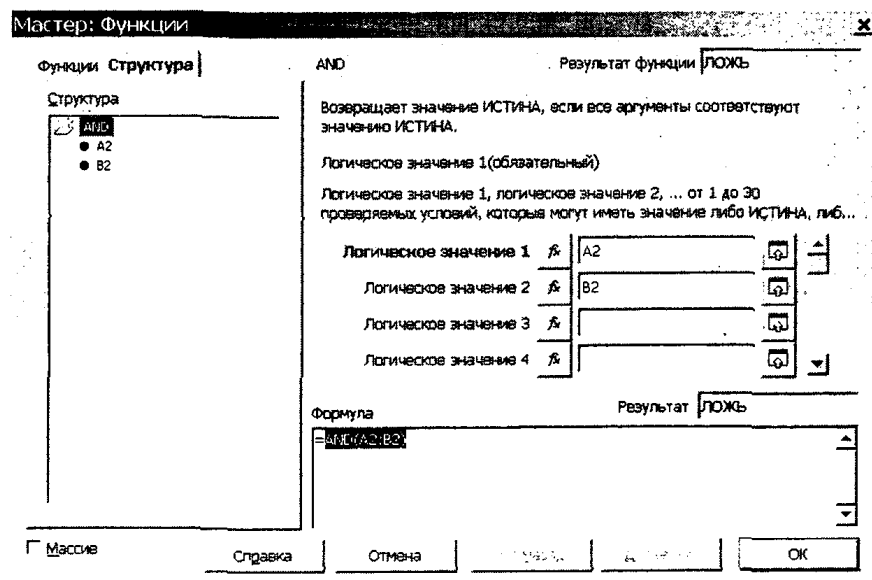


Рис. 3.5. Ввод аргументов логической функции



Сохранение файла электронных таблиц

- После ввода аргументов функций и их формул на листе появятся таблицы истинности трех базовых логических функций (рис. 3.6).

The screenshot shows a spreadsheet window titled 'log.xls - OpenOffice.org Calc'. The active cell C2 contains the formula '=AND(A2;B2)'. The spreadsheet displays truth tables for three logical functions: Конъюнкция (AND), Дизъюнкция (OR), and Инверсия (NOT). The columns are labeled A, B, and the function name. The rows show combinations of 0 and 1 for A and B, and the resulting truth value for the function.

	A	B	Конъюнкция	Дизъюнкция	Инверсия
1	0	0	ЛОЖЬ	0	ЛОЖЬ
3	0	1	ЛОЖЬ	0	ИСТИНА
4	1	0	ЛОЖЬ	1	ЛОЖЬ
5	1	1	ИСТИНА	1	ИСТИНА

Рис. 3.6. Таблицы истинности базовых логических функций в электронных таблицах

- Переименовать лист *Лист1* в *Логические операции*. Сохранить в электронных таблицах Microsoft Excel или OpenOffice.org Calc файл log.xls в универсальном формате. Сохранить в электронных таблицах OpenOffice.org Calc файл log.ods в собственном формате.

Контрольные вопросы

- Перечислите связки в составных высказываниях, знаки логических операций и функции в электронных таблицах, реализующие логические операции умножения, сложения и отрицания.




Задача

- Определить, при каких значениях числа x предикат первого порядка не $((x > 8) \text{ или } (x < -3))$ примет значение:
 - ложь;
 - истина.

- 3.5. Изобразить в декартовой прямоугольной системе координат область, в которой и только в которой истинен предикат второго порядка $(|x| \leq 1)$ и $(|y| \leq 1)$.

Компьютерный практикум



Windows-CD 

- 3.1. В электронных таблицах выполнить практическое задание «Таблицы истинности».



\\informatika10\logic\log.*

3.2.2. Логические выражения

Логические выражения. Каждое составное высказывание можно выразить в виде формулы (логического выражения), в которую войдут логические переменные, обозначающие высказывания, и знаки логических операций.

Для записи составных высказываний в виде логических выражений на формальном языке (языке алгебры логики) в составном высказывании нужно выделить простые высказывания и логические связи между ними.

Запишем в форме логического выражения составное высказывание « $(2 \times 2 = 5)$ или $(2 \times 2 = 4)$ и $(2 \times 2 \neq 5)$ или $(2 \times 2 \neq 4)$ ». Проанализируем составное высказывание. Оно содержит два простых высказывания:

$$A = \{2 \times 2 = 5\} \text{ — ложно (0);}$$

$$B = \{2 \times 2 = 4\} \text{ — истинно (1).}$$

Теперь необходимо записать высказывание в форме логического выражения с учетом последовательности выполнения логических операций.



При выполнении логических операций определен следующий порядок их выполнения: инверсия, конъюнкция, дизъюнкция.

Для изменения указанного порядка могут использоваться скобки:

$$(A \vee B) \& (\bar{A} \vee \bar{B}).$$

Истинность или ложность составных высказываний можно определять формально, руководствуясь законами алгебры логики, не обращаясь к смысловому содержанию высказываний.

Практическое задание «Определение истинности логического выражения». Определить истинность логического выражения $(A \vee B) \& (A \vee \bar{B})$ с использованием таблиц истинности базовых логических операций и с использованием программного калькулятора.

Подставим в логическое выражение значения логических переменных и, используя таблицы истинности базовых логических операций, получим значение логического выражения:

$$(A \vee B) \& (A \vee \bar{B}) = (0 \vee 1) \& (1 \vee 0) = 1 \& 1 = 1.$$



Определение истинности логического выражения с использованием компьютерного калькулятора

1. Запустить компьютерный калькулятор NumLock Calculator.
2. Установить удобный вид калькулятора командой [Вид калькулятора-Программистский].
3. Ввести логическое выражение, подставив в него значения логических переменных.
4. Нажать на клавиатуре клавишу {Enter}, получим значение логического выражения (рис. 3.7), которое совпадает со значением, вычисленным с использованием таблиц истинности.

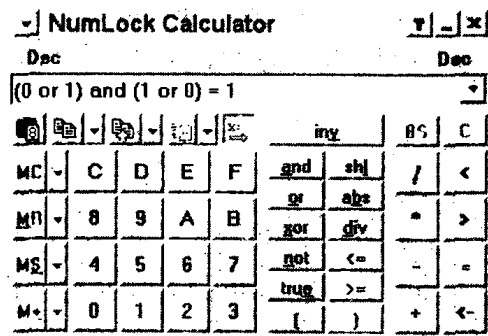


Рис. 3.7. Определение истинности логического выражения с использованием калькулятора

Построение таблиц истинности логических выражений. Для каждого логического выражения можно построить таблицу истинности, которая определяет его истинность или ложность при всех возможных комбинациях исходных значений логических переменных.

При построении таблиц истинности целесообразно руководствоваться определенной последовательностью действий.

Во-первых, необходимо определить количество строк в таблице истинности, которое равно количеству возможных комбинаций значений логических переменных, входящих в логическое выражение. Если количество логических переменных n , то:

$$\text{Количество строк} = 2^n.$$

В нашем случае логическое выражение $(A \vee B) \& (\bar{A} \vee \bar{B})$ имеет две переменные и, следовательно, количество строк в таблице истинности должно быть равно:

$$\text{Количество строк} = 2^n = 2^2 = 4.$$

Во-вторых, необходимо определить количество столбцов в таблице истинности, которое равно количеству логических переменных плюс количество логических операций (при желании можно сократить количество столбцов, объединив несколько логических операций в одном столбце). В нашем случае количество переменных равно двум, а количество логических операций равно пяти, т. е. количество столбцов таблицы истинности равно:

$$\text{Количество столбцов} = 2 + 5 = 7.$$

В-третьих, необходимо построить таблицу истинности с указанным количеством строк и столбцов, обозначить столбцы и внести все возможные наборы значений логических переменных. Наборы входных переменных, во избежание ошибок, рекомендуют вводить следующим образом:

- 1) разделить столбец значений первой переменной пополам и заполнить верхнюю часть колонки нулями, а нижнюю — единицами;
- 2) разделить столбец значений второй переменной на четыре части и заполнить четверти чередующимися группами нулей и единиц, начиная с группы нулей;
- 3) продолжать деление столбцов значений последующих переменных на 8, 16 и т. д. частей и заполнение их группами нулей или единиц до тех пор, пока группа нулей (единиц) не будет состоять из одного символа.

В-четвертых, необходимо заполнить таблицу истинности по столбцам, выполняя базовые логические операции в необходимой последовательности и в соответствии с их таблицами истинности. Теперь мы можем определить значение логической функции для любого набора значений логических переменных.

Построим таблицу истинности для рассмотренного логического выражения (табл. 3.4).

Таблица 3.4. Таблица истинности логического выражения $(A \vee B) \& (\bar{A} \vee \bar{B})$

A	B	$A \vee B$	\bar{A}	\bar{B}	$\bar{A} \vee \bar{B}$	$(A \vee B) \& (\bar{A} \vee \bar{B})$
0	0	0	1	1	1	0
0	1	1	1	0	1	1
1	0	1	0	1	1	1
1	1	1	0	0	0	0

Задание «Таблица истинности логического выражения».
 Для логического выражения $A \& (B \vee \bar{B} \& \bar{C})$ построить таблицу истинности.

Количество логических переменных равно трем, следовательно, количество строк в таблице истинности должно быть:

$$\text{Количество строк} = 2^n = 2^3 = 8.$$

Количество логических операций равно пяти, следовательно, количество столбцов в таблице истинности должно быть:

$$\text{Количество столбцов} = 3 + 5 = 8.$$

Построим таблицу истинности (табл. 3.5).

Таблица 3.5. Таблица истинности логического выражения $A \& (B \vee \bar{B} \& \bar{C})$

A	B	C	\bar{B}	\bar{C}	$\bar{B} \& \bar{C}$	$B \vee (\bar{B} \& \bar{C})$	$A \& (B \vee \bar{B} \& \bar{C})$
0	0	0	1	1	1	1	0
0	0	1	1	0	0	0	0
0	1	0	0	1	0	1	0
0	1	1	0	0	0	1	0
1	0	0	1	1	1	1	1
1	0	1	1	0	0	0	0
1	1	0	0	1	0	1	1
1	1	1	0	0	0	1	1

Равносильные логические выражения. Логические выражения, у которых таблицы истинности совпадают, называются **равносильными**. Для обозначения равносильных логических выражений используется знак «=».

Задание «Равносильность логических выражений». Доказать, что логические выражения $A \& \bar{B}$ и $\overline{A \vee B}$ равносильны.

Построим сначала таблицу истинности для логического выражения $A \& \bar{B}$ (табл. 3.6).

Таблица 3.6. Таблица истинности логического выражения $\bar{A} \& \bar{B}$

A	B	\bar{A}	\bar{B}	$\bar{A} \& \bar{B}$
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0

Построим теперь таблицу истинности для логического выражения $\overline{A \vee B}$ (табл. 3.7).

Таблица 3.7. Таблица истинности логического выражения $\overline{A \vee B}$

A	B	$A \vee B$	$\overline{A \vee B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

Таблицы истинности совпадают, следовательно, логические выражения равносильны:

$$\bar{A} \& \bar{B} = \overline{A \vee B}.$$

Контрольные вопросы

1. Что содержат таблицы истинности и каков порядок их построения?
2. Какие логические выражения называются равносильными?



- 3.6. Записать составное высказывание « $(2 \times 2 = 4$ и $3 \times 3 = 9)$ » или « $(2 \times 2 \neq 4$ и $3 \times 3 \neq 9)$ » в форме логического выражения. Определить его истинность.
- 3.7. Доказать, используя таблицы истинности, что логические выражения $\overline{A \vee B}$ и $A \& B$ равносильны.

Легко заметить, что логическая функция F_2 является функцией логического умножения, F_8 — функцией логического сложения, F_{13} — функцией логического отрицания для аргумента A и F_{11} — функцией логического отрицания для аргумента B .

В обыденной и научной речи кроме базовых логических связок «и», «или», «не» используются и некоторые другие: «если... то ...», «тогда... и только тогда, когда ...» и др. Некоторые из них имеют свое название и свой символ, и им соответствуют определенные логические функции.

Логическое следование (импликация). Логическое следование (импликация) образуется соединением двух высказываний в одно с помощью оборота речи «если ..., то ...».

Логическая операция импликации «если A , то B » обозначается $A \rightarrow B$ и выражается с помощью логической функции F_{14} , которая задается соответствующей таблицей истинности (табл. 3.9).

Таблица 3.9. Таблица истинности логической функции «импликация»

A	B	$F_{14} = A \rightarrow B$
0	0	1
0	1	1
1	0	0
1	1	1



Составное высказывание, образованное с помощью операции логического следования (импликации), ложно тогда и только тогда, когда из истинной посылки (первого высказывания) следует ложный вывод (второе высказывание).

Например, высказывание «Если число делится на 10, то оно делится на 5» истинно, так как истинны и первое высказывание (посылка), и второе высказывание (вывод).

Высказывание «Если число делится на 10, то оно делится на 3» ложно, так как из истинной посылки делается ложный вывод.

Однако операция логического следования несколько отличается от обычного понимания слова «следует». Если первое высказывание (посылка) ложно, то вне зависимости от истинности или ложности второго высказывания (вывода) состав-

ное высказывание истинно. Это можно понимать таким образом, что из неверной посылки может следовать что угодно.

В алгебре логики все логические функции могут быть выражены путем логических преобразований через три базовые: логическое умножение, логическое сложение и логическое отрицание.

Практическое задание «Функция импликации». Выразить функцию импликации F_{14} через базовые логические функции. Доказать методом сравнения таблиц истинности, что функция импликации равносильна логическому выражению $A \vee B$. Построить таблицу истинности функции импликации в электронных таблицах.

Построим таблицу истинности логического выражения $\overline{A} \vee B$ (табл. 3.10).

Таблица 3.10. Таблица истинности логического выражения $\overline{A} \vee B$

A	B	\overline{A}	$\overline{A} \vee B$
0	0	1	1
0	1	1	1
1	0	0	0
1	1	0	1

Таблицы истинности функции импликации F_{14} (см. табл. 3.9) и логического выражения $\overline{A} \vee B$ (см. табл. 3.10) совпадают, что и требовалось доказать.



Получение таблицы истинности функции импликации

1. В электронных таблицах создать заготовку таблицы истинности функции импликации: создать заголовки и ввести в столбцы A и B значения логических аргументов.
2. В столбце выразить логическую функцию $F_{14} = \overline{A} \vee B$ через логические функции электронных таблиц Microsoft Excel =ИЛИ(НЕ(A2);B2) (в OpenOffice.org Calc =OR(NOT(A2);B2).

Логическое равенство (эквивалентность). Логическое равенство (эквивалентность) образуется соединением двух высказываний в одно с помощью оборота речи «... тогда и только тогда, когда ...».

Логическая операция эквивалентности «A эквивалентно B» обозначается $A \sim B$ и выражается с помощью логической функции F_{10} , которая задается соответствующей таблицей истинности (табл. 3.11).

Таблица 3.11. Таблица истинности логической функции эквивалентности

A	B	$F_{10} = A \sim B$
0	0	1
0	1	0
1	0	0
1	1	1



Составное высказывание, образованное с помощью логической операции эквивалентности, истинно тогда и только тогда, когда оба высказывания одновременно либо ложны, либо истинны.

Рассмотрим, например, два высказывания: A = «Компьютер может производить вычисления» и B = «Компьютер включен». Составное высказывание, полученное с помощью операции эквивалентности, истинно, когда оба высказывания либо истинны, либо ложны:

«Компьютер может производить вычисления тогда и только тогда, когда компьютер включен».

«Компьютер не может производить вычисления тогда и только тогда, когда компьютер не включен».

Составное высказывание, полученное с помощью операции эквивалентности, ложно, когда одно высказывание истинно, а другое ложно:

«Компьютер может производить вычисления тогда и только тогда, когда компьютер не включен».

«Компьютер не может производить вычисления тогда и только тогда, когда компьютер включен».

Практическое задание «Функция эквивалентности». Выразить функцию эквивалентности F_{10} через базовые логические функции. Доказать методом сравнения таблиц истинности, что функция эквивалентности равносильна логическому выражению $(\bar{A} \& \bar{B}) \vee (A \& B)$. Построить таблицу истинности функция эквивалентности в электронных таблицах.

Построим таблицу истинности логического выражения $(\bar{A} \& \bar{B}) \vee (A \& B)$ (табл. 3.12).

Таблица 3.12. Таблица истинности логического выражения $(\bar{A} \& \bar{B}) \vee (A \& B)$

A	B	\bar{A}	\bar{B}	$\bar{A} \& \bar{B}$	$A \& B$	$(\bar{A} \& \bar{B}) \vee (A \& B)$
0	0	1	1	1	0	1
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	1	0	0	0	1	1

Таблицы истинности функции эквивалентности F_{10} (см. табл. 3.11) и логического выражения $(A \& B) \vee (A \& \bar{B})$ (см. табл. 3.12) совпадают, что и требовалось доказать.



Получение таблицы истинности функции эквивалентности

1. В электронных таблицах создать заготовку таблиц истинности функции эквивалентности: создать заголовки и ввести в столбцы A и B значения логических аргументов.
2. В столбце выразить логическую функцию $F_{10} = (A \& \bar{B}) \vee (A \& B)$ через логические функции электронных таблиц Microsoft Excel =ИЛИ(И(НЕ(A2);НЕ(B2)); И(A2;B2)) (в OpenOffice.org Calc =OR(AND(NOT(A2);NOT(B2); AND(A2;B2))).

Контрольные вопросы

1. Какое количество логических функций двух аргументов существует и почему?
2. Какие названия логических функций двух аргументов вы знаете?
3. Какое существует количество логических функций трех аргументов?

Задания

3.8. Доказать с использованием таблиц истинности правильность выражения логических функций через базовые логические функции (конъюнкцию, дизъюнкцию и отрицание):

$$\begin{array}{llll}
 F_1(A,B)=A\&\bar{A}; & F_5(A,B)=\bar{A}\&B; & F_9(A,B)=\bar{A}\vee\bar{B}; & F_{13}(A,B)=\bar{A}; \\
 F_2(A,B)=A\&B; & F_6(A,B)=B; & F_{10}(A,B)=(\bar{A}\&\bar{B})\vee(A\&B); & F_{14}(A,B)=A\vee\bar{B}; \\
 F_3(A,B)=A\&\bar{B}; & F_7(A,B)=(\bar{A}\&B)\vee(A\&\bar{B}); & F_{11}(A,B)=\bar{B}; & F_{15}(A,B)=A\&B; \\
 F_4(A,B)=A; & F_8(A,B)=A\sim B; & F_{12}(A,B)=\bar{A}\vee B; & F_{16}(A,B)=A\sim\bar{A}.
 \end{array}$$

Компьютерный практикум



Windows-CD 

3.5. В электронных таблицах построить таблицы истинности всех логических функций двух переменных.



\\informatika10\logic\log.*

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1																		
2	0	0	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь
3	0	1	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь
4	1	0	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь
5	1	1	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь	ложь

Рис. 3.8. Таблицы истинности логических функций

3.2.4. Логические законы и правила преобразования логических выражений

Законы логики отражают наиболее важные закономерности логического мышления. В алгебре логики законы логики записываются в виде формул, которые позволяют проводить равносильные преобразования логических выражений.

Закон непротиворечия. Высказывание не может быть одновременно истинным и ложным. Если высказывание A истинно, то его отрицание \bar{A} должно быть ложным. Следовательно, логическое произведение высказывания и его отрицания должно быть ложно:



$$A \& \bar{A} = 0$$

Закон исключенного третьего. Высказывание может быть либо истинным, либо ложным, третьего не дано. Это означает, что результат логического сложения высказывания и его отрицания всегда принимает значение истина:



$$A \vee \bar{A} = 1$$

Закон двойного отрицания. Если дважды отрицать некоторое высказывание, то в результате мы получим исходное высказывание:



$$\bar{\bar{A}} = A$$

Законы де Моргана (законы общей инверсии для логического сложения и для логического умножения). Общая инверсия двух логических слагаемых равносильна логическому умножению инвертированных переменных:



$$\overline{A \vee B} = \overline{A} \& \overline{B}$$

Общая инверсия двух логических сомножителей равносильна логическому сложению инвертированных переменных:



$$\overline{A \& B} = \overline{A} \vee \overline{B}$$

Кроме логических законов важное значение для выполнения преобразований логических выражений имеют правила алгебраических преобразований. Многие из них имеют аналоги в знакомой вам алгебре переменных и функций.

Правило коммутативности. В алгебре переменных и функций слагаемые и множители можно менять местами. В алгебре логики можно менять местами логические переменные при операциях логического умножения и логического сложения:

Логическое умножение	Логическое сложение
$A \& B = B \& A$	$A \vee B = B \vee A$

Правило ассоциативности. Если в логическом выражении используется только операция логического умножения или только операция логического сложения, то можно пренебрегать скобками или произвольно их расставлять:

Логическое умножение	Логическое сложение
$(A \& B) \& C = A \& (B \& C)$	$(A \vee B) \vee C = A \vee (B \vee C)$

Правило дистрибутивности. В отличие от алгебры переменных и функций, где за скобки можно выносить только общие множители, в алгебре логики за скобки можно выносить как общие множители, так и общие слагаемые:

Дистрибутивность умножения относительно сложения	Дистрибутивность сложения относительно умножения
$ab + ac = a(b+c)$ – в алгебре $(A \& B) \vee (A \& C) = A \& (B \vee C)$	$(A \vee B) \& (A \vee C) = A \vee (B \& C)$

Правила равносильности. Это правила отсутствия показателей степени у результатов логического сложения и умножения переменных.

Для логического сложения:

$$A \vee A = A.$$

Для логического умножения:

$$A \& A = A.$$

Правила исключения констант

Для логического сложения:

$$A \vee 1 = 1, \quad A \vee 0 = A.$$

Для логического умножения:

$$A \& 1 = A, \quad A \& 0 = 0.$$

Рассмотрим примеры применения законов логики и правил алгебры логики для преобразования логических выражений.

Задание «Преобразование логического выражения». Упростить логическое выражение $(A \& B) \vee (A \& \bar{B})$.

Воспользуемся правилом дистрибутивности и вынесем за скобки A :

$$(A \& B) \vee (A \& \bar{B}) = A \& (B \vee \bar{B}).$$

По закону исключенного третьего $B \vee \bar{B} = 1$, следовательно:

$$A \& (B \vee \bar{B}) = A \& 1.$$

По правилу исключения констант:

$$A \& 1 = A.$$

Задание «Решение логического уравнения». Найти значение логической переменной X из логического уравнения $\overline{X \vee B \vee X \vee A} = B$.

В соответствии с приоритетами выполнения логических операций расставим скобки в левой части логического уравнения:

$$\overline{(X \vee B) \vee (X \vee A)} = B.$$

Для преобразования левой части уравнения воспользуемся для первых и вторых скобок законом де Моргана для логического сложения и для вторых скобок законом двойного отрицания:

$$(\bar{X} \& \bar{A}) \vee (\bar{X} \& A) = B.$$

Согласно распределительному закону для логического сложения, можно вынести \bar{X} за скобки:

$$\bar{X} \& (\bar{A} \vee A) = B.$$

Согласно закону исключенного третьего, $(\bar{A} \vee A) = 1$ и уравнение примет вид:

$$\bar{X} \& 1 = B.$$

Согласно правилу исключения констант, $\bar{X} \& 1 = \bar{X}$ и уравнение примет вид:

$$\bar{X} = B.$$

Инвертируем левую и правую части уравнения и получим решение уравнения:

$$X = \bar{B}.$$

Контрольные вопросы

1. Какие правила преобразования логических выражений справедливы и при преобразовании алгебраических выражений?

З а д а н и я

- 3.9. Доказать справедливость первого $\overline{A \vee B} = \bar{A} \& \bar{B}$ и второго $A \& B = \overline{\bar{A} \vee \bar{B}}$ законов де Моргана, используя таблицы истинности.
- 3.10. Упростить логическое выражение: $(A \vee B) \& (A \vee \bar{B})$.
- 3.11. Решить логическое уравнение: $\overline{X \& B} \& \overline{X \& \bar{B}} = A$.

3.2.5. Решение логических задач

Логические задачи обычно формулируются на естественном языке. В первую очередь их необходимо формализовать, т. е. записать на языке алгебры логики. Полученные логические выражения необходимо упростить и проанализировать. Для этого иногда бывает необходимо построить таблицу истинности полученного логического выражения.

Задание «Логическая задача». В школе в каждой из двух аудиторий может находиться либо кабинет информатики, либо кабинет физики. На аудиториях повесили шуточные таблички, про которые известно, что они либо обе истинны, либо обе ложны. На первой аудитории повесили табличку «По крайней мере, в одной из этих аудиторий размещается кабинет информатики», а на второй аудитории — табличку с

надписью «Кабинет физики находится в другой аудитории». Определите, какой кабинет размещается в каждой из аудиторий.

Переведем условие задачи на язык алгебры логики. Так как в каждой из аудиторий может находиться кабинет информатики, то пусть:

A — «В первой аудитории находится кабинет информатики».

B — «Во второй аудитории находится кабинет информатики».

Поскольку в каждой аудитории обязательно размещается какой-либо из этих двух кабинетов, отрицания этих высказываний будут соответствовать:

\bar{A} — «В первой аудитории находится кабинет физики».

\bar{B} — «Во второй аудитории находится кабинет физики».

Высказывание, содержащееся на табличке первой аудитории, соответствует логическому выражению:

$$X = A \vee B.$$

Высказывание, содержащееся на табличке второй аудитории, соответствует логическому выражению:

$$Y = \bar{A}.$$

Содержащееся в условии задачи утверждение о том, что надписи на табличках либо одновременно истинные, либо одновременно ложные, соответствует истинности функции эквивалентности:

$$(X \rightarrow Y) = 1.$$

Выразим функцию эквивалентности через базовые логические функции и получим:

$$(X \& Y) \vee (\bar{X} \& \bar{Y}) = 1.$$

Подставим вместо X и Y соответствующие логические выражения:

$$((A \vee B) \& \bar{A}) \vee (\overline{(A \vee B) \& \bar{A}}) = 1.$$

Упростим сначала первое слагаемое. В соответствии с правилом дистрибутивности умножения относительно сложения:

$$((A \vee B) \& \bar{A}) = (A \& \bar{A}) \vee (B \& \bar{A}).$$

В соответствии с законом непротиворечия:

$$(A \& \bar{A}) \vee (B \& \bar{A}) = 0 \vee (B \& \bar{A}).$$

В соответствии с правилом исключения констант:

$$0 \vee (B \& \bar{A}) = (B \& \bar{A}).$$

Упростим теперь второе слагаемое. В соответствии с первым законом де Моргана и законом двойного отрицания:

$$\overline{((A \vee B) \& A)} = \overline{(\overline{A} \& \overline{B} \& A)} = (\overline{A} \& A \& \overline{B}).$$

В соответствии с законом непротиворечия:

$$(\overline{A} \& A \& \overline{B}) = (0 \& \overline{B}) = 0.$$

В результате преобразований первого и второго слагаемых получаем:

$$(B \& \overline{A}) \vee 0 = 1.$$

В соответствии с правилом исключения констант:

$$B \& \overline{A} = 1.$$

Полученное логическое выражение оказалось простым, и поэтому его можно проанализировать без построения таблицы истинности. Для того чтобы выполнялось равенство, обе логические переменные должны быть равны 1, а соответствующие им высказывания истинны:

B — «Во второй аудитории находится кабинет информатики».

\overline{A} — «В первой аудитории находится кабинет физики».

Таким образом, логическая задача решена: в первой аудитории находится кабинет физики, а во второй — кабинет информатики.

Компьютерный практикум



Windows-CD 

3.6. Решить приведенную в параграфе логическую задачу методом построения в электронных таблицах таблицы истинности логического уравнения.



\informatika10\logic\log.*

Microsoft Excel - log.xls										
Файл Правка Вид Вставка Формат Сервис Данные Окно Справка										
=ИЛИ(E2,H2)										
	A	B	C	D	E	F	G	H		
1	A	B	\overline{A}	$A \vee B$	$(A \vee B) \& A$	$A \vee \overline{B}$	\overline{A}	$(A \vee B) \& \overline{A}$	$((A \vee B) \& \overline{A}) \vee ((A \vee B) \& A)$	
2	0	0	ИСТИНА	ЛОЖЬ	ЛОЖЬ	ИСТИНА	ЛОЖЬ	ЛОЖЬ	ЛОЖЬ	
3	0	1	ИСТИНА	ИСТИНА	ИСТИНА	ЛОЖЬ	ЛОЖЬ	ЛОЖЬ	ИСТИНА	
4	1	0	ЛОЖЬ	ИСТИНА	ЛОЖЬ	ЛОЖЬ	ИСТИНА	ЛОЖЬ	ЛОЖЬ	
5	1	1	ЛОЖЬ	ИСТИНА	ЛОЖЬ	ЛОЖЬ	ИСТИНА	ЛОЖЬ	ЛОЖЬ	

3.3. Логические основы устройства компьютера

3.3.1. Базовые логические элементы

Дискретный преобразователь, который после обработки входных двоичных сигналов выдает на выходе сигнал, являющийся значением одной из логических операций, называется **логическим элементом**. Базовые логические элементы реализуют три базовые логические операции:

- логический элемент «И» (конъюнктор) — логическое умножение;
- логический элемент «ИЛИ» (дизъюнктор) — логическое сложение;
- логический элемент «НЕ» (инвертор) — инверсию.

Любая логическая операция может быть представлена в виде комбинации трех базовых, поэтому любые устройства компьютера, производящие обработку или хранение информации (сумматоры в процессоре, ячейки памяти в оперативной памяти и др.), могут быть собраны из базовых логических элементов, как из кирпичиков.

Логические элементы компьютера оперируют с сигналами, представляющими собой электрические импульсы. Есть импульс — логическое значение сигнала 1, нет импульса — значение 0. На вход логического элемента поступают сигналы-аргументы, на выходе появляется сигнал-функция.

Преобразование сигнала логическим элементом задается таблицей состояния, которая фактически является таблицей истинности, соответствующей логической функции.

Логический элемент «И». На входы *A* и *B* логического элемента последовательно подаются четыре пары сигналов, а на выходе получается последовательность из четырех сигналов, значения которых определяются в соответствии с таблицей истинности операции логического умножения. На рис. 3.8 изображена логическая схема элемента «И».

Простейшей моделью логического элемента «И» может быть электрическая схема, состоящая из источника тока, лампочки и двух выключателей. Данную схему можно со-

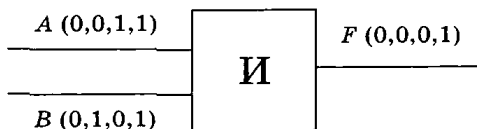


Рис. 3.8. Логический элемент «И»

брать из реальных электрических элементов или с использованием компьютерного конструктора «Начала электроники» (рис. 3.9).

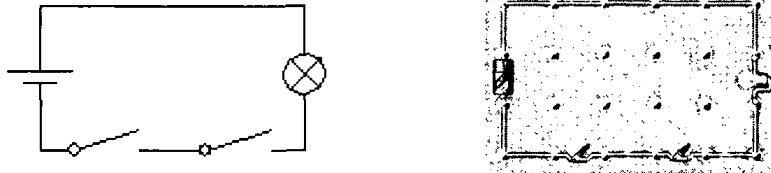


Рис. 3.9. Электрическая схема модели логического элемента «И» и ее реализация в компьютерном конструкторе «Начала электроники»

Из схемы видно, что если оба выключателя замкнуты (на обоих входах 1), по цепи идет ток и лампочка горит (на выходе 1). Если хотя бы один выключатель разомкнут (на одном из входов 0), то тока нет и лампочка не горит (на выходе 0).

Логический элемент «ИЛИ» (рис. 3.10). На входы *A* и *B* логического элемента последовательно подаются четыре пары сигналов, а на выходе получается последовательность из четырех сигналов, значения которых определяются в соответствии с таблицей истинности операции логического сложения.

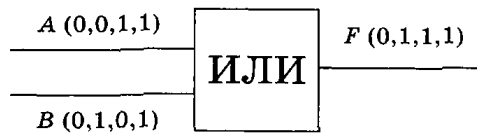


Рис. 3.10. Логический элемент «ИЛИ»

Простейшей моделью логического элемента «ИЛИ» может быть электрическая схема, которую можно собрать из реальных электрических элементов или с использованием компьютерного конструктора «Начала электроники» (рис. 3.11).

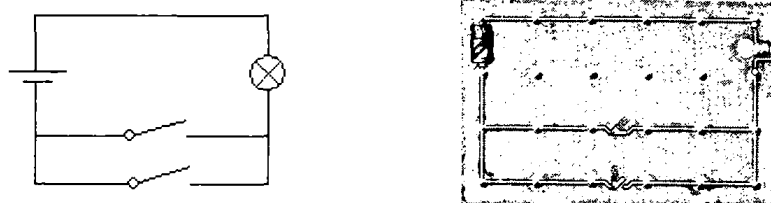


Рис. 3.11. Электрическая схема модели логического элемента «ИЛИ» и ее реализация в компьютерном конструкторе «Начала электроники»

Из схемы видно, что, если хотя бы один выключатель замкнут (на входе 1), по цепи идет ток и лампочка горит (на выходе 1).

Логический элемент «НЕ» (рис. 3.12). На вход A логического элемента последовательно подаются два сигнала, на выходе получается последовательность из двух сигналов, значения которых определяются в соответствии с таблицей истинности логической инверсии.

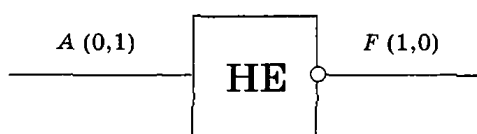


Рис. 3.12. Логический элемент «НЕ»

Простейшей моделью логического элемента «НЕ» может быть электрическая схема — инвертор, которую можно собрать из реальных электрических элементов или с использованием компьютерного конструктора «Начала электроники» (рис. 3.13).

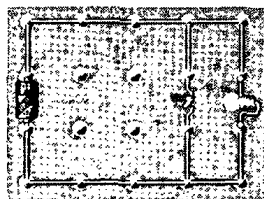
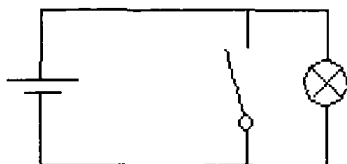


Рис. 3.13. Электрическая схема модели логического элемента «НЕ» и ее реализация в компьютерном конструкторе «Начала электроники»

В схеме инвертора один вход и один выход. Когда переключатель не замкнут (на входе 0), лампочка горит (на выходе 1). Наоборот, когда кнопку переключателя замыкают (на входе 1), лампочка гаснет (на выходе 0).

Контрольные вопросы

1. Объясните действие электрических схем, реализующих модели логических элементов, с точки зрения законов постоянного тока.

Компьютерный практикум

Windows-CD 

3.7. В редакторе схем нарисовать логические и электрические схемы логических элементов «И», «ИЛИ» и «НЕ».



`\\informatika10\logic\log.spl`

3.8. В компьютерном конструкторе «Начала электроники» создать модели электрических схем логических элементов «И», «ИЛИ» и «НЕ».

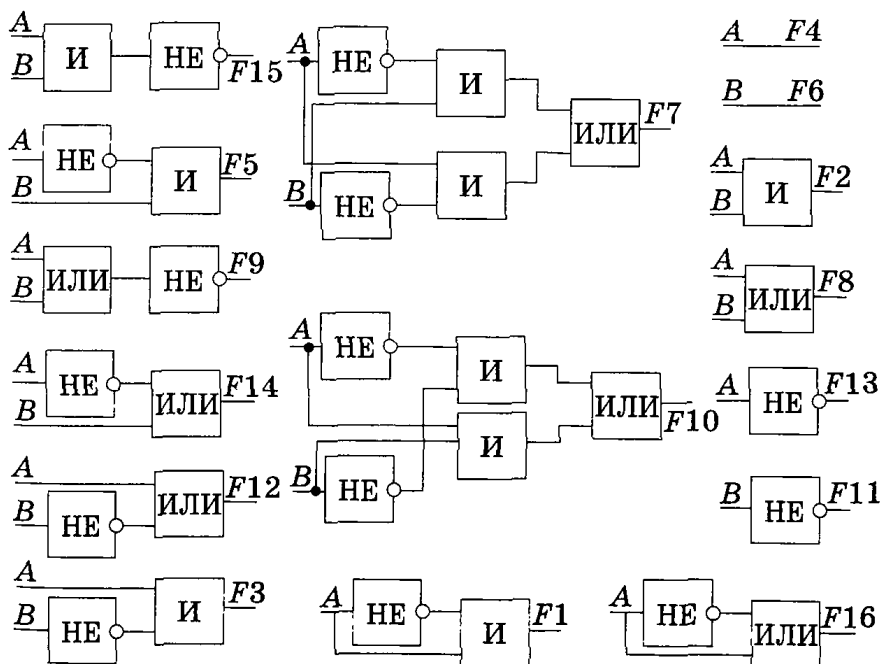


`\\informatika10\logic*.e`

3.9. В редакторе схем нарисовать логические схемы логических функций (см. табл. 3.8, задание 3.7).



`\\informatika10\logic\log.spl`



3.3.2. Сумматор двоичных чисел

В целях максимального упрощения работы компьютера все многообразие математических операций в процессоре сводится к сложению двоичных чисел. Поэтому главной частью

процессора является сумматор, который как раз и обеспечивает такое сложение.

Полусумматор. Вспомним, что при сложении двоичных чисел образуется сумма в данном разряде, при этом возможен перенос в старший разряд. Обозначим слагаемые A и B , перенос P и сумму S . Таблица сложения одноразрядных двоичных чисел с учетом переноса в старший разряд выглядит следующим образом (табл. 3.13).

Таблица 3.13. Таблица сложения одноразрядных двоичных чисел

Слагаемые		Перенос	Сумма
A	B	P	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Из этой таблицы сразу видно, что перенос можно реализовать с помощью операции логического умножения:

$$P = A \& B.$$

Получим теперь формулу для вычисления суммы. Значения суммы более всего совпадают с результатом операции логического сложения (кроме случая, когда на входы подаются две единицы, а на выходе должен получиться нуль).

Нужный результат достигается, если результат логического сложения умножить на инвертированный перенос. Таким образом, для определения суммы можно применить следующее логическое выражение:

$$S = (A \vee B) \& \overline{(A \& B)}.$$

Построим таблицу истинности для данного логического выражения (табл. 3.14) и убедимся в правильности нашего предположения.

Таблица 3.14. Таблица истинности логической функции $S = (A \vee B) \& \overline{(A \& B)}$

A	B	$A \vee B$	$A \& B$	$\overline{A \& B}$	$(A \vee B) \& \overline{(A \& B)}$
0	0	0	0	1	0
0	1	1	0	1	1
1	0	1	0	1	1
1	1	1	1	0	0

Теперь, на основе полученных логических выражений, можно построить из базовых логических элементов схему полусумматора.

По логической формуле переноса легко определить, что для получения переноса необходимо использовать логический элемент «И».

Анализ логической формулы для суммы показывает, что на выходе должен стоять элемент логического умножения «И», который имеет два входа. На один из входов подается результат логического сложения исходных величин $A \vee B$, т. е. на него должен подаваться сигнал с элемента логического сложения «ИЛИ».

На второй вход требуется подать результат инвертированного логического умножения исходных сигналов $A \& B$, т. е. на второй вход подается сигнал с элемента «НЕ», на вход которого, в свою очередь, поступает сигнал с элемента логического умножения «И» (рис. 3.14).

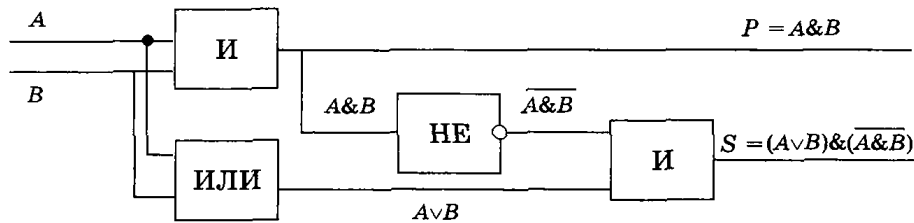


Рис. 3.14. Полусумматор двоичных чисел

Данная схема называется полусумматором, так как реализует суммирование одноразрядных двоичных чисел без учета переноса из младшего разряда.

Полный одноразрядный сумматор. Рассмотрим схему сложения двух n -разрядных двоичных чисел $a_n \dots a_i a_{i-1} \dots a_0$ и $b_n \dots b_i b_{i-1} \dots b_0$:

$$\begin{array}{r}
 p_i p_{i-1} \\
 \hline
 a_n \dots a_i a_{i-1} \dots a_0 \\
 + b_n \dots b_i b_{i-1} \dots b_0 \\
 \hline
 s_{n+1} s_n \dots s_i s_{i-1} \dots s_0
 \end{array}$$

При сложении цифр i -го разряда складываются a_i и b_i , а также p_{i-1} — перенос из $i-1$ -го разряда. Результатом будет s_i — сумма и p_i — перенос в старший разряд. Одноразрядный двоичный сумматор — это устройство с тремя входами и двумя выходами.

Таким образом, полный одноразрядный сумматор должен иметь три входа: A , B — слагаемые и P_0 — перенос из младшего разряда и два выхода: сумму S и перенос P . Таблица сложения в этом случае будет иметь следующий вид (табл. 3.15).

Таблица 3.15. Таблица сложения одноразрядных двоичных чисел с учетом переноса из младшего разряда

Слагаемые		Перенос из младшего разряда	Перенос	Сумма
A	B	P_0	P	S
0	0	0	0	0
0	1	0	0	1
1	0	0	0	1
1	1	0	1	0
0	0	1	0	1
0	1	1	1	0
1	0	1	1	0
1	1	1	1	1

Идея построения полного сумматора точно такая же, как и полусумматора. Из таблицы сложения видно, что перенос (логическая переменная P) принимает значение 1 тогда, когда хотя бы две входные логические переменные одновременно принимают значение 1. Таким образом, перенос реализуется путем логического сложения результатов попарного логического умножения входных переменных A , B , P_0 . Формула переноса получает следующий вид:

$$P = (A \& B) \vee (A \& P_0) \vee (B \& P_0).$$

Для получения значения суммы (логическая переменная S) необходимо результат логического сложения входных переменных A , B , P_0 умножить на инвертированный перенос \bar{P} :

$$S = (A \vee B \vee P_0) \& \bar{P}.$$

Данное логическое выражение дает правильные значения суммы во всех случаях, кроме одного, когда все входные логические переменные принимают значение 1. Действительно:

$$P = (1 \& 1) \vee (1 \& 1) \vee (1 \& 1) = 1,$$

$$S = (1 \vee 1 \vee 1) \& \bar{P} = 1 \& 0 = 0.$$

Для получения правильного значения суммы (для данного случая переменная S должна принимать значение 1) необ-

ходимо сложить полученное выше выражение для суммы с результатом логического умножения входных переменных A , B , P_0 . В результате логическое выражение для вычисления суммы в полном сумматоре принимает следующий вид:

$$S = (A \vee B \vee P_0) \& \overline{P_0} \vee (A \& B \& P_0).$$

Теперь можно построить логическую схему полного одно-разрядного сумматора (рис. 3.15).

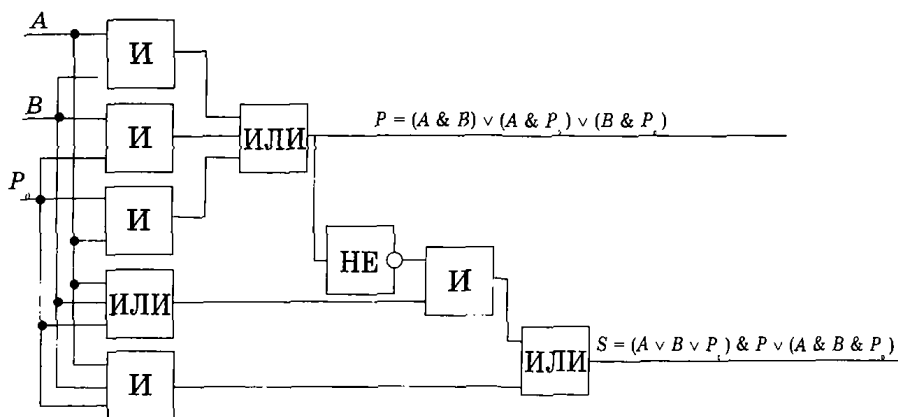


Рис. 3.15. Сумматор двоичных чисел

Многоразрядный сумматор процессора состоит из полных одноразрядных сумматоров. На каждый разряд ставится одно-разрядный сумматор, причем выход (перенос) сумматора млад-шего разряда подключен ко входу сумматора старшего разряда.

Контрольные вопросы

1. Какие значения будут иметь перенос и сумма при суммировании одноразрядных двоичных чисел и переноса из младшего разряда, равных 1? Проследите по логической схеме сумматора.

Компьютерный практикум



Windows-CD

- 3.10. В редакторе схем нарисовать логические схемы полу-сумматора и сумматора одноразрядных двоичных чи-сел.



\\informatika10\logic\log.spl

3.3.3. Триггер

Важнейшей структурной единицей оперативной памяти компьютера, а также внутренних регистров процессора является триггер. Триггер может находиться в одном из двух устойчивых состояний, что позволяет запоминать, хранить и считывать 1 бит информации.

Триггер можно построить из двух логических элементов «ИЛИ» и двух элементов «НЕ». При этом выход первого элемента «НЕ» соединен со входом второго элемента «ИЛИ», а выход второго элемента «НЕ» соединен со входом первого элемента «ИЛИ». Триггер имеет установочный вход S (от англ. set — установка) и вход сброса R (от англ. reset — сброс), а также выход Q (рис. 3.16).

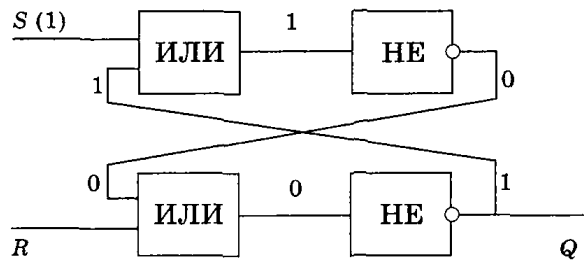


Рис. 3.16. Триггер

Для записи 1 бита на вход S (установочный) подается сигнал 1. Последовательно рассмотрев по логической схеме прохождение сигнала, видим, что на выходе триггера Q устанавливается 1. Триггер переходит в это состояние и будет находиться в нем и после того, как сигнал на входе S исчезнет. Таким образом, триггер будет устойчиво хранить 1 бит.

Для того чтобы сбросить бит данных и подготовиться к приему нового бита, подается сигнал 1 на вход R (сброс), после чего триггер возвратится к исходному «нулевому» состоянию.

Подача на оба входа S и R логической единицы может привести к неоднозначному результату, поэтому такая комбинация входных сигналов запрещена (табл. 3.16).


Таблица 3.16. Таблица состояний входов и выходов триггера

Входы		Выход Q
S	R	
0	0	Q
1	0	1
0	1	0

Контрольные вопросы

1. Проследите по логической схеме триггера, что происходит после поступления сигнала 1 на вход R (сброс).

Компьютерный практикум

Windows-CD 

- 3.11. В редакторе схем нарисовать логическую схему триггера.



\\informatika10\logic\log.spl

Глава 4

Алгоритмизация и основы объектно-ориентированного программирования

VisualStudio-CD

В процессе изучения данной темы рекомендуется использовать программное обеспечение для операционной системы Windows:

- систему объектно-ориентированного программирования Visual Basic 2005;
- систему объектно-ориентированного программирования Visual C#;
- систему объектно-ориентированного программирования Visual J#.



TurboDelphi-CD

- систему объектно-ориентированного программирования Turbo Delphi.



4.1. Алгоритм и кодирование основных алгоритмических структур

4.1.1. Алгоритм и его свойства

Алгоритмы могут описывать процессы преобразования самых разных объектов. Широкое распространение получили вычислительные алгоритмы, которые описывают преобразование числовых данных. Само слово «алгоритм» происходит от «algorithmi» — латинской формы написания имени выдающегося математика IX века аль-Хорезми, который сформулировал правила выполнения арифметических операций.

Результативность и дискретность. Алгоритм должен обеспечивать преобразование объекта из начального состояния в конечное состояние за определенное число дискретных шагов.

Массовость. Один и тот же алгоритм может применяться к большому количеству однотипных объектов.

Детерминированность. Исполнитель должен выполнять команды алгоритма в строго определенной последовательности.

Выполнимость и понятность команд. Алгоритм должен содержать команды, входящие в систему команд исполнителя и записанные на понятном для исполнителя языке.



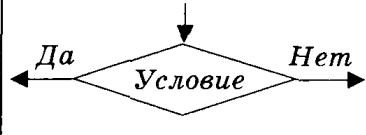
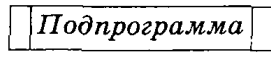
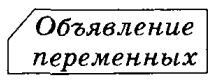
Алгоритм — это строго детерминированная последовательность действий, описывающая процесс преобразования объекта из начального состояния в конечное, записанная с помощью понятных исполнителю команд.

Блок-схемы алгоритмов. Блок-схема позволяет сделать алгоритм более наглядным и выделяет в алгоритме основные алгоритмические структуры (линейная, ветвление, выбор и цикл). Если исполнителем алгоритма является человек, то он может по блок-схеме легко проследить выполнение алгоритма, так как элементы блок-схем соединены стрелками, указывающими шаги выполнения алгоритма.

Элементы алгоритма изображаются на блок-схеме с помощью различных геометрических фигур, внутри которых записывается программный код (табл. 4.1).

Таблица 4.1. Элементы блок-схем

Элемент блок-схемы	Назначение элемента блок-схемы
	Прямоугольник с закругленными углами, применяется для обозначения начала или конца алгоритма
	Параллелограмм, предназначен для описания ввода или вывода данных, имеет один вход сверху и один выход внизу
	Прямоугольник, применяется для описания линейной последовательности команд, имеет один вход сверху и один выход внизу

Элемент блок-схемы	Назначение элемента блок-схемы
	Ромб, служит для обозначения условий в алгоритмических структурах «ветвление» и «выбор», имеет один вход сверху и два выхода (налево, если условие истинно, и направо, если условие ложно)
	Прямоугольник в прямоугольнике, применяется для вызова отдельно описанного алгоритма (подпрограммы)
	Прямоугольник со срезанным углом, применяется для объявления переменных или ввода комментариев

Контрольные вопросы

1. Какие из нижеперечисленных правил являются алгоритмами? Ответ обоснуйте:

- орфографические правила;
- правила выполнения арифметических операций;
- правила техники безопасности;
- правила перевода чисел из одной системы счисления в другую.

4.1.2. Алгоритмические структуры «ветвление» и «выбор»

Алгоритмическая структура «ветвление». В отличие от линейных алгоритмов, в которых команды выполняются последовательно одна за другой, в алгоритмическую структуру «ветвление» входит условие, в случае истинности условия реализуется последовательность команд *Серия 1*, в случае ложности — последовательность команд *Серия 2*.



В алгоритмической структуре «ветвление» одна или другая серия команд выполняется в зависимости от истинности условия.

Алгоритмическая структура «ветвление» может быть зафиксирована графически, с помощью блок-схемы (рис. 4.1).

На языках объектно-ориентированного программирования алгоритмическая структура «ветвление» кодируется с использованием оператора **if**. После первого ключевого слова **if** должно быть размещено условие. После ключевого слова **Then** (в языках Visual C# и Visual J# оно отсутствует) идет последовательность команд (*Серия 1*), которая должна выполняться, если условие принимает значение «истина». После ключевого слова **Else** размещается последовательность команд (*Серия 2*), которая должна выполняться, если условие принимает значение «ложь».

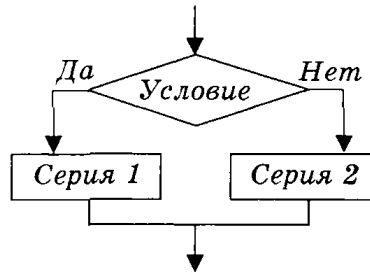


Рис. 4.1. Алгоритмическая структура «ветвление»

В сокращенной форме оператора ключевое слово **Else** отсутствует. (Необязательные части оператора записываются в квадратных скобках.) Тогда, если условие ложно, выполнение оператора условного перехода заканчивается и выполняется следующая строка программы.

 Плакаты. Таблица 7 «Кодирование алгоритмических структур «ветвление» и «выбор»

Алгоритмическая структура «выбор». Алгоритмическая структура «выбор» применяется для реализации ветвления со многими вариантами серий команд. В структуру выбора входят несколько **условий**, проверка которых осуществляется в последовательности их записи в структуре выбора. При истинности одного из условий (*Условие 1* или *Условие 2* и т. д.) выполняется соответствующая последовательность команд (*Серия 1* или *Серия 2* и т. д.). Если ни одно из условий не будет истинно, то будет выполнена последовательность команд *Серия*.



В алгоритмической структуре «выбор» выполняется одна из нескольких последовательностей команд при истинности соответствующего условия.

Алгоритмическая структура «выбор» может быть зафиксирована графически, с помощью блок-схемы (рис. 4.2).

На языках объектно-ориентированного программирования алгоритмическая структура «выбор» кодируется с использо-

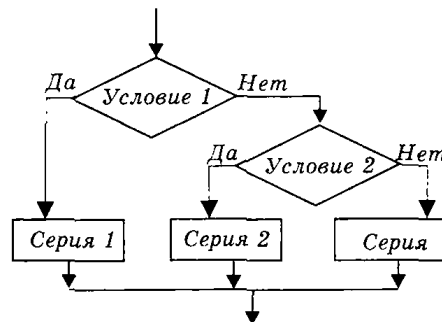


Рис. 4.2. Алгоритмическая структура «выбор»

ванием оператора выбора. На языке программирования Visual Basic .NET оператор выбора начинается с ключевых слов **Select Case**, на языках Visual C# и Visual J# — с ключевого слова **switch**, а на языке Turbo Delphi — с ключевого слова **Case**.


После ключевого слова записывается выражение (переменная или арифметическое выражение). Заданное выражение сравнивается с определенными значениями (или условиями). При истинности одного из условий начинает выполняться соответствующая серия команд. Если ни одно из условий не истинно, то будет выполнена серия команд после ключевого слова **Else** (в языках Visual Basic .NET и Turbo Delphi) или ключевого слова **default** (в языках Visual C# и Visual J#).

В сокращенной форме оператора ключевое слово **Else** (**default**) отсутствует. (Необязательные части оператора записываются в квадратных скобках.) Тогда, если все условия ложны, выполнение оператора выбора заканчивается и выполняется следующая строка программы.


 Плакаты. Таблица 7 «Кодирование алгоритмических структур «ветвление» и «выбор»

З а д а н и я

4.1. Составить и зафиксировать в форме блок-схемы алгоритм тестирования.

 Практическая работа 4.2. Проект «Тест»

4.2. Составить и зафиксировать в форме блок-схемы алгоритм выставления отметки в зависимости от количества ошибок.

 4.7. Графический интерфейс. Проект «Отметка»

4.1.3. Алгоритмическая структура «цикл»

В алгоритмическую структуру «цикл» входит серия команд, выполняемая многократно. Такая последовательность команд называется **телом цикла**.



В алгоритмической структуре «цикл» серия команд (тело цикла) выполняется многократно.

Циклические алгоритмические структуры бывают двух типов:

- циклы со счетчиком, в которых тело цикла выполняется определенное количество раз;
- циклы по условию, в которых тело цикла выполняется, пока истинно условие.

Цикл со счетчиком. Цикл со счетчиком используется, когда заранее известно, какое число повторений тела цикла необходимо выполнить. Количество повторений задается с использованием счетчика.

Когда заранее известно, какое число повторений тела цикла необходимо выполнить, можно воспользоваться оператором цикла со счетчиком **For**. В заголовке цикла устанавливается начальное значение переменной Счетчик, определяется величина ее конечного значения и величина изменения значения за один шаг. Затем располагаются многократно выполняющиеся операторы, являющиеся телом цикла.



Плакаты. Таблица 8 «Кодирование алгоритмической структуры «цикл»


Цикл с условием. Цикл с условием используется, когда заранее неизвестно, какое количество раз должно повториться тело цикла. В таких случаях количество повторений зависит от некоторого условия. Цикл называется **циклом с предусловием**, если условие выхода из цикла стоит в начале, перед телом цикла. Цикл с предусловием не выполняется даже один раз в случае ложности условия.

Цикл с предусловием реализуется с помощью инструкций **While** (в языке Visual Basic .NET **Do While**). Проверка условия выхода из цикла проводится до начала цикла с помощью ключевого слова **While**. Ключевое слово **While** обеспечивает выполнение цикла, пока истинно условие. Как только условие примет значение «ложь», выполнение цикла закончится.

В языке Visual Basic .NET используется также ключевое слово **Until**. Ключевое слово **Until** обеспечивает выполнение цикла до тех пор, пока условие не станет истинным, т. е. пока условие имеет значение «ложь». Как только условие примет значение «истина», выполнение цикла закончится.

Цикл называется **циклом с постусловием**, если условие выхода из цикла стоит в конце, после тела цикла. Цикл с постусловием выполняется обязательно, как минимум, один раз, независимо от того, истинно условие или нет.

Цикл с постусловием реализуется с помощью инструкций **Do** (в языке Turbo Delphi **Repeat**). Проверка условия выхода из цикла проводится после цикла с помощью ключевого слова **While** (в языке Turbo Delphi **Until**). Как только условие примет значение «ложь», выполнение цикла закончится. В языке Visual Basic .NET используется также ключевое слово **Until**.

 Плакаты. Таблица 8 «Кодирование алгоритмической структуры «цикл»

Алгоритмическая структура «цикл» может быть зафиксирована графически, с помощью блок-схемы.

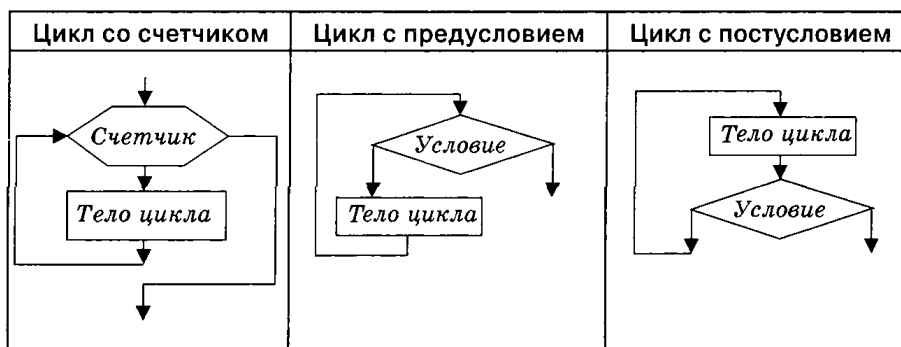



Рис. 4.3. Алгоритмическая структура «цикл»


Вспомогательные алгоритмы. При разработке сложного алгоритма целесообразно стараться выделить в нем последовательности действий, которые выполняют решение каких-либо подзадач и могут многократно вызываться из основного алгоритма. Такие алгоритмы называются вспомогательными и в алгоритмических языках программирования реализуются в форме подпрограмм, процедур или функций, которые вызываются из основной программы.

З а д а н и я

4.3. Составить и зафиксировать в форме блок-схемы алгоритм вычисления факториала числа.

 4.10. Итерация и рекурсия. Проект «Факториал»

4.4. Составить и зафиксировать в форме блок-схемы алгоритм определения максимального элемента в массиве.

 4.16.2. Поиск элемента в массивах.
Проект «Поиск максимального элемента»

4.2. История развития языков программирования

Выполнение алгоритма может быть автоматически реализовано техническими устройствами, среди которых особое место занимает компьютер. При этом говорят, что компьютер исполняет программу (последовательность команд), реализующую алгоритм.



Алгоритм, записанный на «понятном» компьютеру языке программирования, называется **программой**.

Машинный язык. На заре компьютерной эры, в 40–50-е годы XX века, программы писались на машинном языке (computer language) и представляли собой очень длинные последовательности нулей и единиц. Составление и отладка таких программ были чрезвычайно трудоемким делом. Программы на машинных языках были машинно зависимыми, т. е. для каждой ЭВМ необходимо было создавать свою собственную программу, так как в ней в явной форме учитывались аппаратные ресурсы ЭВМ.

Ассемблер. В начале 50-х годов XX века были созданы языки программирования ассемблеры (assembly language). Вместо одних только нулей и единиц программисты теперь могли пользоваться операторами (MOV, ADD, SUB и т. д.), которые были похожи на слова английского языка. Для преобразования текста программы на ассемблере в понятный компьютеру машинный код использовался компилятор, который

загружался в оперативную память ЭВМ. Программы на ассемблере были также машинно зависимыми, т. е. ассемблеры для различных процессоров существенно различались между собой.

Первые языки программирования высокого уровня. С середины 50-х годов XX века начали создаваться первые языки программирования высокого уровня (high-level languages). Эти языки были машинно независимыми языками программирования, так как использовали универсальную компьютерную логику и не были привязаны к типу ЭВМ. Однако для каждого языка и каждого типа ЭВМ должны были быть разработаны собственные компиляторы, которые загружались в оперативную память.

Языки программирования высокого уровня создавались и использовались для решения разных задач:

- язык FORTRAN (расшифровывается FORmula TRANslator — транслятор формул) был предназначен для научных и технических расчетов;
- язык COBOL (Common Business-Oriented Language — стандартный язык для делового применения) в основном предназначался для коммерческих приложений, обрабатывавших большие объемы нечисловых данных;
- язык BASIC (Beginner's All-Purpose Symbolic Instruction Code — универсальный язык символьных инструкций для начинающих) отличается простотой изучения и был ориентирован на начинающих программистов.

Алгоритмические языки программирования. С начала 80-х годов XX века начали создаваться алгоритмические языки программирования, которые позволили программистам перейти к структурному программированию. Отличительной чертой этих языков было использование операторов ветвления, выбора и цикла и отказ от хаотического использования оператора `goto`. Наибольшее влияние на переход к структурному алгоритмическому программированию оказали:

- язык Pascal (назван его создателем Никлаусом Виртом в честь великого физика Блеза Паскаля) — алгоритмический язык, который позволяет легко кодировать основные алгоритмические структуры;
- язык C («Си»), позволяющий создавать быстро и эффективно выполняющийся программный код.

Языки объектно-ориентированного программирования. С 90-х годов XX века начали создаваться объектно-ориентированные языки программирования. В основу этих языков

были положены программные объекты, которые объединяли данные и методы их обработки. Необходимо подчеркнуть, что в языках объектно-ориентированного программирования сохранялся алгоритмический стиль программирования. С течением времени для этих языков были разработаны интегрированные среды разработки, позволяющие визуально конструировать графический интерфейс приложений:

- язык C++ является прямым потомком алгоритмического языка C;
- язык Object Pascal был разработан компанией Borland на основе алгоритмического языка Pascal. После создания интегрированной среды разработки система программирования получила название Delphi, а свободно распространяемая версия — Turbo Delphi;
- язык Visual Basic был создан корпорацией Microsoft на основе языка QBasic для разработки приложений с графическим интерфейсом в среде операционной системы Windows.

Языки программирования для компьютерных сетей. В 90-е годы XX века в связи с бурным развитием Интернета были созданы языки, обеспечивающие межплатформенную совместимость. На подключенных к Интернету компьютерах с различными операционными системами (Windows, Linux, Mac OS и др.) могли выполняться одни и те же программы. Исходная программа на таких языках компилируется в промежуточный код, который исполняется на компьютере встроенной в браузер виртуальной машины:

- язык Java, полноценный объектно-ориентированный язык, был разработан фирмой Sun Microsystems для создания сетевого программного обеспечения;
- язык JavaScript, язык сценариев Web-страниц, разработан компанией Netscape.

Языки программирования на платформе .NET. В настоящее время многие программисты выбирают интегрированную систему программирования Visual Studio .NET на платформе .NET Framework, разработанной корпорацией Microsoft. Эта платформа предоставляет возможность создавать приложения в различных системах объектно-ориентированного программирования, в которых для создания программного кода используются объектно-ориентированные языки программирования, в том числе:

- на языке Visual Basic .NET, созданном на основе языка Visual Basic и сохраняющем простоту своих предшественников;
- на языке Visual C# (читается С-шарп), созданном на основе языков C++ и Java;
- на языке Visual J# (читается J-шарп), созданном на основе языков Java и JavaScript.

История развития языков программирования

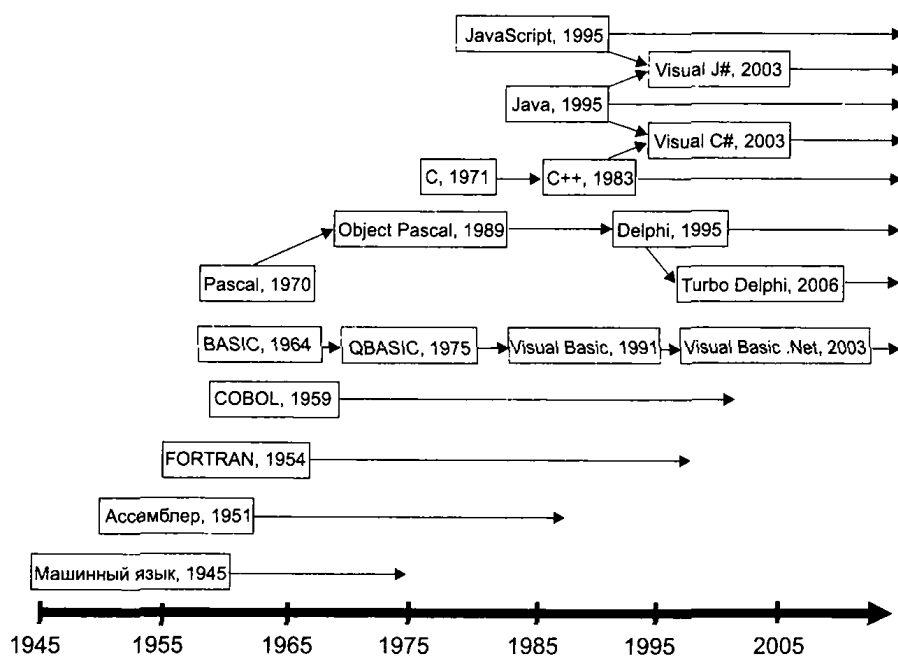


Рис. 4.4. История развития языков программирования

Контрольные вопросы

1. В чем состоит преимущество языков высокого уровня перед машинным языком и ассемблером?
2. Каковы характерные особенности алгоритмических языков программирования? Объектно-ориентированных языков программирования?
3. Какими причинами было обусловлено появление языков Java и JavaScript?
4. На основе каких языков возникли языки программирования на платформе .NET?

4.3. Введение в объектно-ориентированное визуальное программирование

4.3.1. Объекты: свойства и методы

Объекты (Objects). Основной единицей в объектно-ориентированном программировании является программный объект, который объединяет в себе как описывающие его свойства, так и действия объекта (процедуры) — методы. Если говорить образно, то объекты — это существительные, свойства объекта — это прилагательные, а методы объекта — это глаголы.



Программные объекты обладают свойствами, имеют методы, для них можно описать реакцию на события.

Классы объектов являются «шаблонами», определяющими наборы свойств, методов и событий, по которым создаются объекты. Основными классами объектов являются объекты, реализующие графический интерфейс проектов.

Объект, созданный по «шаблону» класса объектов, является экземпляром класса и наследует весь набор свойств, методов и событий данного класса. Каждый экземпляр класса имеет уникальное для данного класса имя. Различные экземпляры класса обладают одинаковым набором свойств, однако значения свойств у них могут отличаться.

Свойства объектов (Properties). Каждый объект обладает определенным набором свойств. Существует несколько основных ситуаций, в которых можно менять свойства объектов. Во время разработки проекта [design] можно установить первоначальные значения свойств объекта.

В режиме выполнения проекта [run] можно установить значения свойств объекта с использованием программного кода. Для присваивания свойству объекта нового значения в левой части строки программного кода необходимо указать имя объекта и затем название свойства. В правой части строки необходимо записать конкретное значение свойства. Например, программный код вывода на надпись текста в различных языках программирования будет выглядеть следующим образом.

Язык Visual Basic .NET:
 Label1.Text = "Текст"



Язык Visual C#:
 Label1.Text = "Текст";

Язык Turbo Delphi:
 Label1.Caption := 'Текст';



В программном коде для доступа к свойствам и методам используется точечная нотация (Dot-запись), при которой имена объектов, свойств и методов отделяются друг от друга знаком точка «.».

Методы объектов (Methods). Для того чтобы объект выполнил какую-либо операцию, необходимо применить метод, которым он обладает. Многие методы имеют аргументы, которые позволяют задать параметры выполняемых действий. Обратиться к методу объекта можно также с использованием точечной нотации, причем аргументы метода заключаются в скобки. Например, для добавления элемента в список в языках объектно-ориентированного программирования используется метод `Add()`.

Язык Visual Basic .NET:
 ListBox1.Items.Add("Элемент списка")



Язык Visual C#:
 listBox1.Items.Add("Элемент списка");

Язык Visual J#:
 listBox1.getItems().Add("Элемент списка");

Язык Turbo Delphi:
 ListBox1.Items.Add('Элемент списка');

Контрольные вопросы

1. Чем различаются понятия «класс объектов» и «экземпляр класса»?
2. В экземпляре класса можно изменить набор свойств? Набор методов? Значения свойств?

4.3.2. События

События (Events). Событие представляет собой изменение некоторого состояния, распознаваемое объектом. Для реакции на это изменение могут быть описаны некоторые методы — обработчики, обрабатывающие события в программном коде. Событие может создаваться пользователем (например, щелчок мышью или нажатие клавиши) или быть результатом воздействия других программных объектов. Каждый элемент управления может реагировать на различные события, однако есть события (например, `Click` — щелчок), на которые реагирует большинство типов элементов управления.

Обработчик события. Для каждого события можно запрограммировать обработчик события (событийную процедуру). Если пользователь производит какое-либо воздействие на элемент графического интерфейса (например, щелчок), в качестве обработчика выполняется некоторая последовательность действий в форме процедуры.

Каждый обработчик события представляет собой процедуру, которая реализует определенный алгоритм. Создание программного кода обработчика события производится с использованием алгоритмических структур различных типов (линейная, ветвление, выбор или цикл).



Обработчик события представляет собой процедуру, которая начинает выполняться после реализации определенного события.

Имя обработчика события (событийной процедуры) включает в себя имя объекта и имя события. После имени событийной процедуры в скобках указываются параметры, которые позволяют правильно обработать событие. В событийной процедуре на языках .NET существует два параметра, а в языке Turbo Delphi — один параметр.

Ниже на четырех языках программирования показан пустой обработчик события `Click` элемента управления `Button1`.

Язык Visual Basic .NET:

```
Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
End Sub
```

Язык Visual C#:

```
private void button1_Click(object sender,
System.EventArgs e)
{
}
```



Язык Visual J#:

```
private void button1_Click (Object sender,
System.EventArgs e)
{
}
```

Язык Turbo Delphi:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
end;
```

Первый параметр, `sender`, предоставляет ссылку на объект, который вызывает событие. Например, при щелчке мышью по кнопке наступает событие `Click` данной кнопки и ее адрес передается обработчику события и сохраняется в аргументе `sender`.

В языках программирования на платформе .NET используется и второй параметр `e`, который передает данные, характерные для обрабатываемого события. Этот параметр обычно имеет тип `System.EventArgs`, однако существуют события, которые требуют особый тип данных.

Например, если обрабатываются события мыши, то используется параметр `MouseEventArgs`. С помощью этого параметра можно получить сведения о координатах мыши, какая была нажата кнопка и сколько было сделано щелчков. Для вывода всех свойств аргумента `e` (рис. 4.5) достаточно ввести в процедуре обработчика события:

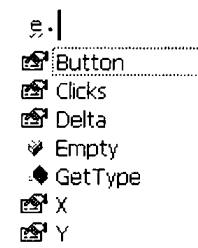


Рис. 4.5. Возможные свойства аргумента `e`



4.13.1. Графика в языках программирования Visual Basic .NET, Visual C# и Visual J#

Контрольные вопросы

1. В чем состоит различие между обработчиками событий в языках программирования на платформе .NET и событийными процедурами в языке Turbo Delphi?

4.3.3. Проекты и приложения

Проект (Project). С одной стороны, система объектно-ориентированного визуального программирования является **системой программирования**, так как позволяет кодировать алгоритмы на данном языке. С другой стороны, система объектно-ориентированного визуального программирования является **средой проектирования**, так как позволяет осуществлять визуальное конструирование графического интерфейса.

Результатом процессов программирования и проектирования является **проект**, который объединяет в себе программный код и графический интерфейс.

В объектно-ориентированном программировании проект может включать несколько форм, причем каждой форме, с помощью которой реализуется графический интерфейс проекта, соответствует свой **программный модуль формы**. Кроме того, в состав проекта могут входить отдельные самостоятельные **программные модули**.



Проект включает в себя **программные модули форм** и самостоятельные **программные модули** в виде отдельных файлов. Проект может быть запущен на выполнение только из системы объектно-ориентированного программирования.

Решения (Solution). В системах объектно-ориентированного программирования Visual Basic .NET, Visual C# и Visual J# проекты объединяются в решения, а в системе Turbo Delphi — в группы. Решение (группа) включает один или несколько проектов, которые в упорядоченном виде в системах Visual Basic .NET, Visual C# и Visual J# отображаются в *Обзревателе решений* (рис. 4.8), а системе Turbo Delphi — в окне

Project Manager (рис. 4.11). Решение (группа) создается автоматически при создании нового проекта и при необходимости к решению можно добавлять новые проекты. Решения (группы) позволяют работать с несколькими проектами в пределах одного экземпляра системы объектно-ориентированного программирования.

Интерпретаторы и компиляторы. Для того чтобы процессор мог выполнить программу, эта программа и данные, с которыми она работает, должны быть загружены в оперативную память.

Итак, мы создали программу на языке программирования (некоторый текст) и загрузили ее в оперативную память. Теперь мы хотим, чтобы процессор ее выполнил, однако процессор «понимает» команды на машинном языке, а наша программа написана на языке программирования. Как быть?

Необходимо, чтобы в оперативной памяти находилась программа-переводчик (**транслятор**), автоматически переводящая с языка программирования на машинный язык. Компьютер может выполнять программы, написанные только на том языке программирования, транслятор которого размещен в оперативной памяти компьютера.

Трансляторы языков программирования бывают двух типов: **интерпретаторы и компиляторы**. Интерпретатор — это программа, которая обеспечивает последовательный «перевод» инструкций программы на машинный язык с одновременным их выполнением. Поэтому при каждом запуске программы на выполнение эта процедура повторяется. Достоинством интерпретаторов является удобство отладки программы (поиска в ней ошибок), так как возможно «пошаговое» ее исполнение, а недостатком — сравнительно малая скорость выполнения.

Компилятор действует иначе, он переводит весь текст программы на машинный язык и сохраняет его в исполнимом файле (обычно с расширением *exe*). Затем этот уже готовый к исполнению файл, записанный на машинном языке, можно запускать на исполнение многократно. Достоинством компиляторов является большая скорость выполнения программы, а недостатком — трудоемкость отладки, так как невозможно пошаговое выполнение программы.

Системы объектно-ориентированного программирования позволяют программисту контролировать в интегрированной среде выполнение программ с помощью отладчика. Это дает возможность отлаживать программу пошагово.



Приложение интегрирует программный код и графический интерфейс в одном исполнимом файле, который может запускаться непосредственно в операционной системе.

Этапы разработки проектов. Создание проектов и приложений в системах объектно-ориентированного программирования можно условно разделить на несколько этапов:

1. Создание графического интерфейса проекта. На форму помещаются элементы управления, которые должны обеспечить взаимодействие проекта с пользователем.
2. Установка значений свойств объектов графического интерфейса. В режиме конструирования задаются значения свойств формы и элементов управления, помещенных ранее на форму.
3. Создание и редактирование программного кода. Создаются заготовки обработчиков событий (двойной щелчок мышью по элементу управления вызывает заготовку обработчика того события, которое является для данного элемента управления наиболее часто используемым). Затем в редакторе программного кода производится ввод и редактирование программного кода обработчиков событий.
4. Сохранение проекта. Так как проекты включают в себя несколько файлов, рекомендуется для каждого проекта создать отдельную папку на диске. Сохранение проекта производится с помощью пунктов меню *Файл*.
5. Компиляция проекта в приложение. Сохраненный проект может выполняться только в самой системе программирования. Для того чтобы преобразовать проект в приложение, которое может выполняться непосредственно в среде операционной системы, необходимо выполнить компиляцию проекта, в процессе которой приложение сохраняется в исполнимом файле (с расширением *exe*).

Контрольные вопросы

1. В чем состоит различие между интерпретаторами и компиляторами?
2. В чем состоит различие между проектом и приложением?

4.4. Система объектно-ориентированного программирования Microsoft Visual Studio .NET

4.4.1. Платформа .NET Framework

Платформа .NET Framework является основой системы программирования Microsoft Visual Studio .NET. Она позволяет программировать на различных языках (Visual Basic .NET, Visual C#, Visual J# и др.) и достаточно легко переносить приложения с одного языка программирования на другой.



Платформа .NET Framework существует в двух версиях:

- .NET Framework 1.1 входит в состав Visual Studio .NET 2003;
- .NET Framework 2.0 входит в состав Visual Studio 2005 Express Edition.

Общезыковая среда выполнения программ. Основным компонентом .NET Framework является общезыковая среда выполнения программ (CLR — Common Language Runtime), которая обеспечивает компиляцию программ в два этапа. Сначала программа, написанная на объектно-ориентированном языке, компилируется в промежуточный код на общем промежуточном языке (CIL — Common Intermediate Language), а затем он компилируется в машинный код.



Язык CIL по своему назначению и набору команд похож на язык ассемблер, однако он ориентирован не на систему команд конкретного кремниевого процессора, а на систему команд общезыковой среды выполнения программ CLR.

Проект «Консольное приложение». На платформе .NET Framework с использованием языков программирования Visual Basic .NET, Visual C# и Visual J# создать консольное приложение (т. е. приложение, не имеющее графического интерфейса), которое выводит название языка программирования.



Проект «Консольное приложение» на языке программирования Visual Basic .NET

1. В простом текстовом редакторе, например в Блокноте, введем программный код консольного приложения.

```
Imports System
Class HelloVB
Shared Sub Main()
Console.WriteLine("Язык программирования Visual Basic")
End Sub
End Class
```

2. Сохранить программный код консольного приложения в файле HelloVB.vb на жестком диске, например в папке D:\HelloVB\.

Для компиляции программного кода необходимо использовать компилятор командной строки языка программирования Visual Basic .NET vbc.exe, который входит в состав платформы .NET Framework. Компилятор находится в папке C:\WINDOWS\Microsoft.NET\Framework\v1.1.4322.

3. Скопировать компилятор командной строки vbc.exe в папку D:\HelloVB\.

Создадим исполнимый файл консольного приложения HelloVB.exe.

4. Запустить окно командной строки командой [Пуск-Программы-Стандартные-Командная строка].
5. В окне командной строки перейти на диск D:, а затем с использованием команды CD (change directory) перейти в папку D:\HelloVB\, где хранятся файл с программным кодом консольного приложения и компилятор командной строки.
6. В окне командной строки ввести команду:
D:\HelloVB>vbc.exe HelloVB.vb

Теперь можно запустить полученный исполнимый файл консольного приложения HelloVB.exe.

7. Ввести команду:
D:\HelloVB>HelloVB.exe
8. В окне командной строки будет напечатано (рис. 4.6):
Язык программирования Visual Basic

```

Командная строка
Microsoft Windows XP [Версия 5.1.2600.1
(C) Корпорация Майкрософт, 1985-2001.
C:\Documents and Settings\НДУ>D:
D:\>CD HelloUB
D:\HelloUB>vbcomp.exe HelloUB.vb
Microsoft (R) Visual Basic .NET Compiler version 7.10.3052.4
for Microsoft (R) .NET Framework version 1.1.4322.573
Copyright (C) Microsoft Corporation 1987-2002. All rights reserved.
D:\HelloUB>HelloUB.exe
Язык программирования Visual Basic
  
```

Рис. 4.6. Создание и запуск консольного приложения



Проект «Консольное приложение» на языке программирования C#

1. В текстовом редакторе Блокнот ввести программный код консольного приложения:

```

using System;

class HelloC
{
    static void Main(string[] args)
    {
        Console.WriteLine("Язык программирования C#");
    }
}
  
```

2. Сохранить программный код консольного приложения в файле HelloC.cs на жестком диске, например в папке D:\HelloC\.

Для компиляции программного кода необходимо использовать компилятор командной строки языка программирования C# csc.exe, который входит в состав платформы .NET Framework. Компилятор находится в папке C:\WINDOWS\Microsoft.NET\Framework\v1.1.4322.

Создадим исполнимый файл консольного приложения HelloC.exe.

3. Запустить окно командной строки командой [Пуск-Программы-Стандартные-Командная строка].
4. В окне командной строки перейти на диск D:, а затем с использованием команды CD (change directory) перейти в

папку D:\HelloC\, где хранится файл с программным кодом консольного приложения.

5. Ввести команду компиляции:
D:\HelloC>C:\WINDOWS\Microsoft.NET\Framework\v1.1.4322\csc.exe HelloC.cs

Теперь можно запустить полученный исполнимый файл консольного приложения HelloC.exe.

6. В окне командной строки ввести команду:
D:\HelloC>HelloC.exe
7. В окне командной строки будет напечатано:
Язык программирования C#




Проект «Консольное приложение» на языке программирования J#

1. В текстовом редакторе Блокнот ввести программный код консольного приложения:
import System;
Console.WriteLine("Язык программирования J#");
2. Сохранить программный код консольного приложения в файле HelloJ.jsl на жестком диске, например в папке D:\HelloJ.
3. Запустить окно командной строки командой [Пуск-Программы-Стандартные-Командная строка].
4. В окне командной строки перейти на диск D:, а затем с использованием команды CD (change directory) перейти в папку D:\HelloJ, где хранится файл с программным кодом консольного приложения.
5. Ввести команду компиляции:
D:\HelloJ>C:\WINDOWS\Microsoft.NET\Framework\v1.1.4322\jsc.exe HelloJ.jsl
6. В окне командной строки ввести команду:
D:\HelloJ>HelloJ.exe
7. В окне командной строки будет напечатано:
Язык программирования J#

Компьютерный практикум



Windows-CD 

- 4.1. Создать в текстовом редакторе проект «Консольное приложение».

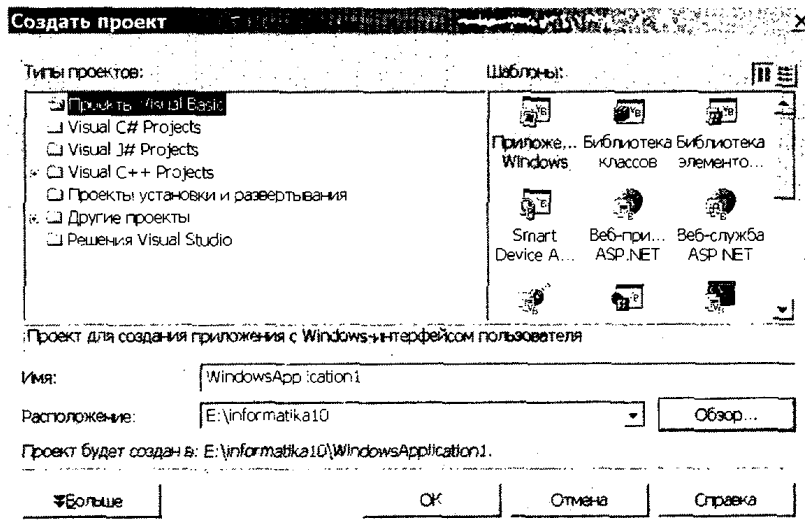


informatika10\...\Hello*

4.4.2. Интегрированная среда разработки языков Visual Basic .NET, Visual C# и Visual J#

Состав Visual Studio .NET. Microsoft Visual Studio .NET — это инструмент разработки программ, позволяющий писать программы на нескольких языках программирования .NET. В состав Visual Studio .NET входят следующие языки программирования (рис. 4.7):

- Visual Basic .NET;
- Visual C# (произносится Си-шарп);
- Visual J# (произносится Джей-шарп);
- Visual C++ (произносится С плюс плюс);
- ASP.NET (Web Developer);
- Справочная система.



4.7. Выбор языка программирования в Visual Studio.NET при создании нового проекта

Русскоязычная справка по
Visual Studio .NET

<http://msdn.microsoft.com/library/rus/>



Система объектно-ориентированного программирования Visual Studio .NET существует в двух версиях:

- Visual Studio .NET 2003 — полностью локализованная русскоязычная версия;
- Visual Studio 2005 Express Edition — англоязычная свободно распространяемая версия для начинающих программистов.

Интегрированная среда разработки. Visual Studio .NET является системой визуального объектно-ориентированного программирования на платформе .NET Framework. Система программирования Visual Studio .NET предоставляет пользователю для работы со всеми языками .NET удобный, причем одинаковый, графический интерфейс IDE (Integrated Development Environment — Интегрированная среда разработки) (рис. 4.8).

Визуальное конструирование графического интерфейса проекта выполняется в *Конструкторе форм (Windows*

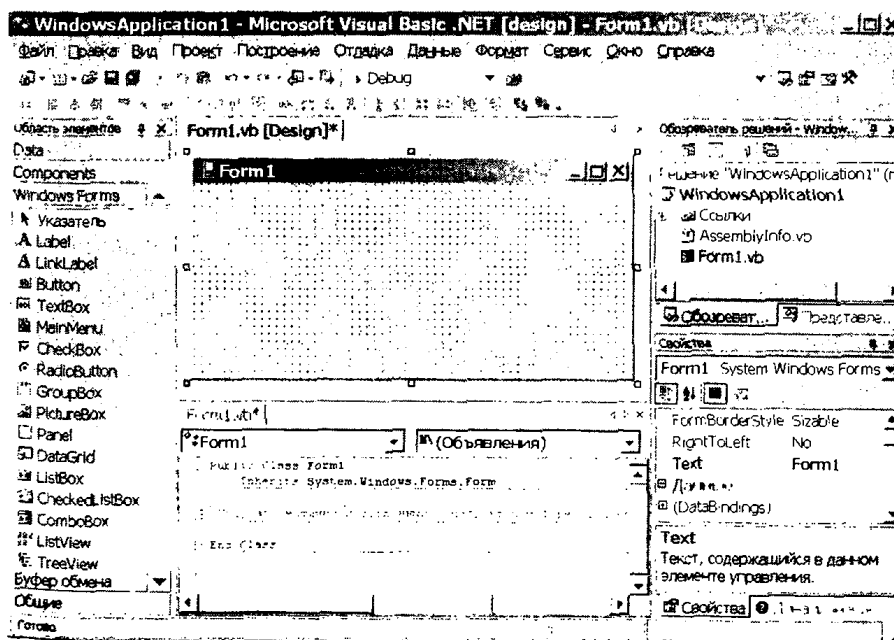


Рис. 4.8. Интегрированная среда разработки Visual Studio .NET (на примере Visual Basic .NET 2003)

Forms Designer), который по умолчанию размещается на вкладке *Form1.vb[Design]*. Конструктор форм располагается в центре окна интегрированной среды разработки и содержит **форму** (по умолчанию *Form1*), являющуюся основой графического интерфейса проекта. На форму можно поместить различные **элементы управления**: кнопки (*Button*), текстовые поля (*TextBox*), надписи (*Label*) и т. д.

Пиктограммы элементов управления располагаются на панели *Область элементов (ToolBox)*, которая размещается в левой части окна интегрированной среды разработки.

С формой связан программный код проекта, для ввода и редактирования которого служит *Редактор программного кода* (по умолчанию размещается на вкладке *Form1.vb*). Редактор программного кода поддерживает язык проекта, т. е. предлагает подсказки при вводе программы, выделяет синтаксические ошибки, структурирует программный код отступами и т. д.

Для перехода в окно программного кода можно просто щелкнуть по ярлыку вкладки *Form1.vb* или ввести команду [*Вид-Код*] (*[View-Code]*). Для обратного перехода в *Конструктор форм* можно щелкнуть по ярлыку вкладки *Form1.vb [Design]* или ввести команду [*Вид-Конструктор*] (*[View-Designer]*).

Первые части названий окна конструктора форм и окна редактора программного кода совпадают не случайно, так как в них создается и редактируется один и тот же файл *Form1.vb*. Этот файл и другие файлы проекта можно увидеть в окне *Обозреватель решений (Solution Explorer)*, который размещается в правом верхнем углу интегрированной среды разработки.

Справа располагается окно *Свойства (Properties)*. Окно содержит список свойств, относящихся к выбранному объекту (форме или элементу управления на форме). В левом столбце находятся названия свойств, а в правом — их значения. Установленные по умолчанию значения могут быть изменены.

Настройка интегрированной среды разработки. Рекомендуется провести настройку интегрированной среды разработки. Для этого необходимо ввести команду [*Сервис-Параметры...*] (*[Tools-Options...]*) и в появившемся диалоговом окне *Параметры* установить нужные параметры *Конструктора форм, Редактора программного кода* и др.

В процессе создания графического интерфейса проекта важно упорядоченно разместить на форме элементы управле-

ния. Для этого необходимо в левой части диалогового окна *Параметры* выбрать пункт *Конструктор Windows Form*, а в правой части установить шаг сетки на форме и ее видимость (рис. 4.9).

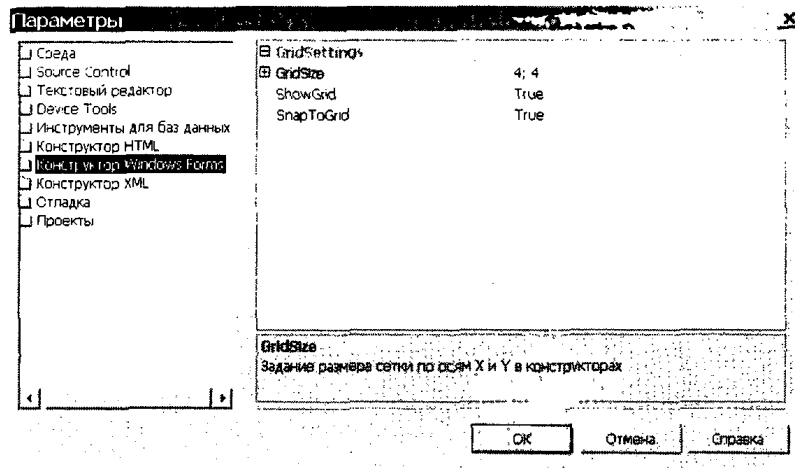


Рис. 4.9. Установка параметров *Конструктора форм*

В процессе создания, редактирования и отладки программного кода проекта целесообразно пронумеровать строки программы. Для этого необходимо в левой части диалогового окна *Параметры* выбрать пункт *Текстовый редактор*, а в правой части установить флажок в пункте *Номера строк* (рис. 4.10).

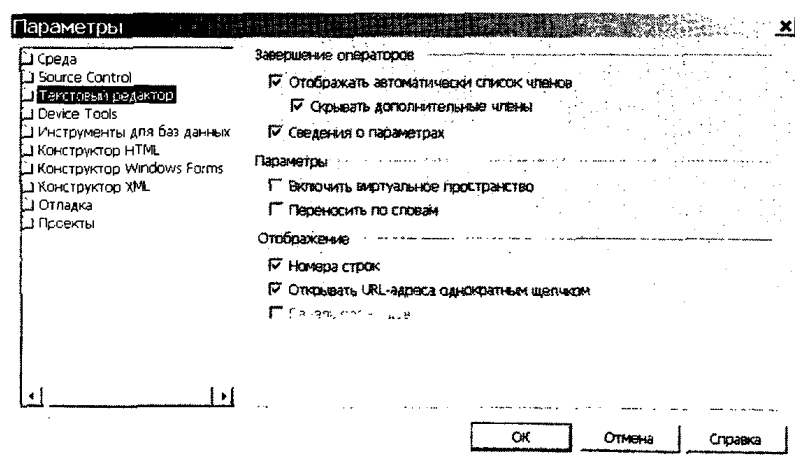



Рис. 4.10. Установка параметров *Редактора программного кода*


 **Преобразование проектов из формата Visual Basic 5.0 или 6.0 в формат Visual Basic .NET 2003.** Для преобразования проектов более ранних версий языка (VB 5.0 CSE, Visual Basic 6.0) в проекты на Visual Basic .NET используется *Мастер обновлений*. Для преобразования проекта необходимо ввести команду [*Файл-Открыть-Проект...*], найти в папке проекта на языке Visual Basic 5.0 или 6.0 файл с расширением *vbr* и следовать указаниям пяти диалоговых окон *Мастера Visual Basic*.

Преобразование проектов из формата Visual Basic .NET 2003 в формат Visual Basic 2005 Express Edition. Для преобразования проекта необходимо ввести команду [*File-Open-Project...*], найти в папке проекта на языке Visual Basic .NET 2003 файл с расширением *vbrproj* и следовать указаниям пяти диалоговых окон *Visual Basic Upgrade Wizard*.

Компьютерный практикум



4.2. Создать в системе программирования Visual Studio проект «Консольное приложение».

Windows-CD 



...\informatika10...\Hello*

4.5. Система объектно-ориентированного программирования Turbo Delphi

Система объектно-ориентированного программирования Turbo Delphi позволяет визуально создавать графический интерфейс к проектам на языке программирования Object Pascal.

Окно системы программирования Turbo Delphi. Система программирования Turbo Delphi предоставляет пользователю удобный графический интерфейс в процессе разработки проекта. После запуска Turbo Delphi появится окно системы программирования Turbo Delphi, которое включает в себя (рис. 4.11):

- строку заголовка *Turbo Delphi*;
- строку *Главного меню* под строкой заголовка;

- кнопки с пиктограммами наиболее часто используемых команд под строкой *Главного меню* слева.

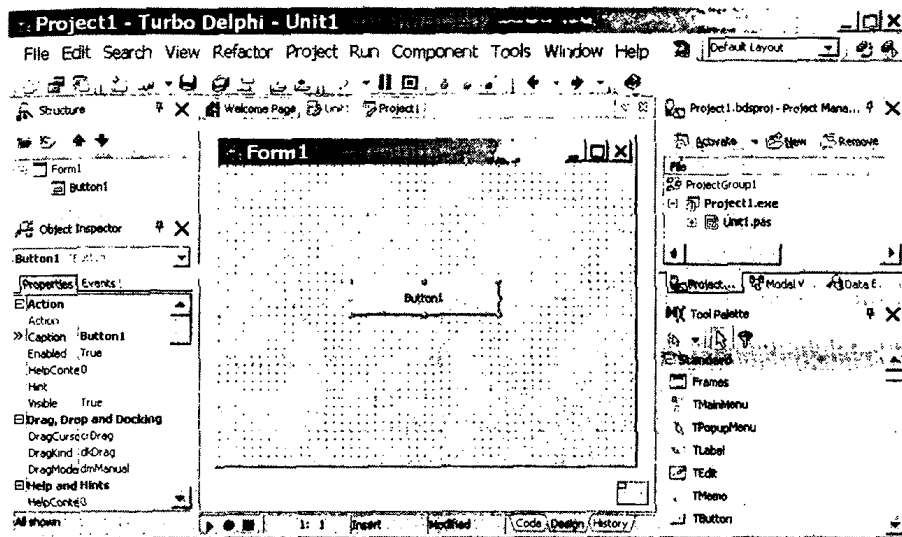


Рис. 4.11. Система программирования Turbo Delphi

Окно Конструктор форм. В центре располагается окно *Конструктор форм*. Для создания нового проекта необходимо ввести команду [*File-New-VCL Form Application*]. Окно представляет собой форму (в данном случае *Form1*), на которой происходит визуальное конструирование графического интерфейса разрабатываемого проекта. Размеры формы можно менять, перетаскивая мышью правую или нижнюю границу формы.

Окно *Конструктор форм* вызывается командой [*View-Toggle Form/Unit*].

Первоначально форма пуста, в дальнейшем, в процессе создания графического интерфейса проекта, на ней размещаются элементы управления.

Окно Программный код. С формой связан программный модуль, содержащий программные коды процедур. Для ввода и редактирования текста программы служит окно *Программный код* (в данном случае *Unit1.pas*), которое вызывается командой [*View-Toggle Form/Unit*].

Окно элементов управления. Справа внизу располагается *Окно элементов управления (Tool Palette)*, содержащее пиктограммы элементов управления. Стандартный

набор элементов управления размещается на вкладке *Standard* и включает в себя 16 классов объектов — это командная кнопка `TButton`, текстовое поле `TEdit`, надпись `TLabel` и т. д.

На вкладках *Addihional*, *Win32* и других располагаются дополнительные элементы управления: графическое поле `TImage`, список изображений `TImageList` и др.

Выбрав щелчком мыши в окне элементов управления *Tool Palette* нужный элемент, мы можем поместить его на форму проектируемого приложения. Процесс размещения на форме элементов управления аналогичен рисованию графических примитивов с использованием графического редактора.

Фактически мы размещаем на форме экземпляры определенных классов объектов. Например, выбрав класс объектов `TButton`, мы можем разместить на форме неограниченное количество экземпляров этого класса, т. е. командных кнопок `Button1`, `Button2`, `Button3` и т. д.

Окно Инспектор объектов. Слева внизу располагается окно *Инспектор объектов (Object Inspector)*. В верхней части окна размещается раскрывающийся список объектов, на вкладке *Свойства (Properties)* содержится список свойств, а на вкладке *События (Events)* — перечень событий, относящихся к выбранному объекту (форме или элементу управления на форме). На рисунке выбран объект `Button1` из класса `TButton`.

Вкладка *Свойства (Properties)* разделена на две колонки. В левой колонке находятся имена свойств, а в правой — их значения. Установленные по умолчанию значения могут быть изменены. Свойством объекта является количественная или качественная характеристика этого объекта (размеры, цвет, шрифт и др.). Для некоторых свойств предусмотрена возможность выбора из раскрывающегося списка значений, например, из списка можно выбрать значение цвета фона элемента управления (свойство `Color`).

Вкладка *События (Events)* также разделена на две колонки. В левой колонке находятся имена событий, а в правой можно ввести их значения.

Окно *Object Inspector* вызывается командой [*View-Object Inspector*].

Окно Менеджер проектов. Окно *Менеджер проектов (Project Manager)* располагается в верхнем правом углу. Оно отображает в виде иерархического каталога все составные части текущего проекта (в данном случае *Project1.exe*) и позволяет переключаться между ними.

Окно *Менеджер проектов* вызывается командой [*View-Project Manager*].

Окно *Дерево объектов*. В верхнем левом углу располагается окно *Дерево объектов (Structure)*, отображающее перечень объектов графического интерфейса проекта (форму и размещенные на форме элементы управления).

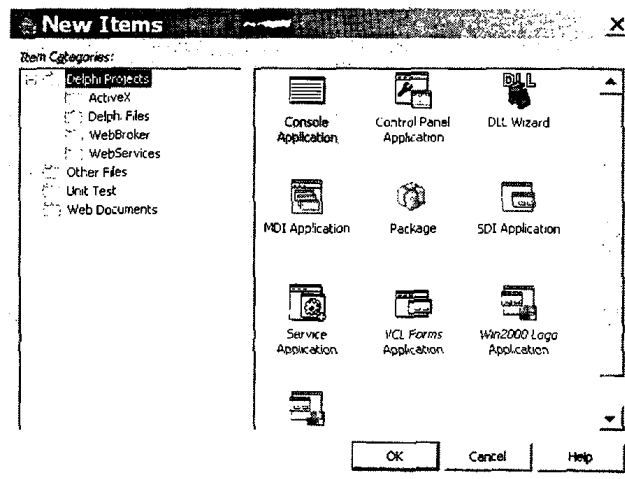
Окно *Дерево объектов (Structure)* вызывается командой [*View-Structure*].

Консольное приложение. В системе объектно-ориентированного программирования Turbo Delphi создадим консольное приложение, т. е. приложение, не имеющее графического интерфейса. Приложение должно выводить название языка программирования в окне командной строки и с помощью диалогового окна.



Проект «Консольное приложение» на языке программирования Turbo Delphi

1. Запустить систему программирования командой [*Пуск-Программы-Borland Developer Studio 2006-Turbo Delphi*].
2. Создать новый проект командой [*File-Other...*].
В появившемся диалоговом окне *New Items* выбрать тип приложения *Console Application*.



В окне *Программный код* появится заготовка программы. Ввести программный код пользователя между служебными словами *begin* и *end*.

Для вывода текста в окно программной строки использовать оператор `writeln()`.

Для вывода диалогового окна с текстом использовать функцию `MessageDlg()`, для использования которой необходимо подключить модуль диалогов с использованием инструкции `uses Dialogs`.

```
3. program Project1;
   uses
     Dialogs;
   {$APPTYPE CONSOLE}
   begin
     writeln('Program language Delphi');
     MessageDlg('Язык программирования Turbo Delphi',
       MtInformation, [mbOk], 0);
   end.
```

4. Сохранить программный код консольного приложения на жестком диске в папке ...\`HelloDelphi` командой [`File-Save As...`].

Теперь можно запустить консольное приложение.

5. Ввести команду [`Run-Run`].
6. В окне командной строки будет выведено (рис. 4.12):
`Program language Turbo Delphi`
 В диалоговом окне будет выведено (см. рис. 4.12):
`Язык программирования Turbo Delphi`

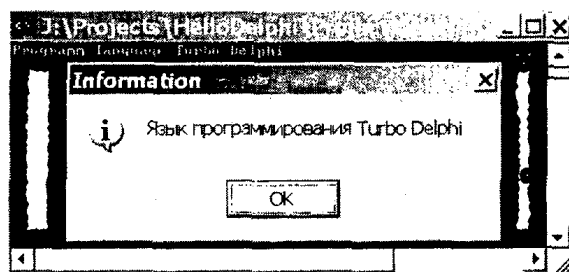



Рис. 4.12. Создание и запуск консольного приложения

Компьютерный практикум



TurboDelphi-CD 

- 4.3. Создать в системе программирования Turbo Delphi проект «Консольное приложение».



...\`ProjectsHelloDelphi`

4.6. Переменные

Тип переменной. Тип переменной определяется типом данных, которые могут быть значениями переменной. В объектно-ориентированных языках программирования переменные могут быть различных типов:

- в типах **Byte**, **Short** (**SmallInt** в Turbo Delphi), **Integer** (**Int** в Visual C# и Visual J#) и **Long** (**Int64** в Turbo Delphi) значениями являются целые числа;
- в типах **Single** (**float** в Visual C# и Visual J#), **Double** и **Decimal** (в Visual Basic .NET и Visual C#), значениями являются числа с плавающей запятой;
- в типах **Char** и **String** значениями являются символы и последовательности символов;
- в типе **Boolean** (**bool** в Visual C#) логическими значениями являются истина **True** или ложь **False**.

Различные типы переменных требуют для своего хранения в оперативной памяти компьютера различное количество ячеек (байтов) и могут принимать различные диапазоны значений.

Плакаты. Таблица 5 «Типы переменных»

Имя переменной. Имя каждой переменной (идентификатор) уникально и не может меняться в процессе выполнения программы. Имя переменной:

- должно начинаться с буквенного символа или с подчеркивания «_»;
- может содержать только буквенные символы, десятичные цифры и подчеркивания;
- должно содержать, по крайней мере, один буквенный или цифровой символ, если оно начинается с подчеркивания;
- не должно составлять более чем 1023 знака.



Рекомендуется для большей понятности текстов программ для программиста в имена переменных включать приставку, которая обозначает тип переменной. Тогда имена целочисленных переменных будут начинаться с приставок **byt**, **int** и **lng**, содержащих числа с плавающей запятой — **sng** и **dbl**, строковых **chr** и **str**, и логических — **bln**.

Объявление переменной. Важно, чтобы исполнитель программы (компьютер) «понимал», переменные какого типа

используются в программе. При объявлении переменной используется ключевое слово и указывается ее тип.

Область действия переменной. Область действия переменной, т. е. область, в которой она доступна для использования, может быть **локальной** или **глобальной**. Локальная переменная доступна только внутри процедуры или программного модуля, и к ней невозможно обращение из другой процедуры или модуля. Если переменная определена внутри процедуры, то она может быть вызвана только в этой процедуре, если она определена внутри программного модуля, то она может быть вызвана только в этом модуле.

К глобальным переменным может быть произведено обращение из всех программных модулей проекта.

Присваивание переменным значений. Переменная может получить или изменить значение с помощью **оператора присваивания**. В языках объектно-ориентированного программирования Visual Basic .NET, Visual C# и Visual J# в качестве оператора присваивания используется знак «равно» «=», а в языке Turbo Delphi — знаки «двоеточие» и «равно» «:=».

При выполнении оператора присваивания переменная, имя которой указано слева от знака равенства, получает значение, равное значению правой части. Это может быть, как в алгоритмических языках, выражение (арифметическое, строковое или логическое), а также значение свойства объекта или результат выполнения метода.

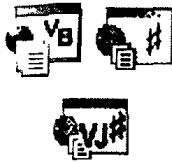
Проект «Переменные». Создать проект, который позволит продемонстрировать использование различных типов переменных, арифметических, строковых и логических выражений и операции присваивания. Каждый тип переменных разместить на отдельной вкладке.

В окне *Конструктор форм* с использованием панели *Область элементов* на форму поместим элементы управления, которые должны обеспечить взаимодействие проекта с пользователем.



Создание графического интерфейса проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

1. Для создания вкладок на форме элемент управления TabControl1 (PageControl1 в Turbo Delphi).



Создание вкладок на языках Visual Basic .NET, Visual C# и Visual J#

2. Выделить элемент управления PageControl1 и в окне Свойства у свойства TabPages активизировать значение (Коллекция).
3. В появившемся окне Редактор коллекции TabPages (рис. 4.13) создать три вкладки на панели инструментов, нажав три раза кнопку Добавить. В списке Свойства TabPages1: в свойстве Text ввести названия вкладок: Числовые, Строковые, Логические.

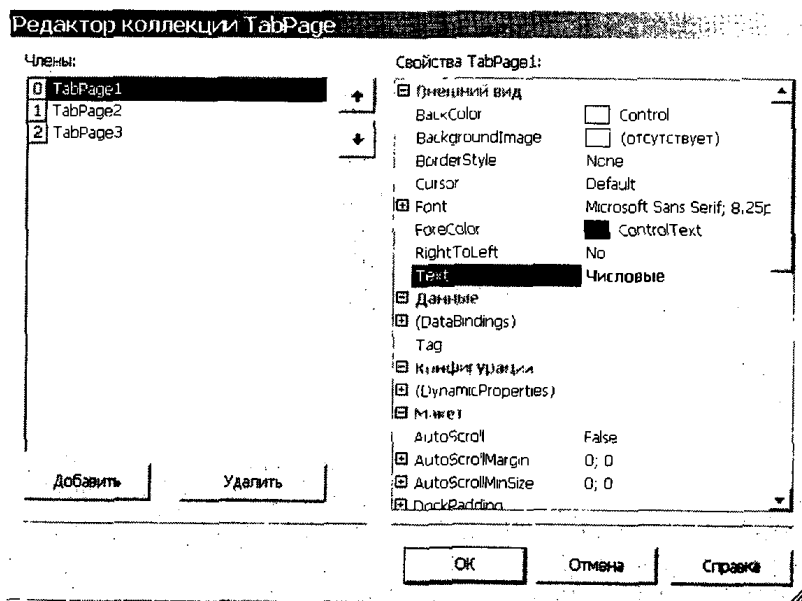


Рис. 4.13. Редактор коллекции вкладок



Создание вкладок на языке Turbo Delphi

2. Выделить элемент управления PageControl1, щелкнуть правой кнопкой мыши и в контекстном меню выбрать пункт New Page. Прodelать эти действия три раза.
3. В окне Object Inspector в свойстве Caption ввести названия вкладок: Числовые, Строковые, Логические.

4. На вкладке *TabPages1* (*TabSheets1* в Turbo Delphi) разместить:
 - для ввода чисел два текстовых поля `TextBox1` (`Edit1` в Turbo Delphi) и `TextBox2` (`Edit2` в Turbo Delphi);
 - для вывода чисел надписи `Label1`, `Label2` и `Label3`;
 - для создания обработчика события кнопку `Button1`;
 - для вывода поясняющих текстов пять надписей.
5. На вкладке *TabPages2* (*TabSheets2* в Turbo Delphi) разместить:
 - для ввода символа и строки два текстовых поля `TextBox3` (`Edit3` в Turbo Delphi) и `TextBox4` (`Edit4` в Turbo Delphi);
 - для вывода строки надпись `Label10`;
 - для создания обработчика события кнопку `Button2`;
 - для вывода поясняющих текстов три надписи.
6. На вкладке *TabPages3* (*TabSheets3* в Turbo Delphi) разместить:
 - для вывода логического значения надпись `Label20`;
 - для создания обработчика события кнопку `Button3`;
 - для вывода поясняющих текстов пять надписей.

На языке программирования Visual Basic .NET создадим обработчик события, реализующий вывод значений числовых переменных.



Создание программного кода на языке программирования Visual Basic .NET

```

7. Dim bytA, bytB As Byte, intC As Integer, sngD As
Single, dblE As Double
Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
  bytA = Val(TextBox1.Text)
  bytB = CByte(TextBox2.Text)
  intC = bytA / bytB
  sngD = bytA / bytB
  dblE = bytA / bytB
  Label1.Text = intC
  Label2.Text = sngD
  Label3.Text = dblE
End Sub

```

На языке программирования Visual C# создадим обработчик события, реализующий вывод значений строкового выражения.



Создание программного кода на языке программирования Visual C#

```

7. char chrS;
   String strS;
   private void button2_Click(object sender,
   System.EventArgs e)
   {chrS = Convert.ToChar(textBox3.Text);
     strS = textBox4.Text;
     label10.Text = chrS + strS;
   }

```

На языке программирования Turbo Delphi создадим обработчик события, реализующий вывод значений логического выражения.



Создание программного кода на языке программирования Turbo Delphi


```

7. var
   blnL1: boolean;
   blnL2: boolean;
   blnL3: boolean;
   procedure TForm1.Button3Click(Sender: TObject);
   begin
     blnL1 := 5 > 3;
     blnL2 := 2 * 2 = 5;
     blnL3 := blnL1 and blnL2;
     Label20.Caption := BoolToStr(blnL3, True);
   end;

```



Запуск проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

8. Ввести команду [*Отладка-Начать*] (на языке Turbo Delphi [*Run-Run*]) или щелкнуть по кнопке  на панели инструментов окна интегрированной среды разработки.
9. На вкладке *Числовые* в текстовые поля ввести числа и щелкнуть по кнопке *Вычислить bytA/bytB*. На надписи будут выведены значения числовых переменных различных типов (рис. 4.14).
На вкладке *Строковые* в текстовые поля ввести символ и строку и щелкнуть по кнопке *Вывести chrS+strS*. На надпись будет выведена строка (см. рис. 4.14).

На вкладке *Логические* щелкнуть по кнопке *Определить lblL1+lblL2*. На надпись будет выведено значение логического выражения (см. рис. 4.14).

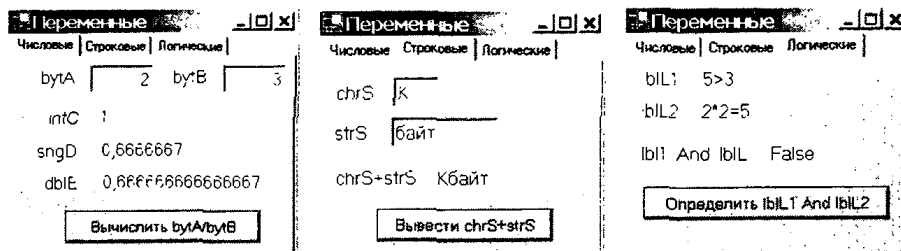


Рис. 4.14. Проект «Переменные»

Контрольные вопросы

1. В чем состоит разница между типом, именем и значением переменной?
2. Что происходит в процессе объявления переменной в оперативной памяти компьютера?

Компьютерный практикум



- 4.4. Создать проект «Переменные».

Windows-CD



...\informatika10...\Variables\

TurboDelphi-CD



...\Projects\Variables\

4.7. Графический интерфейс

Система объектно-ориентированного программирования позволяет визуализировать процесс создания графического интерфейса разрабатываемого проекта. Графический интер-

фейс необходим для реализации интерактивного диалога пользователя с исполняемым проектом.

Форма. Основой для создания графического интерфейса разрабатываемого проекта является **форма**, которая представляет собой окно, в котором размещаются элементы управления. Необходимо отметить, что графический интерфейс проекта может включать в себя несколько форм.



Форма — это объект, представляющий собой окно на экране, в котором размещаются элементы управления.

Элементы управления. Визуальное конструирование графического интерфейса проекта состоит в том, что на форму с помощью мыши помещаются и «рисуются» те или иные **элементы управления**.

Перечислим классы элементов управления, они имеют различное назначение в графическом интерфейсе проекта:

- текстовые поля `TextBox` (`Edit` в языке Turbo Delphi), надписи `Label` и списки `ListBox` и `ComboBox` для ввода и вывода данных;
- графические поля `PictureBox` (`Image` в языке Turbo Delphi) для вывода графики,
- командные кнопки `Button`, переключатели `RadioButton` и флажки `CheckBox` для организации интерактивного диалога пользователя с проектом;
- главное меню `MainMenu` для создания меню формы;
- панель инструментов `ToolBar` для создания панели инструментов формы;
- коллекция изображений `ImageList` для хранения изображений;
- диалоги `ColorDialog` и `FontDialog` для выбора цвета и шрифта;
- диалоги `OpenFileDialog` и `SaveFileDialog` для выбора файла при открытии и сохранении.



Плакаты. Таблицы 2–4 «Элементы управления: свойства, методы и события»

На форму может быть помещено несколько экземпляров одного класса элементов управления. Например, несколько

кнопок, каждая из которых обладает индивидуальными значениями свойств (надпись, размеры и др.).



Элементы управления — это объекты, являющиеся элементами графического интерфейса проекта и реагирующие на события, производимые пользователем или другими программными объектами.

Форма и каждый класс элементов управления обладают определенным набором свойств, методов и событий. Однако есть свойства, методы и события, которыми обладают формы и большинство элементов управления. С помощью свойств `Width` и `Height` устанавливается размер формы или элемента управления, с помощью метода `Show()` можно показать форму или элемент управления, на событие `Click` они реагируют.

Системы объектно-ориентированного программирования `Visual Basic .NET`, `Visual C#` и `Visual J#` имеют одинаковый набор элементов управления, который практически совпадает с набором элементов управления языка `Turbo Delphi`. Это позволяет конструировать практически одинаковые графические интерфейсы проектов во всех вышеперечисленных системах объектно-ориентированного программирования.

Автоматическая генерация кода элементов графического интерфейса. При размещении на форме элементов управления в системах объектно-ориентированного программирования производится автоматическая генерация программного кода. Устанавливаются значения свойств формы и элементов управления, которые определяют местоположение элемента графического интерфейса, его размеры, цвет, параметры шрифта и др.

В языках программирования `Visual Basic .NET`, `Visual C#` и `Visual J#` этот программный код сохраняется в файле программного кода формы в разделе *Код, автоматически созданный конструктором форм Windows*. В языке `Turbo Delphi` этот программный код сохраняется в отдельном файле `Unit1.dfm`, связанном с формой.

В качестве примера приведем автоматическую генерацию программного кода, возникающего при размещении на форме кнопки.

**Язык Visual Basic .NET:**

```

Me.Button1.BackColor = System.Drawing.
SystemColors.Control
Me.Button1.Font = New System.Drawing.Font
("Microsoft Sans Serif", 12.0)
Me.Button1.Location = New System.Drawing.
Point(104, 64)
Me.Button1.Name = "Button1"
Me.Button1.Size = New System.Drawing.Size
(152, 32)
Me.Button1.Text = "Вычислить"

```

Язык Visual C#:

```

this.button1.BackColor = System.Drawing.
SystemColors.Control;
this.button1.Location = new System.Drawing.
Point(88, 72);
this.button1.Name = "button1";
this.button1.Size = new System.Drawing.
Size(192, 40);
this.button1.Text = "Вычислить";

```

Язык Visual J#:

```

this.button1.set_BackColor(System.Drawing.
SystemColors.get_Control());
this.button1.set_Font(new System.Drawing.
Font("Microsoft Sans Serif", 12F);
this.button1.set_Location(new System.Drawing.
Point(104, 64));
this.button1.set_Name("button1");
this.button1.set_Size(new System.Drawing.
Size(132, 32));
this.button1.set_Text("Вычислить");

```

Язык Turbo Delphi:

```

object Button1: TButton
Left = 68
Top = 59
Width = 156
Height = 30
Font.Color = clWindowText
Font.Height = -16
Font.Name = 'MS Sans Serif'

```

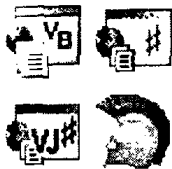
Графический интерфейс проектов на языках программирования Visual Basic .NET, Visual J#, Visual C# и Turbo Delphi будет практически одинаковым, так как в этих языках используется одинаковый набор элементов управления (табл. 4.2).

Таблица 4.2. Некоторые элементы управления в языках
Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

Элемент управления	Visual Basic .NET	Visual C#	Visual J#	Turbo Delphi
Текстовое поле	TextBox1	textBox1	textBox1	Edit1
Надпись	Label1	label1	label1	Label1
Список	Listbox1	listBox1	listBox1	Listbox1
Поле со списком	ComboBox1	comboBox1	comboBox1	ComboBox1
Счетчик	NumericUpDown1	numericUpDown1	numericUpDown1	UpDown1
Ползунок	TrackBar1	trackBar1	trackBar1	TrackBar1
Переключатель	RadioButton1	radioButton1	radioButton1	RadioButton1

Проект «Отметка». Создать проект, который в зависимости от количества ошибок, введенных с использованием различных элементов управления (списка, поля со списком, текстового поля, счетчика, ползунка и переключателей), выводит на надпись отметку.

В окне *Конструктор форм* с использованием панели *Область элементов* на форму поместим элементы управления, которые должны обеспечить взаимодействие проекта с пользователем.

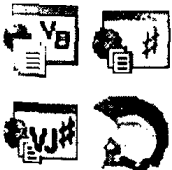


Создание графического интерфейса проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

1. Разместить на форме (рис. 4.15):

- для ввода количества ошибок список `ListBox1`, поле со списком `ComboBox1`, текстовое поле `TextBox1`, счетчик `NumericUpDown1`, ползунок `TrackBar1` и переключатели `RadioButton1`, `RadioButton2`, `RadioButton3`, `RadioButton4`, `RadioButton5`;
- для вывода отметки надпись `Label1`;
- для вывода поясняющих текстов надписи `Label2` и `Label3`.

Сделаем графический интерфейс более привлекательным.



Установка значений свойств объектов графического интерфейса на языках Visual Basic .NET, Visual C #, Visual J # и Turbo Delphi

2. Присвоить форме и элементам управления новые значения свойств с помощью диалогового окна *Свойства (Object Inspector)* в Turbo Delphi).

Для каждого элемента управления существует одно событие (событие по умолчанию), чаще всего возникающее для данного элемента управления (табл. 4.3). После двойного щелчка по элементу управления в редакторе программного кода открывается заготовка процедуры обработки такого события.

Таблица 4.3. События по умолчанию некоторых элементов управления в языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

Элемент управления	Visual Basic .NET	Visual C#	Visual J#	Turbo Delphi
Текстовое поле	TextChanged	TextChanged	TextChanged	Change
Надпись	Click	Click	Click	Click
Список	SelectedIndexChanged	SelectedIndexChanged	SelectedIndexChanged	Click
Поле со списком	SelectedIndexChanged	SelectedIndexChanged	SelectedIndexChanged	Change
Счетчик	ValueChanged	ValueChanged	ValueChanged	Click
Ползунок	Scroll	Scroll	Scroll	Change
Переключатель	CheckedChanged	CheckedChanged	CheckedChanged	Click

На языке программирования Visual Basic .NET объявим переменную и создадим обработчик события, реализующий вывод отметки на надпись в зависимости от количества ошибок, выбранных в списке. Создадим заготовку обработчика события щелчком по списку и введем программный код. Для преобразования количества ошибок, выбранного как элемент списка строкового типа, в целое число используем функцию `CByte()`. Для реализации выбора отметки используем оператор `Select Case`.



Создание программного кода на языке программирования Visual Basic .NET

3. Dim N As Byte

```
Private Sub ListBox1_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ListBox1.SelectedIndexChanged
    N = CByte(ListBox1.Text)
    Select Case N
        Case 0
            Label1.Text = "Отлично"
        Case 1
            Label1.Text = "Хорошо"
        Case 2
            Label1.Text = "Удовлетворительно"
        Case 3
            Label1.Text = "Плохо"
        Case Else
            Label1.Text = "Очень плохо"
    End Select
End Sub
```

На языке программирования Turbo Delphi объявим переменную и создадим обработчик события, реализующий вывод отметки на надпись в зависимости от количества ошибок, введенных в текстовое поле. Для преобразования введенного в текстовое поле количества ошибок строкового типа в целое число используем функцию `StrToInt()`. Для реализации выбора отметки используем оператор `Case`.



Создание программного кода на языке программирования Turbo Delphi

3. var

```
N: byte;
procedure TForm1.Edit1Change(Sender: TObject);
```

```

begin
N := StrToInt(Edit1.Text);
Case N Of
    0: Label1.Caption := 'Отлично';
    1: Label1.Caption := 'Хорошо';
    2: Label1.Caption := 'Удовлетворительно';
    3: Label1.Caption := 'Плохо';
    Else Label1.Caption := 'Очень плохо';
end;
end;

```

На языке программирования Visual C# объявим переменную и создадим обработчик события, реализующий вывод отметки на надпись в зависимости от количества ошибок, введенных с помощью ползунка. Так как свойство ползунка Value имеет тип `int`, объявим переменную такого же типа. Для реализации выбора отметки используем оператор `switch`.



Создание программного кода на языке программирования Visual C#

```

3. int n;
private void trackBar1_Scroll(object sender,
System.EventArgs e)
{n = trackBar1.Value;
    switch (n)
    {case 0:
        label1.Text = "Отлично";
        break;
        case 1:
        label1.Text = "Хорошо";
        break;
        case 2:
        label1.Text = "Удовлетворительно";
        break;
        case 3:
        label1.Text = "Плохо";
        break;
        default:
        label1.Text = "Очень плохо";
        break;
    }
}

```

На языке программирования Visual J# создадим обработчики событий, реализующие вывод отметки на надпись в за-

зависимости от количества ошибок, введенных с помощью переключателей. Для каждого переключателя создадим обработчик событий, выполняющийся при выборе этого переключателя.



Создание программного кода на языке программирования Visual J#

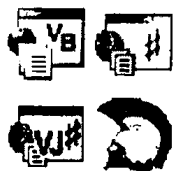
```

3. private void radioButton1_CheckedChanged (Object
sender, System.EventArgs e)
{
    label1.setText ("Отлично");
}
private void radioButton2_CheckedChanged (Object
sender, System.EventArgs e)
{
    label1.setText ("Хорошо");
}
private void radioButton3_CheckedChanged (Object
sender, System.EventArgs e)
{
    label1.setText ("Удовлетворительно");
}
private void radioButton4_CheckedChanged (Object
sender, System.EventArgs e)
{
    label1.setText ("Плохо");
}
private void radioButton5_CheckedChanged (Object
sender, System.EventArgs e)
{
    label1.setText ("Очень плохо");
}
}
    
```

Выполнение проекта. Загрузка проекта в систему объектно-ориентированного программирования производится путем активизации в папке проекта основного файла проекта:

- язык Visual Basic .NET — файл с расширением vbproj;
- язык Visual C# — файл с расширением csproj;
- язык Visual J# — файл с расширением vjsproj;
- язык Turbo Delphi — файл с расширением dpr.

Запустим проект, после этого система программирования переходит в режим выполнения проекта [*run*], в котором редактирование графического интерфейса или программного кода невозможно.



Запуск проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

4. Ввести команду [*Отладка-Начать*] (*[Run-Run]* на языке Turbo Delphi) или щелкнуть по кнопке ► на панели инструментов окна интегрированной среды разработки.
5. Ввести количество ошибок (например, 2) с использованием элементов управления:
 - списка;
 - поля со списком;
 - текстового поля;
 - счетчика;
 - ползунка;
 - одного из переключателей.

На надпись будет выведена отметка (в данном случае «Удовлетворительно») (см. рис. 4.15).

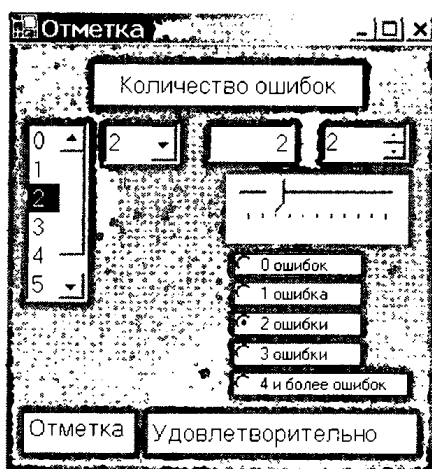


Рис. 4.15. Проект «Отметка»

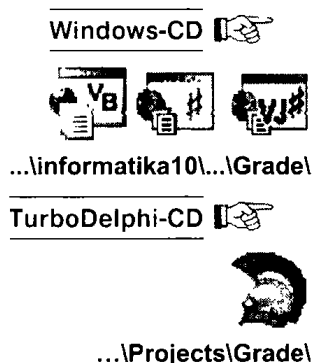
Контрольные вопросы

1. Какие элементы управления целесообразно использовать при конструировании графического интерфейса проекта, если необходимо: вводить и выводить данные? Организовать диалог с пользователем?

2. Что происходит в системе программирования после помещения на форму элемента управления?

Компьютерный практикум

4.5. Создать проект «Отметка».



4.8. Пространство имен .NET

Важный компонент платформы .NET — общая для всех языков программирования библиотека классов. Ее наличие позволяет разработчикам использовать единую систему программных объектов для языков программирования платформы .NET. Пространство имен .NET включает более 7000 имен и поэтому организовано в иерархическую систему имен (табл. 4.4).

Таблица 4.4. Некоторые наиболее значимые ветви пространства

Ветвь пространства имен	Содержание
System.Console	Методы и функции работы с консольным окном
System.Math	Математические методы и функции
System.Drawing	Графические методы и функции
System.IO	Методы и функции работы с файлами
System.Windows.Forms	Элементы графического интерфейса
Microsoft.VisualBasic	Методы и функции языка Visual Basic

Для использования в программном коде имени программного объекта необходимо указывать полный путь к нему в пространстве имен с использованием точечной нотации. Например, чтобы вывести в консольном окне текст, необходимо записать полный путь к функции `Write()` в пространстве имен `.NET`:

```
System.Console.Write("Текст")
```

Однако удобнее указать один раз в начале программного кода вызов ветви пространства имен `System`.

На языке программирования Visual Basic .NET:

```
Imports System
```

На языке программирования C#:

```
using System;
```

На языке программирования J#:

```
import System.*;
```

4.4.1. Платформа .NET Framework

Не все ветви пространства имен системой программирования загружаются по умолчанию. Например, для использования методов и функций языка Visual Basic в языках Visual C# и Visual J# необходимо подключить пространство имен `Microsoft.VisualBasic`. Для этого в проекте надо добавить ссылку на библиотеку `Microsoft.VisualBasic.dll`.

Одной из наиболее часто встречающихся операций в программировании является преобразование типов данных.

Проект «Функции преобразования типов». Создать проект, в котором число, вводимое в текстовое поле, преобразуется из строкового типа в числовой тип, а затем при выводе на надпись оно же преобразуется из числового типа в строковый.



Создание графического интерфейса проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

1. Разместить на форме (рис. 4.18):

- текстовое поле `TextBox1` (`Edit1` в Turbo Delphi) для ввода числа;
- три надписи `Label1`, `Label2` и `Label3` (две надписи в Visual C# и одну надпись в Visual J# и Turbo Delphi) для вывода числа;

- кнопку `Button1` для создания процедуры-обработчика события.

Преобразование типов данных на языке Visual Basic .NET возможно с помощью функций трех различных типов:

- во-первых, это функции `Val()`, `Str()` и др., которые использовались и в предыдущих версиях языка программирования Basic;
- во-вторых, это функции `Cdbl()`, `CStr()` и др., которые используются в последних версиях языка программирования Visual Basic;
- в-третьих, это функции `ToDouble()`, `ToString()` и др., которые используются на платформе .NET и принадлежат ветви системы имен `System.Convert`.

В программном коде используем функции преобразования типов данных всех трех типов.



Создание программного кода на языке программирования Visual Basic .NET

2. Dim A, B, C As Double

```
Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
'Преобразование из строкового типа в числовой
A = Val(TextBox1.Text)
B = System.Convert.ToDouble(TextBox1.Text)
C = Cdbl(TextBox1.Text)
'Преобразование из числового типа в строковый
Label1.Text = Str(A)
Label2.Text = System.Convert.ToString(B)
Label3.Text = CStr(C)
End Sub
```

Преобразование типов данных на языке Visual C# возможно с помощью функций двух различных типов:

- во-первых, это функции `Val()`, `Str()` и др., которые используются в языке Visual Basic и принадлежат ветви системы имен `Microsoft.VisualBasic.Conversion`;
- во-вторых, это функции `ToDouble()`, `ToString()` и др., которые используются на платформе .NET и принадлежат ветви системы имен `System.Convert`.

В программном коде используем функции преобразования типов данных двух типов.



Создание программного кода на языке программирования Visual C#

На языке Visual C# для преобразования типов данных с использованием функций `Val()`, `Str()` необходимо подключить и использовать пространство имен `Microsoft.VisualBasic`.

2.1. Ввести команду [Проект-Добавить ссылку...].

В появившемся диалоговом окне *Add Reference* (рис. 4.16) на вкладке `.NET` из списка выбрать имя компонента *Microsoft.VisualBasic.NET Runtime* и щелкнуть по кнопке *Выбрать*.

Выбранный компонент добавится в поле *Выбранные компоненты*:

Щелкнуть по кнопке *ОК*.

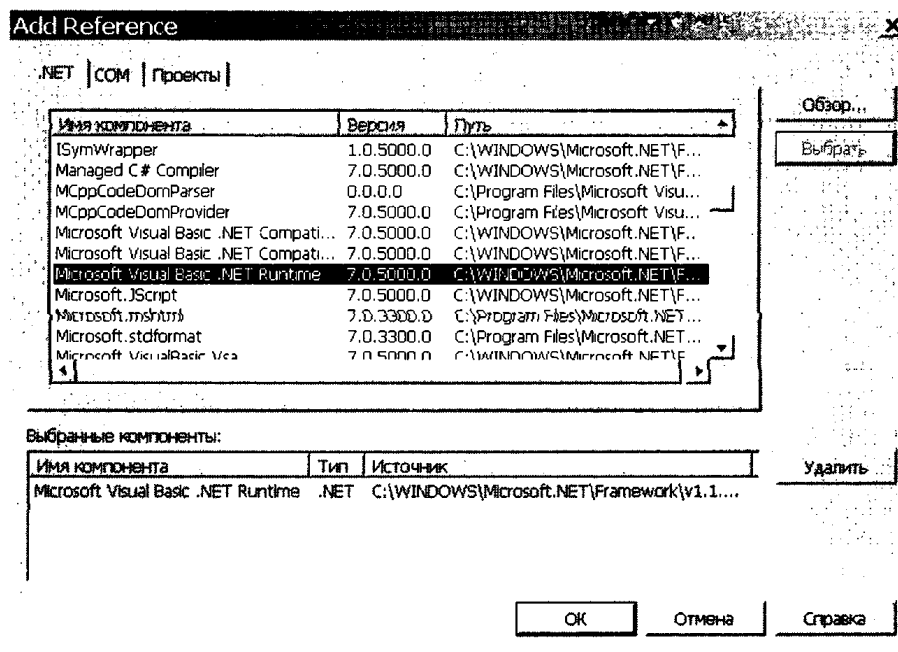


Рис. 4.16. Диалоговое окно выбора ветвей пространства имен

2.2. В *Обзревателе решений* в проекте *convert* в разделе ссылок *References* добавится ссылка *Microsoft.VisualBasic* на соответствующее пространство имен (рис. 4.17).

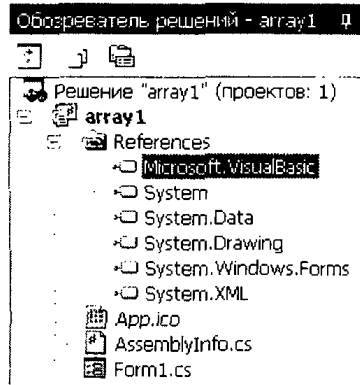


Рис. 4.17. Ссылки на пространства имен

Теперь можно добавить ссылку на пространство имен в обработчик событий.

```
2.3. double A, B;
private void button1_Click(object sender,
System.EventArgs e)
{ //Преобразование из строкового типа в числовой
A = Microsoft.VisualBasic.Conversion.Val
(textBox1.Text);
B = System.Convert.ToDouble(textBox1.Text);
//Преобразование из числового типа в строковый
label1.Text = Microsoft.VisualBasic.Conversion.
Str(A);
label2.Text = System.Convert.ToString(B);
}
```

Преобразование типов данных на языке Visual J# возможно с помощью функций `ToDouble()`, `ToString()` и др., которые используются на платформе .NET и принадлежат ветви системы имен `System.Convert`.



Создание программного кода на языке программирования Visual J#

```
2. double A;
private void button1_Click (Object sender, System.
EventArgs e)
{ //Преобразование из строкового типа в числовой
A = System.Convert.ToDouble(textBox1.get_Text());
```

```
//Преобразование из числового типа в строковый
labell.set_Text(System.Convert.ToString(A));
}
```

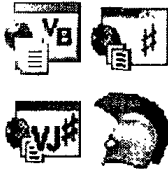
Преобразование типов данных на языке Turbo Delphi возможно с помощью функций `StrToFloat()`, `FloatToStr()` и др.



Создание программного кода на языке программирования Turbo Delphi

2. var

```
A: double;
procedure TForm1.Button1Click(Sender: TObject);
begin
//Преобразование из строкового типа в числовой
A := StrToFloat(Edit1.Text);
//Преобразование из числового типа в строковый
Label1.Caption := FloatToStr(A);
end;
```



Запуск проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

3. Запустить проект на выполнение и ввести в текстовое поле число (например, 3,14).

На надписи будет выведено это же число, подвергнутое двойному преобразованию с использованием функций трех типов (см. рис. 4.18).

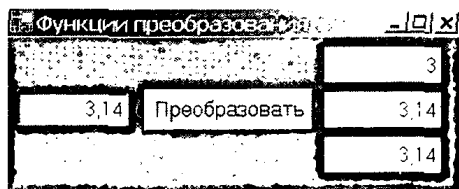


Рис. 4.18. Проект «Функции преобразования типов» на языке Visual Basic .NET

Компьютерный практикум

4.6. Создать проект «Функции преобразования типов».

Windows-CD 



...\informatika10\...\Convert\

TurboDelphi-CD 



...\Projects\Convert\

Контрольные вопросы

1. Почему в проекте «Функции преобразования типов» на первую надпись выводится целое число, а не число с дробной частью?

4.9. Процедуры и функции

4.9.1. Процедуры

При разработке сложного алгоритма целесообразно выделить в нем последовательности действий, которые могут многократно вызываться из основного алгоритма. Такие алгоритмы называются **вспомогательными** и в алгоритмических языках программирования реализуются в форме **подпрограмм**, которые вызываются из основной программы.

В объектно-ориентированных языках программирования Visual Basic .NET и Turbo Delphi вспомогательные алгоритмы реализуются с помощью **процедур**, а в языках Visual C# и Visual J# — с помощью **методов**.

Каждой подпрограмме дается уникальное название — имя процедуры или метода и устанавливается список параметров. Параметры разделяются между собой запятой и содержат определение типов.

Любой параметр может использоваться как входной параметр, т. е. после вызова процедуры (метода) он получает значение из вызывающей процедуры. Процедуры не возвращают значений (в методах об этом свидетельствует ключевое слово **void**).

Запуск процедур (методов) не связывается с какими-либо событиями, а реализуется путем вызова по имени из других процедур.



Процедура представляет собой подпрограмму, которая не возвращает значения и начинается выполняться после ее вызова из другой процедуры.

Процедура Main(). В языке Visual Basic .NET процедура Main() (метод Main в языках Visual C# и Visual J#) является «стартовой точкой» проекта, т. е. первой процедурой, к которой осуществляется доступ при запуске проекта. В этой процедуре можно объявить глобальные переменные проекта, а также определить, какая форма загружается первой при запуске проекта.

4.14. Модульный принцип построения решений (групп) и проектов

Передача параметров по значению и по ссылке. В процедурах в языках программирования Visual Basic .NET и Turbo Delphi, а также в методах на языках Visual C# и Visual J# после имени в скобках указывается список типов и имен параметров, т. е. переменных. При вызове процедуры ее параметрам должны быть переданы значения.

Каждая переменная имеет значение и уникальный адрес, соответствующий положению переменной в оперативной памяти компьютера. Объектно-ориентированные языки программирования позволяют передавать в процедуру как значение переменной (**передача по значению**), так и ее адрес в оперативной памяти (**передача по ссылке**).

При передаче по значению в процедуру передается только копия переменной, т. е. процедура не получает доступ к переменной в памяти и не может изменять ее значение. В языках объектно-ориентированного программирования передача по значению осуществляется по умолчанию. Тем не менее в языке Visual Basic .NET для передачи по значению необходимо перед параметром добавить ключевое слово **ByVal** (от англ. by value — по значению).



При передаче параметров **по значению** вызываемая процедура не может изменить значение переменной в вызывающей процедуре.

При передаче переменной по ссылке процедуре передается ссылка на ее адрес в оперативной памяти, таким образом, процедура получает доступ к переменной в памяти и может ее изменить. Чтобы передать переменную по ссылке, необходимо перед соответствующим параметром в списке параметров добавить ключевое слово. В языке Visual Basic .NET это ключевое слово **ByRef** (от англ. by reference — по ссылке), в языке Visual C# — ключевое слово **ref**, в языке Turbo Delphi — ключевое слово **var**.



При передаче параметров по ссылке вызываемая процедура может изменить значение переменной в вызывающей процедуре.



При вызове процедур ключевые слова, определяющие способ передачи параметров, не указываются.

Проект «Передача по ссылке и по значению». Создать проект, в котором:

- в вызываемой процедуре один параметр передается по ссылке, а другой — по значению и значения переменных внутри процедуры изменяются;
- в вызывающей процедуре-обработчике события переменным присваиваются начальные значения, вызывается процедура, а затем значения переменных выводятся на надписи.



Создание графического интерфейса проекта на языках Visual Basic .NET, Visual C# и Turbo Delphi



1. Разместить на форме (рис. 4.19):

- текстовые поля `TextBox1` и `TextBox2` для ввода начального значения переменных;
- надписи `Label1` и `Label2` для вывода конечного значения переменных;
- кнопку `Button1` для создания процедуры-обработчика события;
- четыре надписи для вывода поясняющих текстов.

В программных кодах процедур на языках Visual Basic .NET и Turbo Delphi, а также программных кодах метода на языке Visual C# первый параметр *X* передается по ссылке, а второй параметр *Y* передается по значению.

Создадим программный код проекта, в котором:

- объявим две переменные;
- создадим процедуру, умножающую значения переменных, передаваемых по ссылке и по значению, на коэффициент 5;
- создадим обработчик события, реализующий присваивание переменным начальных значений, вызов первой процедуры и осуществляющий вывод конечных значений переменных на надписи.



Создание программного кода на языке программирования Visual Basic .NET

2. Объявить переменные и создать код вызываемой процедуры:

```
Dim RefA, ValB As Byte
Sub RefVal(ByRef RefA, ByVal ValB)
  RefA = RefA * 5
  ValB = ValB * 5
End Sub
```

3. Создать код вызывающей процедуры-обработчика события:

```
Private Sub Button1_Click(ByVal sender As System.
Object, ByVal e As System.EventArgs) Handles But-
ton1.Click
  RefA = Val(TextBox1.Text)
  ValB = Val(TextBox2.Text)
  RefVal(RefA, ValB)
  Label1.Text = Str(RefA)
  Label2.Text = Str(ValB)
End Sub
```



Создание программного кода на языке программирования Visual C#

2. Объявить переменные и создать код вызываемой процедуры:

```
int RefA;
int ValB;
void RefVal (ref int RefA, int ValB)
{RefA = RefA*5;
  ValB = ValB*5;
}
```

3. Создать код вызывающей процедуры-обработчика события:

```
private void button1_Click(object sender, System.
EventArgs e)
```

```

(RefA = Convert.ToInt32(textBox1.Text);
 ValB = Convert.ToInt32(textBox2.Text);
 RefVal(ref RefA, ValB);
 label1.Text = Convert.ToString(RefA);
 label2.Text = Convert.ToString(ValB);
)

```



Создание программного кода на языке программирования Turbo Delphi

2. Объявить переменные и создать код вызываемой процедуры:

```

var
  RefA, ValB: byte;
procedure RefVal(var RefA: byte; ValB: byte);
begin
  RefA := RefA * 5;
  ValB := ValB * 5;
end;

```

3. Создать код вызывающей процедуры-обработчика события:

```

procedure TForm1.Button1Click(Sender: TObject);
begin
  RefA := StrToInt(Edit1.Text);
  ValB := StrToInt(Edit2.Text);
  RefVal(RefA, ValB);
  Label1.Caption := IntToStr(RefA);
  Label2.Caption := IntToStr(ValB);
end;

```



Запуск проекта на языках Visual Basic .NET, Visual C# и Turbo Delphi

4. Запустить проект на выполнение и ввести в текстовые поля начальные значения переменных (например, 3). На надписи будут выведены конечные значения переменной, передаваемой по ссылке (в данном случае, 15), и переменной, передаваемой по значению (в данном случае 3) (см. рис. 4.19).

Что и требовалось показать!

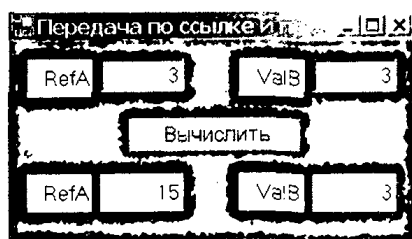



Рис. 4.19. Проект «Передача по ссылке и по значению»

Контрольные вопросы

1. В чем состоит разница при передаче значений по значению и по ссылке?


Компьютерный практикум

- 4.7. Создать проект «Передача по ссылке и по значению».

Windows-CD 



...\\informatika10\...\RefVal\


TurboDelphi-CD 



...\\Projects\RefVal\

4.9.2. Функции

Функции являются подпрограммами, которые возвращают значения и поэтому могут использоваться в выражениях. Наиболее широко используются встроенные функции: математические, преобразования типов данных, обработки строк, даты и времени и др.

 Плакаты. Таблица 6 «Встроенные функции (методы)»

Программист может создать свои функции. В языках Visual Basic .NET и Turbo Delphi функция начинается с ключевого слова **function**, затем следует имя функции и в

скобках список ее параметров (аргументов). Тип возвращаемого функцией значения указывается после списка параметров.

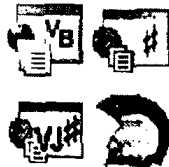
В языках Visual C# и Visual J# функции реализуются с помощью методов и начинаются с ключевого слова, определяющего тип возвращаемого методом значения (**int**, **double** и т. д.). Затем следует имя метода и в скобках список его параметров (аргументов).

Для определения возвращаемого функцией (методом) значения может использоваться как имя функции, так и переменная *result*, которой в программном коде присваивается значение функции. Для возврата значения функции с помощью переменной *result* используется ключевое слово **return**.



Функция представляет собой подпрограмму, которая возвращает значение и может входить в состав выражений.

Проект «Функция». Создать проект, в котором определяется функция (например, умножения двух чисел $F = x * y$), выводится на надпись значение выражения, в которое входит функция (например, $\sqrt{F^2 + F}$).



Создание графического интерфейса проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

1. Разместить на форме (рис. 4.20):
 - для ввода значений аргументов функции два текстовых поля `TextBox1` и `TextBox2` (`Edit1` и `Edit2` в Turbo Delphi);
 - для вывода значения функции надпись `Label1`;
 - для создания обработчика события кнопку `Button1`.



Создание программного кода на языке программирования Visual Basic .NET

2. Создать код вызываемой функции:


```
Function F(ByVal X, ByVal Y) As Integer
    F = X * Y
End Function
```
3. Импортировать ветвь пространства имен `System.Math`, которая необходима для использования математических

функций, объявить переменные и создать код вызывающей процедуры-обработчика события:

```
Imports System.Math
...
Dim X, Y As Integer
Private Sub Button1_Click(ByVal sender As System.
Object, ByVal e As System.EventArgs) Handles
Button1.Click
X = Val(TextBox1.Text)
Y = Val(TextBox2.Text)
Label1.Text = Str(Sqrt(Pow(F(X, Y), 2) + F(X, Y)))
End Sub
```



Создание программного кода на языке программирования Visual C#

2. Создать код вызываемой функции:

```
public int F(int X, int Y)
{int result;
result = X * Y;
return result;
}
```

3. Объявить переменные и создать код вызывающей процедуры-обработчика события:

```
int X, Y;
private void button1_Click(object sender, System.
EventArgs e)
{X = Convert.ToInt32(textBox1.Text);
Y = Convert.ToInt32(textBox2.Text);
label1.Text = Convert.ToString(Math.Sqrt(Math.
Pow(F(X, Y), 2) + F(X, Y)));
}
```



Создание программного кода на языке программирования Visual J#

2. Создать код вызываемой функции:

```
public int F(int X, int Y)
{int result;
result = X * Y;
return result;
}
```

3. Объявить переменные и создать код вызывающей процедуры-обработчика события:

```
int X, Y;
private void button1_Click (Object sender, System.
EventArgs e)
```

```

(X = System.Convert.ToInt32(textBox1.get_Text());
Y = System.Convert.ToInt32(textBox2.get_Text());
label1.set_Text(System.Convert.ToString(System.
Math.Sqrt(System.Math.Pow(F(X, Y),2) + F(X, Y)))));
}

```



Создание программного кода на языке программирования Turbo Delphi

2. Создать код вызываемой процедуры:


```

function F(X,Y:integer): int64;
begin
  F := X*Y;
end;

```
3. Импортировать модуль Math, который необходим для использования математических функций, объявить переменные и создать код вызывающей процедуры-обработчика события:


```

uses Math;
...
var
  X, Y: integer;
procedure TForm1.Button1Click(Sender: TObject);
begin
  X := StrToInt(Edit1.Text);
  Y := StrToInt(Edit2.Text);
  Label1.Caption := FloatToStr(Sqrt(Power(F(X, Y),
  2) + F(X, Y)));
end;

```



Запуск проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

4. Запустить проект на выполнение и ввести в текстовые поля значения переменных (например, 2 и 3). На надписи будет выведено вычисленное значение выражения (см. рис. 4.20).

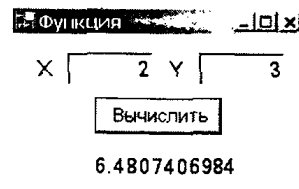


Рис. 4.20. Проект «Функция»


Контрольные вопросы

1. В чем состоит различие между процедурами и функциями?
2. Может ли входить в состав выражения процедура? Функция?

Компьютерный практикум




4.8. Создать проект «Функция».

Windows-CD 



...\informatika10...\Function\

TurboDelphi-CD 



...\Projects\Function\

4.10. Итерация и рекурсия

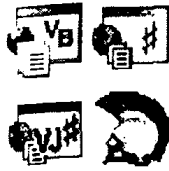
Функция (метод в языках программирования Visual C# и Visual J#), которая обращается сама к себе, называется **рекурсивной функцией**. Одну и ту же задачу можно часто решить двумя способами: с помощью рекурсии (с использованием рекурсивной функции) и с помощью итерации (с использованием цикла). Однако использование рекурсии позволяет экономить оперативную память компьютера.

Вычисление факториала числа — это классический пример, на котором можно показать использование итерации и рекурсии. Факториал целого положительного числа N — это произведение целых чисел от 1 до N , обозначается как $N!$:

$$N! = 1 \times 2 \times 3 \times \dots \times (N-1) \times N.$$

Проект «Факториал (итерация)». Создать проект, в котором факториал числа вычисляется с использованием цикла.

Будем вводить число в текстовое поле и выводить шаги вычисления факториала этого числа в список.



Создание графического интерфейса проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

1. Разместить на форме (рис. 4.21):

- текстовое поле TextBox1 для ввода числа;
- список ListBox1 для вывода шагов вычисления факториала в качестве элементов списка;
- надписи для вывода поясняющих текстов.

Для вычисления факториала числа (переменная N) воспользуемся циклом со счетчиком, в котором счетчик цикла (переменная I) будет принимать значения от минимального (1) до максимального (значение переменной N). В теле цикла будем последовательно в каждом шаге цикла умножать факториал (переменная F) на значение счетчика (переменная I) и выводить их значения в список.



Создание программного кода на языке программирования Visual Basic .NET

2. Объявить переменные, щелчком по текстовому полю создать заготовку процедуры-обработчика события и ввести программный код:

```
Dim N, I As Byte, F As Long
Private Sub TextBox1_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles TextBox1.TextChanged
ListBox1.Items.Clear()
N = Val(TextBox1.Text)
F = 1
For I = 1 To N
F = F * I
ListBox1.Items.Add(Str(I) & "! =" & Str(F))
Next I
End Sub
```



Запуск проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

3. Запустить проект на выполнение и ввести в текстовое поле число (например, 20).

В списке будут выведены шаги вычисления факториала введенного числа (см. рис. 4.21).

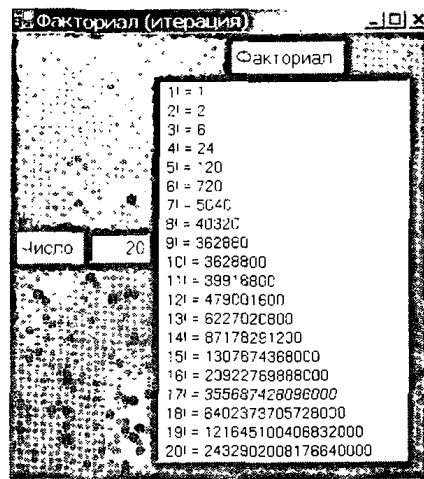
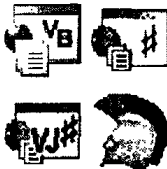


Рис. 4.21. Проект «Факториал (итерация)»

Проект «Факториал (рекурсия)». Создать проект, в котором факториал числа вычисляется с использованием рекурсивной функции.

Будем вводить число в текстовое поле и выводить шаги вычисления факториала этого числа в список.



Создание графического интерфейса проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

1. Разместить на форме (рис. 4.22):

- текстовое поле Edit1 для ввода числа;
- список ListBox1 для вывода шагов вычисления факториала в качестве элементов списка;
- надписи для вывода поясняющих текстов.

Формулу для вычисления факториала можно записать следующим образом:

$$N! = N \times (N-1) \times (N-2) \times \dots \times 2 \times 1 = N \times (N-1)!$$

Следовательно, факториал числа $N!$ равен произведению числа N на факториал числа $(N-1)!$. В свою очередь, факториал числа $(N-1)!$ — это произведение числа $N-1$ на факториал числа $(N-2)!$ и т. д.

Таким образом, если вычисление факториала числа $N!$ реализовать как функцию, то в теле этой функции должен быть вызов функции вычисления факториала числа $(N-1)!$, т. е. получим рекурсивную функцию, так как она вызывает сама себя.



Создание программного кода на языке программирования Turbo Delphi

Для вычисления факториала числа создадим рекурсивную функцию `Factorial(N: byte)`, аргументом которой является число N . В зависимости от значения числа N с использованием оператора ветвления `if...then...else` будем вычислять значение функции. Если функция вызывается с аргументом, равным 1, то она возвращает значение 1, в противном случае она обращается к самой себе и возвращает произведение $N * \text{Factorial}(N-1)$.

2. Создать рекурсивную функцию, реализующую вычисление факториала:

```
function Factorial(N:byte): int64;
begin
  if N = 1
  then Factorial := 1
  else Factorial := N*Factorial(N-1);
end;
```

Для вывода шагов вычисления факториала воспользуемся циклом со счетчиком, в котором счетчик цикла (переменная I) будет принимать значения от минимального (1) до максимального (значение переменной N). В теле цикла будем вызывать рекурсивную функцию, вычисляющую факториал, и выводить значения счетчика цикла и функции в список.

3. Объявить переменные, щелчком по текстовому полю создать заготовку процедуры-обработчика события и ввести программный код:

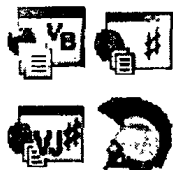
```
var
  I: byte;
  N: byte;

procedure TForm1.Edit1Change(Sender: TObject);
begin
  N := StrToInt(Edit1.Text);
  ListBox1.Items.Clear;
  For I := 1 To N Do
```

```

begin
  ListBox1.Items.Add(IntToStr(I) + '! =' +
    IntToStr(Factorial(I)));
end;
end;

```



Запуск проекта на языках Visual Basic
.NET, Visual C#, Visual J# и Turbo Delphi

4. Запустить проект на выполнение и ввести в текстовое поле число (например, 20).
В списке будут выведены шаги вычисления факториала введенного числа (см. рис. 4.22).

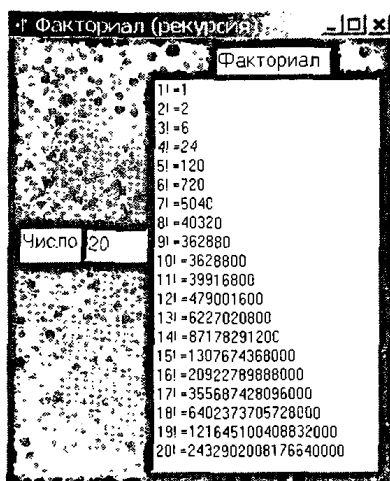



Рис. 4.22. Проект «Факториал (рекурсия)»

Контрольные вопросы

1. Почему не удастся в вышерассмотренных проектах на языках объектно-ориентированного программирования вычислить факториал числа, большего 20?

Компьютерный практикум

Windows-CD 

4.9. Создать проект «Факториал (итерация)».



...\informatika10\...\Factorial1\

TurboDelphi-CD 




...\Projects\Factorial1\

4.10. Создать проект «Факториал (рекурсия)».



\informatika10\...\Factorial2\

TurboDelphi-CD 



...\Projects\Factorial2\

4.11. Делегаты

Делегаты. Делегаты используются в среде .NET Framework для построения механизмов обработки событий. Когда возникает событие, делегат вызывает соответствующий обработчик события.

Делегат по умолчанию автоматически связывается с одним обработчиком события. В программном коде языков .NET это реализуется по-разному. В языке Visual Basic .NET непосредственно в заготовке обработчика события делегат указывается после служебного слова **Handles**, например обработчик события `Button1_Click` вызывается делегатом `Button1.Click`.

Делегат события может вызывать несколько различных обработчиков событий. Групповой делегат обеспечивает уведомление «один делегат ко многим обработчикам событий», т. е. последовательно вызывает несколько обработчиков событий из списка вызова.

Обработчик событий может вызываться делегатами различных событий, т. е. обеспечивается уведомление «многие делегаты к одному обработчику события». В этом случае связь между делегатами и обработчиками событий устанавливается программистом вручную. Например, событие, возникающее при нажатии кнопки, и событие, возникающее при щелчке по команде меню, могут использовать делегаты, которые затем вызовут один и тот же обработчик событий для одинаковой обработки этих разных событий.

Делегаты используются в среде .NET Framework для вызова процедур-обработчиков событий. Механизм действия делегатов следующий: событие — создание делегата события — вызов процедур-обработчиков событий.

По умолчанию событие создает «свой» делегат, который автоматически вызывает «свой» обработчик события. Например, щелчок по кнопке `Button1` соответствует событию `Click`, которое создает делегат `Button1.Click`, вызывающий процедуру-обработчик событий `Button1_Click`.

При создании процедуры-обработчика события (метода) делегат автоматически записывается:

- на языке Visual Basic .NET в строке имени процедуры после ключевого слова **Handles** (англ. «управляемый»);
- на языках Visual C# и Visual J# в разделе программного кода Windows Form Designer generated code, после ключевого слова **this** (англ. «этот»).

Язык Visual Basic .NET:

```
Private Sub Button1_Click(ByVal sender As
    System.Object, ByVal e As System.EventArgs)
    Handles Button1.Click
```



Язык Visual C#:

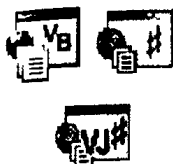
```
this.button1.Click += new System.
    EventHandler(this.button1_Click);
```

Язык Visual J#:

```
this.button1.add_Click(new System.
    EventHandler(this.button1_Click));
```

Проект «Делегаты». Создать проект, в котором один делегат вызывает два обработчика событий, а обработчик события вызывается тремя разными делегатами. Делегаты событий «щелчок по кнопке», «щелчок по пункту меню» и «щелчок

по кнопке на панели инструментов» вызывают два обработчика событий «изменение текста», которые по умолчанию создаются для списка и текстового поля со списком. В результате после щелчка по кнопке, пункту меню и кнопке на панели инструментов должен выводиться текст «Событие произошло» в качестве элемента списка в поле списка и в текстовом поле со списком.



Создание графического интерфейса проекта на языках Visual Basic .NET, Visual C# и Visual J#

1. Разместить на форме (рис. 4.23):

- список `ListBox1` и поле со списком `ComboBox1` для вывода текста «Событие произошло» в качестве элементов списка;
- кнопку `Button1`, меню `MenuItem1` и панель инструментов `ToolBar1` для создания делегатов событий.

Щелчком по списку создать заготовку по умолчанию обработчика события `ListBox1_SelectedIndexChanged`.

Щелчком по текстовому полю со списком создать заготовку по умолчанию обработчика события `ComboBox1_SelectedIndexChanged`.

Введем код обработчиков событий, который выводит текст «Событие произошло» в качестве элемента списка в поле списка и в текстовое поле со списком.

В языке Visual Basic .NET делегаты вводятся непосредственно в заготовки обработчиков событий. Для этого после ключевого слова **Handles** необходимо ввести делегаты событий «щелчок по кнопке» `Button1.Click`, «щелчок по пункту меню» `MenuItem1.Click` и «щелчок по кнопке на панели инструментов» `ToolBar1.Click`.



Создание программного кода процедур-обработчиков событий и делегатов на языке программирования Visual Basic .NET

```
2. Private Sub ListBox1_SelectedIndexChanged(ByVal
sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click, MenuItem1.Click,
ToolBar1.Click
ListBox1.Items.Add("Событие произошло")
End Sub
```



```

Private Sub ComboBox1_SelectedIndexChanged(ByVal
sender As System.Object, ByVal e As System.Event-
Args) Handles Button1.Click, MenuItem1.Click,
ToolBar1.Click
ComboBox1.Items.Add("Событие произошло")
End Sub

```

В языках Visual C# и Visual J# делегаты вводятся в разделе программного кода Windows Form Designer generated в области описания соответствующего элемента управления.



Создание делегатов на языке программирования Visual C#

- Ввести делегаты события «щелчок по кнопке» button1.Click для обработчиков событий listBox1_SelectedIndexChanged и comboBox1_SelectedIndexChanged:

```

this.button1.Click += new System.EventHandler
(this.listBox1_SelectedIndexChanged);
this.button1.Click += new System.EventHandler
(this.comboBox1_SelectedIndexChanged);

```
- Ввести делегаты для события «щелчок по пункту меню» menuItem1.Click для обработчиков событий listBox1_SelectedIndexChanged и comboBox1_SelectedIndexChanged:

```

this.menuItem1.Click += new System.EventHandler
(this.listBox1_SelectedIndexChanged);
this.menuItem1.Click += new
System.EventHandler(this.comboBox1_
SelectedIndexChanged);

```



Создание делегатов на языке программирования Visual J#

- В разделе программного кода Windows Form Designer generated code ввести делегаты события «щелчок по кнопке» button1.Click для обработчиков событий listBox1_SelectedIndexChanged и comboBox1_SelectedIndexChanged:

```

this.button1.add_Click(new System.EventHandler
(this.listBox1_SelectedIndexChanged));
this.button1.add_Click(new System.EventHandler
(this.comboBox1_SelectedIndexChanged));

```
- Ввести делегаты для события «щелчок по пункту меню» menuItem1.add_Click:

```

menuItem1.add_Click:

```

```
this.menuItem1.add_Click(new System.EventHandler
(this.listBox1_SelectedIndexChanged));
this.menuItem1.add_Click(new System.EventHandler
(this.comboBox1_SelectedIndexChanged));
```



Запуск проекта на языках Visual Basic .NET, Visual C# и Visual J#

4. Запустить проект на выполнение и щелкнуть по кнопке, пункту меню и кнопке панели инструментов.

В списке и поле со списком будут выведены тексты «Событие произошло» (см. рис. 4.23).

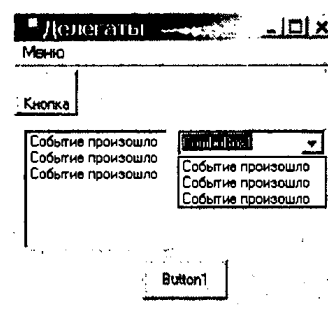


Рис. 4.23. Проект «Делегаты»

Контрольные вопросы

1. Может ли один и тот же делегат вызывать различные обработчики событий? Может ли один и тот же обработчик события вызываться различными делегатами?

Компьютерный практикум



Windows-CD

- 4.11. Создать проект «Делегаты».



...\\informatika10\...\Delegate\

4.12. Алгоритмы перевода чисел и их кодирование на языках объектно-ориентированного программирования

4.12.1. Алгоритм перевода целых чисел

2.8.1. Перевод целых чисел из десятичной системы в двоичную, восьмеричную и шестнадцатеричную

Алгоритм позволяет формализовать решение задачи. Запись алгоритма должна производиться на языке, понятном исполнителю. Если исполнителем алгоритма является человек, то алгоритм может быть записан на естественном языке (русском, английском, китайском и др.).

Для того чтобы сделать алгоритм более наглядным, часто используют блок-схемы. На блок-схеме хорошо видна структура алгоритма, по которой исполнителю (человеку) удобно отслеживать процесс его выполнения.

Для того чтобы этот алгоритм мог автоматически выполнить исполнитель компьютер, он должен быть записан на понятном для этого исполнителя языке, т. е. на языке программирования.

Реализуем алгоритмы перевода чисел из десятичной в двоичную, восьмеричную и шестнадцатеричную системы счисления в форме проекта на языках объектно-ориентированного программирования. Последовательно запишем эти алгоритмы на естественном языке, в форме блок-схемы и на языках программирования.

Проект «Перевод целых чисел». Создать проект, реализующий перевод целых десятичных чисел в двоичную, восьмеричную и шестнадцатеричную системы счисления.



Создание графического интерфейса проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

1. Разместить на форме (рис. 4.27):

- текстовое поле `TextBox1` для ввода целого числа в десятичной системе счисления;
- надпись `Label1` для вывода целого двоичного числа;

- надпись Label2 для вывода целого восьмеричного числа;
 - надпись Label3 для вывода целого шестнадцатеричного числа;
 - кнопку Button1 для создания обработчика события, реализующего перевод целых чисел в двоичную систему счисления;
 - кнопку Button2 для создания обработчика события, реализующего перевод целых чисел в восьмеричную систему счисления;
 - кнопку Button3 для создания обработчика события, реализующего перевод целых чисел в шестнадцатеричную систему счисления.
2. Сделать графический интерфейс более привлекательным. Присвоить элементам управления новые значения свойств с помощью диалогового окна *Свойства*.

Алгоритм перевода целых десятичных чисел в двоичную систему счисления на естественном языке:

- 1) Ввести десятичное целое число.
- 2) В цикле с предусловием, пока исходное целое десятичное число или целое частное больше 0, выполнить вычисления:
 - 2.1) Вычислить остаток от деления исходного целого десятичного числа или целого частного на основание новой системы (на 2).
 - 2.2) Выполнить целочисленное деление целого десятичного числа или целого частного на основание новой системы (на 2).
 - 2.3) Записать полученный остаток от деления слева от двоичного числа (остатки, записанные в обратном порядке, образуют двоичное число).
- 3) Вывести двоичное целое число.

Алгоритм перевода целых десятичных чисел в двоичную систему счисления в форме блок-схемы (с использованием языка Visual Basic .NET). Программный код удобнее записывать на том языке программирования, на котором будет кодироваться алгоритм для его выполнения на компьютере (рис. 4.24).

Кодирование алгоритма на языке программирования Visual Basic .NET. Алгоритм, записанный на естественном языке и изображенный в виде блок-схемы, рассчитан на исполнителя человека. Для того чтобы алгоритм исполнил компьютер, он должен быть закодирован на языке программирования.

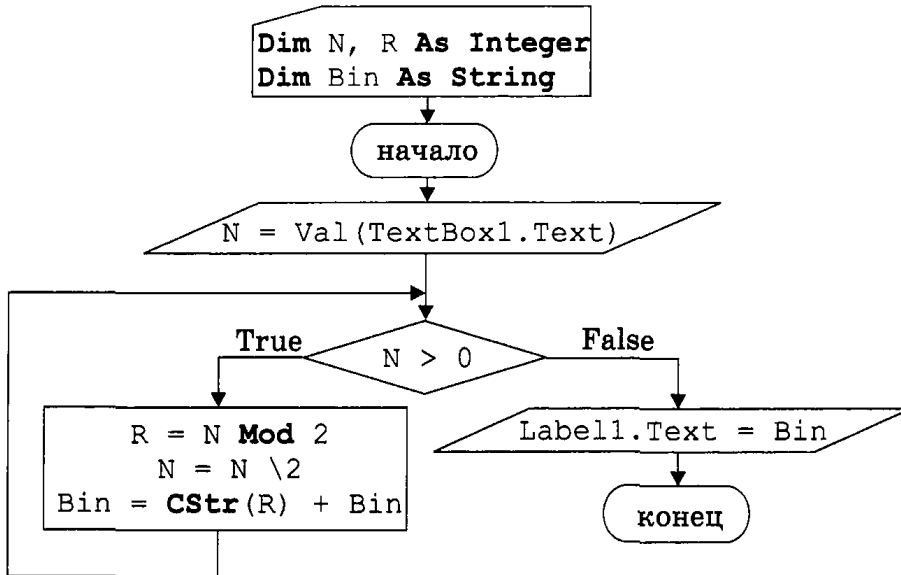


Рис. 4.24. Блок-схема алгоритма перевода целых десятичных чисел в двоичную систему счисления

Кодирование алгоритма существенно упростится, если использовать записанный на естественном языке алгоритм в качестве комментариев в программе. Комментарии не используются компьютером, так как не компилируются вместе с программой и не выполняются. Однако комментарии существенно облегчают понимание хода выполнения программы человеком.

Алгоритм на естественном языке можно скопировать и вставить в программу, поместив соответствующий символ в начале каждой строки алгоритма, чтобы превратить ее в комментарий. Затем программный код на языке программирования из блок-схемы можно разместить под каждой строкой комментария, и получится программа, которая уже документирована.



В языке программирования Visual Basic .NET комментарий в строке начинается с символа апостроф «'». В языках программирования Turbo Delphi, Visual C# и Visual J# строки комментария начинаются с двух символов косой черты «//».



Создание обработчика события, реализующего перевод целых десятичных чисел в двоичную систему счисления, на языке программирования Visual Basic .NET

3. Объявить переменные:

```
Dim N As Integer 'десятичное число
Dim R As Integer 'остаток от деления исходного
'целого десятичного числа или целого частного на
'основание новой системы
Dim Bin As String 'двоичное число в строковой
'форме
```

4. Создать обработчик события:

```
Private Sub Button1_Click(ByVal sender As System.
Object, ByVal e As System.EventArgs) Handles
Button1.Click
'1.Ввести десятичное целое число и другие
'начальные значения
N = Val(TextBox1.Text)
Label1.Text = ""
Bin = ""

'2.В цикле с предусловием, пока исходное целое
'десятичное число или целое частное больше 0,
'выполнить вычисления.
Do While N > 0

'2.1.Вычислить остаток от деления исходного
'целого десятичного числа или целого частного на
'основание новой системы (на 2).
R = N Mod 2

'2.2.Выполнить целочисленное деление целого
'десятичного числа или целого частного на
'основание новой системы (на 2).
N = N \ 2

'2.3. Записать полученный остаток от деления
'слева от двоичного числа (остатки, записанные в
'обратном порядке, образуют двоичное число).
Bin = CStr(R) + Bin

Loop

'3.Вывести двоичное целое число
Label1.Text = Bin
End Sub
```



**Создание обработчика события,
реализующего перевод целых десятичных
чисел в двоичную систему счисления,
на языке программирования Turbo Delphi**

3. Объявить переменные:

```
var  
N, R: integer;  
Bin: string;
```

4. Создать обработчик события:

```
procedure TForm1.Button1Click(Sender: TObject);  
begin  
  N := StrToInt(Edit1.Text);  
  Labell1.Caption := '';  
  Bin := '';  
  while N > 0 do  
  begin  
    R := N mod 2;  
    N := N div 2;  
    Bin := FloatToStr(R) + Bin;  
  end;  
  Labell1.Caption := Bin;  
end;
```

Алгоритм перевода целых десятичных чисел в восьмеричную систему счисления на естественном языке:

- 1) Ввести десятичное целое число.**
- 2) В цикле с предусловием, пока исходное целое десятичное число или целое частное больше 0, выполнить вычисления:**
 - 2.1) Вычислить остаток от деления исходного целого десятичного числа или целого частного на основание новой системы (на 8).**
 - 2.2) Выполнить целочисленное деление целого десятичного числа или целого частного на основание новой системы (на 8).**
 - 2.3) Записать полученный остаток от деления слева от восьмеричного числа (остатки, записанные в обратном порядке, образуют восьмеричное число).**
- 3) Вывести восьмеричное целое число.**

Алгоритм перевода целых десятичных чисел в восьмеричную систему счисления в форме блок-схемы (с использованием языка Visual C#) (рис. 4.25):

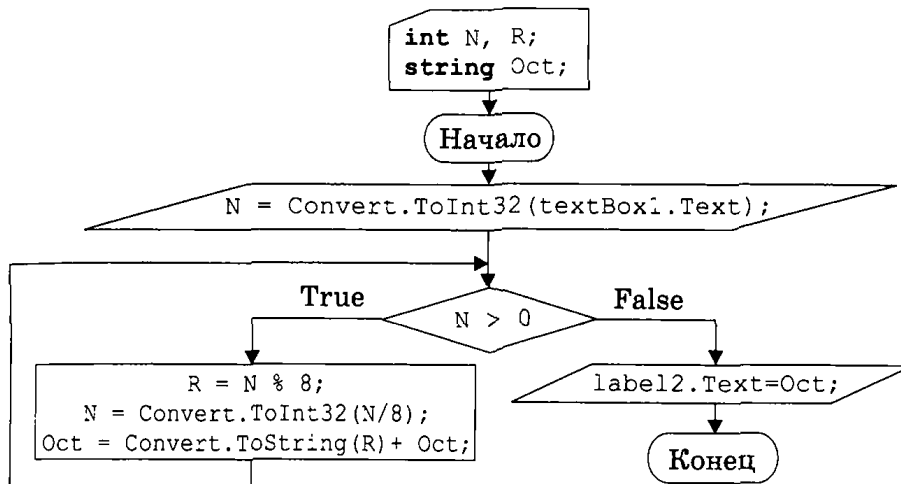


Рис. 4.25. Блок-схема алгоритма перевода целых десятичных чисел в восьмеричную систему счисления



Создание обработчика события, реализующего перевод целых десятичных чисел в восьмеричную систему счисления, на языке программирования Visual C#

3. Объявить переменные:

```

int N; //десятичное число
int R; //остаток от деления исходного целого
//десятичного числа или целого частного
//на основание новой системы
string Oct; //восьмеричное число в строковой
//форме
    
```

4. Создать обработчик события:

```

private void button2_Click(object sender, System.
EventArgs e)
{ //1.Ввести десятичное целое число и другие
//начальные данные.
N = Convert.ToInt32(textBox1.Text);
label2.Text = "";
Oct = "";
    
```



```
//2.В цикле с предусловием, пока исходное целое
//десятичное число или целое частное больше 0,
//выполнить вычисления:
while (N > 0)

{ //2.1. Вычислить остаток от деления исходного
//целого десятичного числа или целого частного
//на основание новой системы (на 8).
R = N % 8;

//2.2. Выполнить целочисленное деление целого
//десятичного числа или целого частного на
//основание новой системы (на 8).
N = Convert.ToInt32(N/8);

//2.3. Записать полученный остаток от деления
//слева от двоичного числа (остатки, записанные
//в обратном порядке, образуют восьмеричное
//число).
Oct = Convert.ToString(R)+ Oct;
}
//3.Вывести восьмеричное целое число.
label2.Text = Oct;
}
```

Алгоритм перевода целых десятичных чисел в шестнадцатеричную систему счисления на естественном языке:

- 1) Ввести десятичное целое число.
- 2) В цикле с предусловием, пока исходное целое десятичное число или целое частное больше 0, выполнить вычисления:
 - 2.1) Вычислить остаток от деления исходного целого десятичного числа или целого частного на основание новой системы (на 16).
 - 2.2) Выполнить целочисленное деление целого десятичного числа или целого частного на основание новой системы (на 16).
 - 2.3) Выразить остатки от деления цифрами новой системы счисления.
 - 2.4) Записать полученный остаток от деления слева от шестнадцатеричного числа (остатки, записанные в обратном порядке, образуют шестнадцатеричное число).
- 3) Вывести шестнадцатеричное целое число.

Алгоритм перевода целых десятичных чисел в шестнадцатеричную систему счисления в форме блок-схемы (с использованием языка Visual J#) (рис. 4.26).

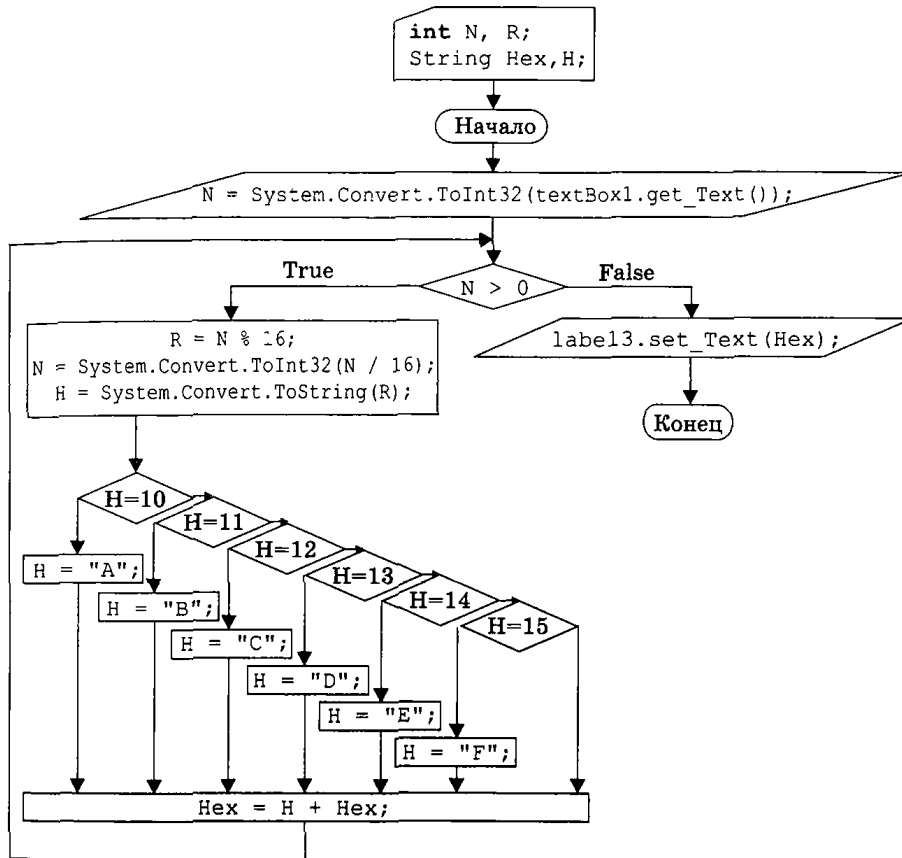


Рис. 4.26. Блок-схема алгоритма перевода целых десятичных чисел в шестнадцатеричную систему счисления



Создание обработчика события, реализующего перевод целых десятичных чисел в шестнадцатеричную систему счисления, на языке программирования Visual J#

3. Объявить переменные:

```

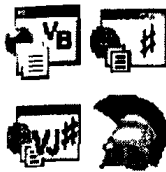
int N; //десятичное число
int R; //остаток от деления исходного целого
//десятичного числа или целого частного
//на основание новой системы
String Hex; //шестнадцатеричное число
//в строковой форме
    
```

```
String H; //разряд шестнадцатеричного числа  
//в строковой форме
```

4. Создать обработчик события:

```
private void button1_Click (Object sender, Sys-  
tem. EventArgs e)  
{//1.Ввести десятичное целое число и другие  
//начальные данные.  
N = System.Convert.ToInt32(textBox1.get_Text());  
label3.set_Text("");  
Hex = "";  
  
//2.В цикле с предусловием, пока исходное целое  
//десятичное число или целое частное больше 0,  
//выполнить вычисления:  
while (N > 0)  
  
{//2.1. Вычислить остаток от деления исходного  
//целого десятичного числа или целого частного  
//на основание новой системы (на 16).  
R = N % 16;  
  
//2.2. Выполнить целочисленное деление целого  
//десятичного числа или целого частного  
//на основание новой системы (на 16).  
N = System.Convert.ToInt32(N / 16);  
  
//2.3. Выразить остатки от деления цифрами  
//новой системы счисления.  
H = System.Convert.ToString(R);  
switch (R)  
{case 10:  
H = "A";  
break;  
case 11:  
H = "B";  
break;  
case 12:  
H = "C";  
break;  
case 13:  
H = "D";  
break;  
case 14:  
H = "E";  
break;  
case 15:  
H = "F";  
break;  
}
```

```
//2.4. Записать полученный остаток от деления
//слева от шестнадцатеричного числа (остатки,
//записанные в обратном порядке, образуют
//шестнадцатеричное число).
Hex = H + Hex;
}
//3. Вывести шестнадцатеричное целое число.
label3.set_Text (Hex);
}
```



Запуск проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

- Запустить проект на выполнение и ввести в текстовое поле десятичное число (например, 255). Щелкнуть по кнопкам, на надписи будут выведены результаты перевода, т. е. двоичное, восьмеричное и шестнадцатеричное числа (см. рис. 4.27).

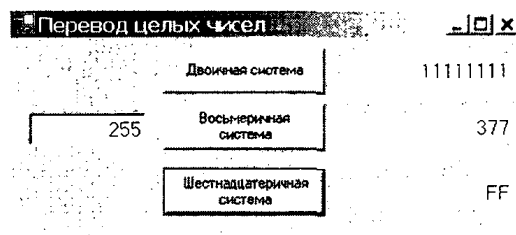


Рис. 4.27. Проект «Перевод целых чисел»

Компьютерный практикум



Windows-CD

- Создать проект «Перевод целых чисел».



...\informatika10\...\IntegerNumber\

TurboDelphi-CD



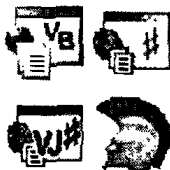
...\Projects\IntegerNumber\

4.12.2. Алгоритм перевода дробных чисел



2.8.2. Перевод дробей из десятичной системы в двоичную, восьмеричную и шестнадцатеричную

Проект «Перевод дробных чисел». Создать проект, реализующий перевод дробных десятичных чисел в двоичную, восьмеричную и шестнадцатеричную системы счисления.



Создание графического интерфейса проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

1. Разместить на форме (рис. 4.31):
 - текстовое поле `TextBox1` для ввода дробного числа в десятичной системе счисления;
 - надпись `Label1` для вывода дробного двоичного числа;
 - надпись `Label2` для вывода дробного восьмеричного числа;
 - надпись `Label3` для вывода дробного шестнадцатеричного числа;
 - кнопку `Button1` для создания обработчика события, реализующего перевод дробных чисел в двоичную систему счисления;
 - кнопку `Button2` для создания обработчика события, реализующего перевод дробных чисел в восьмеричную систему счисления;
 - кнопку `Button3` для создания обработчика события, реализующего перевод дробных чисел в шестнадцатеричную систему счисления.
2. Сделать графический интерфейс более привлекательным. Присвоить элементам управления новые значения свойств с помощью диалогового окна *Свойства*.

Алгоритм перевода дробного десятичного числа в двоичную систему счисления на естественном языке:

- 1) Ввести десятичное дробное число.
- 2) В цикле с предусловием, пока исходная дробь или дробная часть произведения не станет равной нулю или не будет достигнута требуемая точность представления числа, выполнить вычисления:
 - 2.1) Выполнить умножение десятичной дроби или полученной дробной части на основание новой системы счисления (на 2).

- 2.2) Вычислить дробную часть произведения.
- 2.3) Записать полученную целую часть произведения справа от двоичного числа (целые части произведения, записанные в прямом порядке, образуют двоичное число).
- 3) Вывести двоичное дробное число.

Алгоритм перевода дробного десятичного числа в двоичную систему счисления в форме блок-схемы (с использованием языка Visual Basic .NET). Программный код удобнее записывать на том языке программирования, на котором будет кодироваться алгоритм для его выполнения на компьютере (рис. 4.28):

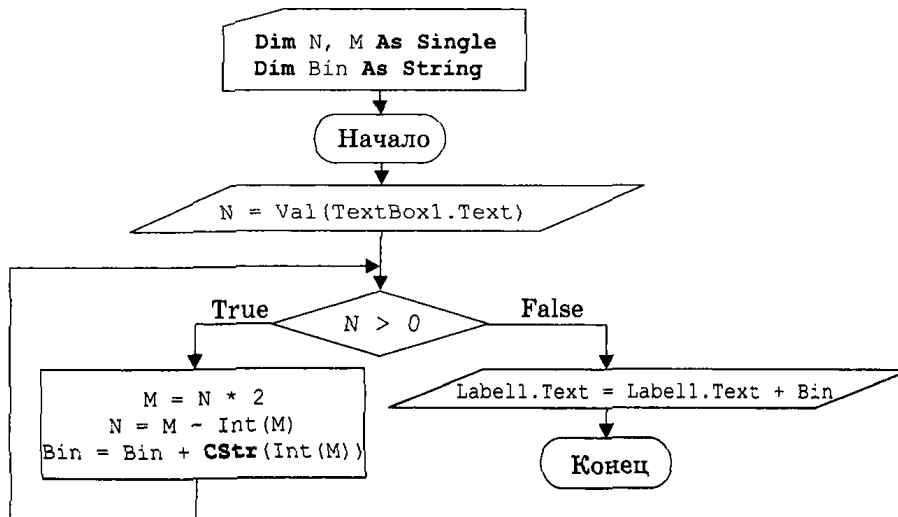


Рис. 4.28. Блок-схема алгоритма перевода дробных десятичных чисел в двоичную систему счисления

Кодирование алгоритма на языке программирования Visual Basic .NET. При реализации второго шага алгоритма цикл с предусловием может иметь бесконечное число шагов, в результате получается бесконечная периодическая дробь. Однако при реализации алгоритма на компьютере количество шагов ограничивается типом переменной, в которой хранится дробная часть произведения и которая входит в предусловие. Если дробная часть произведения хранится в переменной N, которая имеет тип Single, то ее значение может хранить 7–8 значащих цифр в десятичной системе счисления, что соответствует 23–25 значащим цифрам в двоичной системе счисления.



Создание обработчика события, реализующего перевод дробных десятичных чисел в двоичную систему счисления, на языке программирования Visual Basic .NET

3. Объявить переменные:

```
Dim N As Single 'десятичная дробь
Dim M As Single 'произведение десятичной дроби
'или полученной дробной части на основании новой
'системы счисления
Dim Bin As String 'двоичная дробь в строковой
'форме
```

4. Создать обработчик события:

```
Private Sub Button1_Click(ByVal sender As System.
Object, ByVal e As System.EventArgs) Handles
Button1.Click
'1.Ввести десятичное дробное число и другие
'начальные данные.
N = Val(TextBox1.Text)
Label1.Text = "0."
Bin = ""

'2.В цикле с предусловием, пока исходная дробь
'или дробная часть произведения не станет равной
'нулю или не будет достигнута требуемая точность
'представления числа, выполнить вычисления:
Do While N > 0

'2.1.Выполнить умножение десятичной дроби или
'полученной дробной части на основании новой
'системы счисления (на 2).
M = N * 2

'2.2.Вычислить дробную часть произведения.
N = M - Int(M)

'2.3.Записать полученную целую часть произведения
'справа от двоичного числа (целые части
'произведения, записанные в прямом порядке,
'образуют двоичное число).
Bin = Bin + CStr(Int(M))
Loop
```

```
'3. Вывести двоичное дробное число.
Labell.Text = Labell.Text + Bin
End Sub
```



Создание обработчика события, реализующего перевод дробных десятичных чисел в двоичную систему счисления, на языке программирования Turbo Delphi

3. Объявить переменные:

```
var
N, M: single;
Bin: string;
```

4. Создать обработчик события:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
N := StrToFloat(Edit1.Text);
Labell.Caption := '0.';
Bin := '';
while N > 0 Do
begin
M := N * 2;
N := Frac(M);
Bin := Bin + FloatToStr(Int(M));
end;
Labell.Caption := Labell.Caption + Bin;
end;
```

Алгоритм перевода дробного десятичного числа в восьмеричную систему счисления на естественном языке:

- 1) Ввести десятичное дробное число.
- 2) В цикле с предусловием, пока не будет достигнуто определенное количество знаков восьмеричного дробного числа, выполнить вычисления:
 - 2.1) Выполнить умножение десятичной дроби или полученной дробной части на основание новой системы счисления (на 8).
 - 2.2) Вычислить дробную часть произведения.
 - 2.3) Записать полученную целую часть произведения справа от восьмеричного числа (целые части произведения, записанные в прямом порядке, образуют восьмеричное число).
- 3) Вывести восьмеричное дробное число.

Алгоритм перевода дробного десятичного числа в восьмеричную систему счисления в форме блок-схемы (с использованием языка Visual C#) (рис. 4.29):

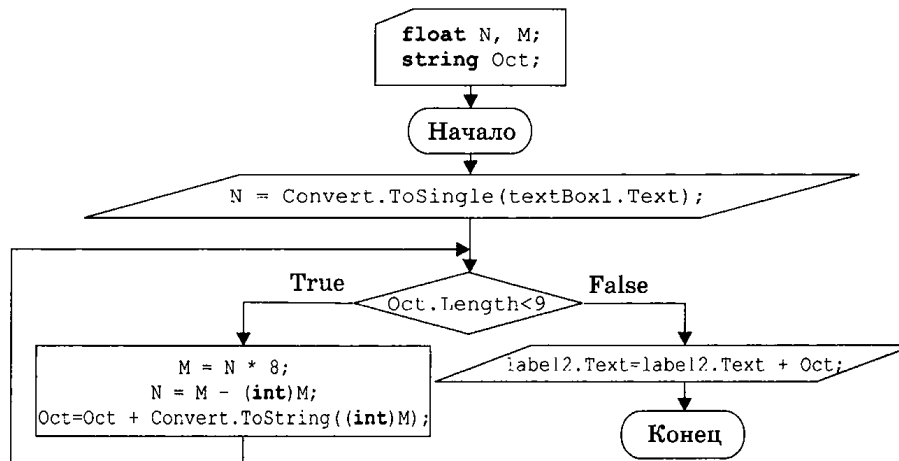


Рис. 4.29. Блок-схема алгоритма перевода дробных десятичных чисел в восьмеричную систему счисления

Кодирование алгоритма на языке программирования Visual C#. При реализации второго шага алгоритма в цикле с предусловием в качестве условия используем сравнение длины восьмеричной дроби с заданным числом. Цикл будет выполняться до тех пор, пока количество знаков в восьмеричной дроби не достигнет заданного количества.



Создание обработчика события, реализующего перевод дробных десятичных чисел в восьмеричную систему счисления, на языке программирования Visual C#

3. Объявить переменные:

```

float N; //десятичная дробь
float M; //произведение десятичной дроби или
//полученной дробной части на основание новой
//системы счисления
string Oct; //восьмеричная дробь в строковой
//форме
  
```

4. Создать обработчик события:

```

private void button2_Click(object sender, System.
EventArgs e)
  
```

```

{ //1. Ввести десятичное дробное число и другие
//начальные данные.
N = Convert.ToSingle(textBox1.Text);
label2.Text = "0.";
Oct = "";

//2. В цикле с предусловием, пока не получим
//восьмеричную дробь заданной длины, произвести
//вычисления:
while (Oct.Length < 9)

{ //2.1. Выполнить умножение исходной десятичной
//дроби или полученной дроби на основание новой
//системы (на 8).
M = N * 8;

//2.2. Вычислить дробную часть произведения.
N = M - (int)M;

//2.3. Записать полученную целую часть
//произведения справа от восьмеричного числа
//(целые части произведения, записанные в прямом
//порядке, образуют восьмеричное число).
Oct = Oct + Convert.ToString((int)M);
}
//3. Вывести восьмеричное дробное число.
label2.Text = label2.Text + Oct;
}

```

Алгоритм перевода дробного десятичного числа в шестнадцатеричную систему счисления на естественном языке:

- 1) Ввести десятичное дробное число.
- 2) В цикле со счетчиком определенное количество раз выполнить вычисления:
 - 2.1) Выполнить умножение десятичной дроби или полученной дробной части на основание новой системы счисления (на 16).
 - 2.2) Вычислить дробную часть произведения.
 - 2.3) Записать полученную целую часть произведения справа от шестнадцатеричного числа (целые части произведения, записанные в прямом порядке, образуют шестнадцатеричное число).
- 3) Вывести шестнадцатеричное дробное число.

Алгоритм перевода дробного десятичного числа в шестнадцатеричную систему счисления в форме блок-схемы (с использованием языка Visual J#) (рис. 4.30):

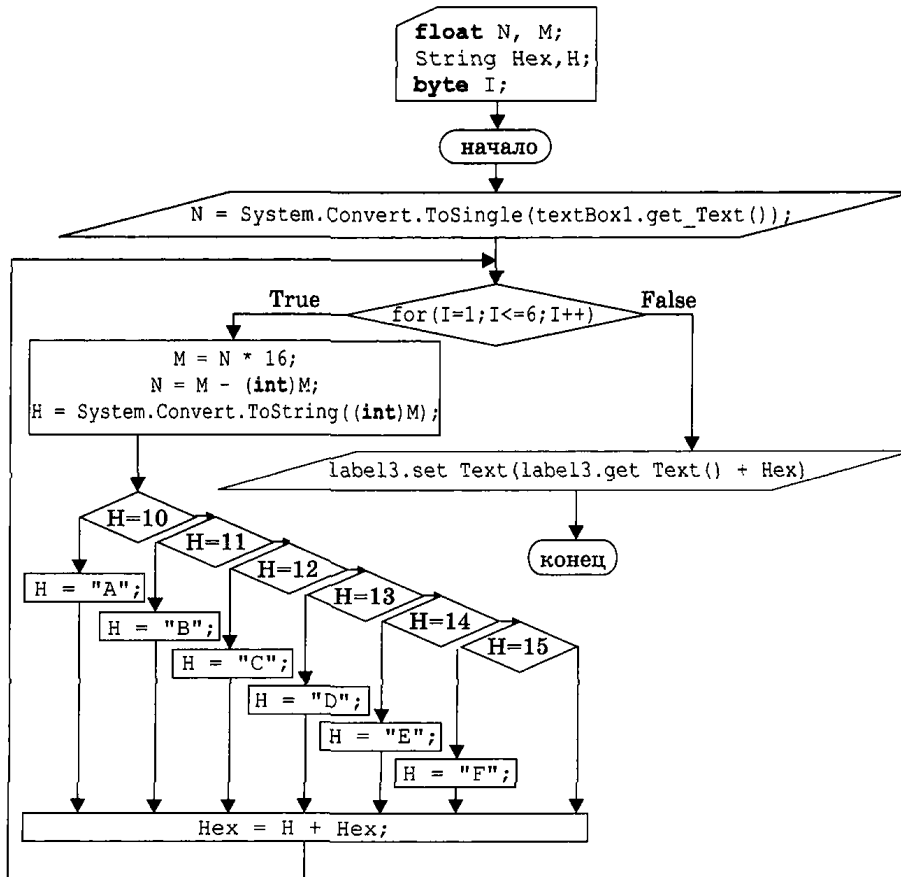


Рис. 4.30. Блок-схема алгоритма перевода дробных десятичных чисел в шестнадцатеричную систему счисления

Кодирование алгоритма на языке программирования Visual J#. Реализуем второй шаг алгоритма в форме цикла со счетчиком, т. е. вычисление шестнадцатеричной дроби произведем определенное количество раз.



**Создание обработчика события,
реализующего перевод дробных десятичных
чисел в шестнадцатеричную систему счисления,
на языке программирования Visual J#**

3. Объявить переменные:

```
float N; //десятичная дробь
float M; //произведение десятичной дроби или
//полученной дробной части на основании новой
//системы счисления
String Hex; //шестнадцатеричная дробь
//в строковой форме
String H; //разряд шестнадцатеричной дроби в
//строковой форме
byte I; //счетчик цикла
```

4. Создать обработчик события:

```
private void button3_Click (Object sender, System.
EventArgs e)
{//1.Ввести десятичное дробное число и другие
//начальные данные.
N = System.Convert.ToSingle(textBox1.get_Text());
label3.set_Text("0.");
Hex = "";

//2.В цикле со счетчиком определенное
//количество раз произвести вычисления:
for (I = 1; I <= 6; I++)
{//2.1.Выполнить умножение десятичной дроби или
//полученной дробной части на основании новой
//системы счисления (на 16).
M = N * 16;

//2.2.Вычислить дробную часть произведения.
N = M - (int)M;

//2.3.Выразить целые части произведений цифрами
//новой системы счисления.
H = System.Convert.ToString((int)M);
switch ((int)M)
{case 10:
H = "A";
break;
case 11:
H = "B";
```

```

break;
case 12:
H = "C";
break;
case 13:
H = "D";
break;
case 14:
H = "E";
break;
case 15:
H = "F";
break;
}
//2.4.Записать полученную целую часть
//произведения справа от шестнадцатеричного
//числа (целые части произведения, записанные в
//прямом порядке, образуют шестнадцатеричное
//число).
Hex = Hex + H;
}

//3.Вывести шестнадцатеричное дробное число.
label3.set_Text(label3.get_Text() + Hex);
}

```



Запуск проекта на языках Visual Basic
.NET, Visual C#, Visual J# и Turbo Delphi

5. Запустить проект на выполнение и ввести в текстовое поле десятичную дробь (например, 0,7). Щелкнуть по кнопкам, на надписи будет выведен результат перевода, т. е. двоичная, восьмеричная и шестнадцатеричная дроби (см. рис. 4.31).

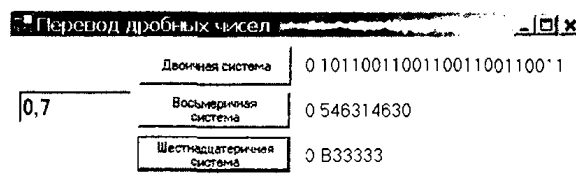



Рис. 4.31. Проект «Перевод дробных чисел»


Компьютерный практикум

Windows-CD 

4.13. Создать проект «Перевод дробных чисел».



...\informatika10\...\FractionNumber\

TurboDelphi-CD 



...\Projects\FractionNumber\

4.13. Графика в объектно-ориентированных языках программирования

4.13.1. Графика в языках программирования Visual Basic .NET, Visual C# и Visual J#

Интегрированная система программирования Visual Studio для создания рисунков, рисования текста и отображения графических изображений на формах и элементах управления использует GDI (Graphics Device Interface — графический интерфейс устройств).

Объект **Graphics** (Область рисования). Для использования GDI сначала необходимо создать область рисования, т. е. объект **Graphics**. Объект **Graphics** можно создать тремя различными способами.

Первый способ состоит в использовании метода `CreateGraphics()` формы или элемента управления, на котором надо отобразить графику. Например, создадим объект `Graph1` типа `Graphics` для графического поля `PictureBox1`.

Язык Visual Basic .NET:

```
Dim Graph1 As Graphics = PictureBox1.  
CreateGraphics()
```



Языки Visual C# и Visual J#:

```
Graphics Graph1 = pictureBox1.  
CreateGraphics();
```

Второй способ используется для создания растрового изображения, которое можно сохранить как графический файл. Сначала необходимо объявить создание растрового изображения Image1 определенного размера, затем объявить создание объекта Graph1 типа Graphics из растрового изображения Image1 и, наконец, присвоить свойству Image формы или элемента управления (например, PictureBox1.Image) значение Image1.

Язык Visual Basic .NET:

```
Dim Image1 As New Bitmap(200, 200)
Dim Graph1 As Graphics = Graphics.
FromImage(Image1)
PictureBox1.Image = Image1
```



Язык Visual C#:

```
Bitmap image1 = new Bitmap(200,200);
Graphics Graph1 = Graphics.FromImage(image1);
pictureBox1.Image = image1;
```

Язык Visual J#:

```
Bitmap image1 = new Bitmap(200,200);
Graphics Graph1 = Graphics.FromImage(image1);
pictureBox1.set_Image(image1);
```

Третий способ использует событие Paint формы или элемента управления, которое происходит при их создании или обновлении. В обработчике этого события одним из аргументов является e типа System.Windows.Forms.PaintEventArgs. В программном коде обработчика события можно объявить создание объекта Graph1 типа Graphics как свойства аргумента e.

Язык Visual Basic .NET:

```
Private Sub PictureBox1_Paint(ByVal sender
As Object, ByVal e As System.Windows.Forms.
PaintEventArgs) Handles PictureBox1.Paint
Dim Graph1 As Graphics = e.Graphics
End Sub
```



Язык Visual C#:

```
private void pictureBox1_Paint(object sender,
System.Windows.Forms.PaintEventArgs e)
{
Graphics Graph1 = e.Graphics;
}
```



Язык Visual J#:

```
private void pictureBox1_Paint(Object sender,
System.Windows.Forms.PaintEventArgs e)
{
Graphics Graph1 = e.get_Graphics();
}
```

Перо. Объект Pen (Перо) определяет цвет и ширину линии рисования. В разделе объявления переменных необходимо определить имя объекта (например, Pen1), установить цвет (например, красный Color.Red) и ширину линии в пикселях (например, 3).

Язык Visual Basic .NET:

```
Dim Pen1 As New Pen(Color.Red, 3)
```



Язык Visual C#:

```
Pen Pen1 = new Pen(Color.Red, 3);
```

Язык Visual J#:

```
Pen Pen1 = new Pen(Color.get_Red(), 3);
```

Кисть. Объект Brush (кисть) определяет цвет и стиль закрашивания прямоугольников, окружностей и других замкнутых фигур. В разделе объявления переменных необходимо определить имя объекта (например, Brush1) и установить тип закрашки и цвет (например, сплошная закрашка синего цвета SolidBrush(Color.Blue)).

Язык Visual Basic .NET:

```
Dim Brush1 As New SolidBrush(Color.Blue)
```



Язык Visual C#:

```
SolidBrush Brush1 = new SolidBrush(Color.Blue);
```

Язык Visual J#:

```
SolidBrush Brush1 = new SolidBrush(Color.
get_Blue());
```

Цвет. Цвет устанавливается как значение свойства Color. Можно установить цвет с использованием нескольких десятков цветовых констант. Ниже приведены примеры установки

зеленого цвета для объекта Pen1 (перо) и желтого цвета для объекта Brush1 (кисть).



Языки Visual Basic .NET и Язык Visual C#:

```
Pen1.Color = Color.Green
Brush1.Color = Color.Yellow
```

Язык Visual J#:

```
Pen1.set_Color(Color.get_Green());
Brush1.set_Color(Color.get_Yellow());
```

Для установки цвета в 24-битовой палитре цветов RGB используется метод Color.FromArgb(Red, Green, Blue), аргументами которого являются три числа в диапазонах 0 до 255 (интенсивности красного, зеленого и синего цветов). Например, так можно установить пурпурный цвет для кисти Brush1.



Языки Visual Basic .NET и Язык Visual C#:

```
Brush1.Color = Color.FromArgb(255, 0, 255)
```

Язык Visual J#:

```
Brush1.set_Color(Color.FromArgb(255, 0, 255));
```

Цвет пера или кисти можно также установить с использованием элемента управления ColorDialog. Для этого данный элемент управления необходимо поместить на форму и ввести в программный код следующие строки:



Языки Visual Basic .NET и Язык Visual C#:

```
ColorDialog1.ShowDialog()
Pen1.Color = ColorDialog1.Color
```

Язык Visual J#:

```
colorDialog1.ShowDialog();
Pen1.set_Color(colorDialog1.get_Color());
```

После выполнения первой строки программного кода появится диалоговое окно *Цвет* (рис. 4.32). Вторая строка программного кода присваивает перу цвет, который можно выбрать в диалоговом окне *Цвет* с помощью мыши из палитры

48 основных цветов. После щелчка по кнопке *Определить цвет* в раскрывшемся диалоговом окне цвет можно выбрать из полной палитры 16 777 216 цветов, а также установить путем ввода в текстовые поля интенсивностей базовых цветов в системе цветопередачи RGB или оттенка, контрастности и яркости в системе цветопередачи HSB.

1.1.3. Палитры цветов в системах цветопередачи RGB, CMYK и HSB

Информатика-9

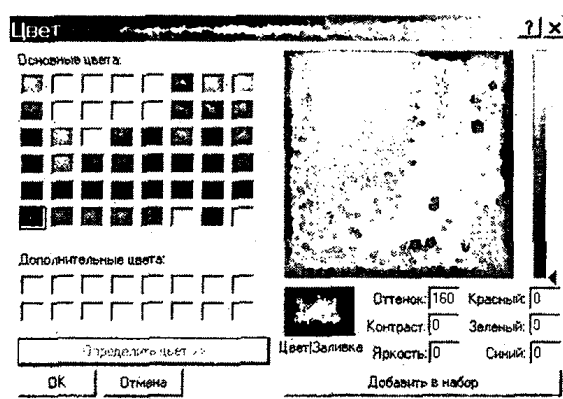


Рис. 4.32. Диалоговое окно *Цвет*

Графические методы. Графические фигуры рисуются с использованием графических методов. Замкнутые фигуры, такие как прямоугольники или эллипсы, состоят из двух частей — контура и внутренней области. Контур рисуется с использованием заданного пера, а внутренняя область закрашивается с использованием заданной кисти.



В языках Visual C# и Visual J# строки программного кода заканчиваются знаком точка с запятой «;».

`DrawLine()` — метод рисования линии, аргументами которого являются перо определенного цвета и толщины (например, `Pen1`), а также координаты концов линии `X1`, `Y1` и `X2`, `Y2`.



Языки Visual Basic .NET, Visual C# и Visual J#:
`Graph1.DrawLine(Pen1, X1, Y1, X2, Y2)`

`DrawRectangle()` — метод рисования прямоугольника, аргументами которого являются перо определенного цвета и

толщины (например, Pen1), а также координаты левого верхнего угла X1, Y1, ширина Width и высота Height.



Языки Visual Basic .NET, Visual C# и Visual J#:
 Graph1.DrawRectangle(Pen1, X1, Y1, Width, Height)

FillRectangle() — метод закрашки прямоугольника с использованием кисти определенного цвета.



Языки Visual Basic .NET, Visual C# и Visual J#:
 Graph1.FillRectangle(Brush1, X1, Y1, Width, Height)

DrawEllipse() — метод рисования окружности или эллипса, аргументами которого являются перо определенного цвета и толщины (например, Pen1), а также координаты левого верхнего угла описанного прямоугольника X1, Y1, ширина Width и высота Height.



Языки Visual Basic .NET, Visual C# и Visual J#:
 Graph1.DrawEllipse(Pen1, X1, Y1, Width, Height)

FillEllipse() — метод закрашки окружности или эллипса с использованием кисти определенного цвета.



Языки Visual Basic .NET, Visual C# и Visual J#:
 Graph1.FillEllipse(Brush1, X1, Y1, Width, Height)

i Для рисования точки с заданными координатами X1 и Y1 можно использовать методы DrawRectangle(Pen1, X1, Y1, 1, 1) или DrawEllipse(Pen1, X1, Y1, 1, 1), в которых аргументы Width и Height равны 1.

Graph1.Clear() — метод, заданным цветом (например, белым) стирающий изображения в области рисования.



Языки Visual Basic .NET и Visual C#:
 Graph1.Clear(Color.White)

Язык Visual J#:
 Graph1.Clear(Color.get_White());

Рисование текста. Метод DrawString() позволяет выводить текст в область рисования. Аргументами метода является строка текста, шрифт, кисть и координаты начала строки. Объекты шрифт (например, drawFont) и кисть (например, drawBrush) необходимо объявить.

Язык Visual Basic .NET:

```
Dim drawFont As New Font("Arial", 12)
Dim drawBrush As New SolidBrush(Color.Black)
```



Язык Visual C#:

```
Font drawFont = new Font("Arial", 12);
Brush drawBrush = new SolidBrush(Color.Black);
```

Язык Visual J#:

```
Font drawFont = new Font("Arial", 12);
Brush drawBrush = new SolidBrush(Color.get_
Black());
```

Рисование текста в левом верхнем углу области рисования можно осуществить так:



Языки Visual Basic .NET, Visual C# и Visual J#:

```
Graph1.DrawString("Текст", drawFont,
drawBrush, 0, 0)
```

Проект «Графический редактор». Создать проект, который позволит рисовать линии, прямоугольники, закрашенные прямоугольники, окружности и закрашенные окружности. Ввод координат осуществлять щелчками по графическому полю. Для рисования фигур использовать меню и панель инструментов. Обеспечить возможность установки для пера и кисти любого цвета из полной палитры цветов. Обеспечить возможность открытия и сохранения графических файлов.



Создание графического интерфейса проекта на языках Visual Basic .NET, Visual C# и Visual J#



1. Поместить на форму (рис. 4.36):
 - графическое поле PictureBox1, которое будет использоваться в качестве области рисования;

- четыре надписи Label1, Label2, Label3 и Label4 для вывода координат;
- четыре надписи для вывода имен координат;
- меню MainMenu1 для создания меню проекта;
- панель инструментов ToolBar1 для создания шести кнопок, обеспечивающих рисование графических примитивов и очистку поля рисования;
- коллекцию изображений ImageList1 для хранения изображений, которые будут помещены на кнопки панели инструментов;
- диалог ColorDialog1, который позволяет выбрать цвет с использованием диалогового окна *Цвет*;
- диалог OpenFileDialog1, который позволяет выбрать файл для открытия с использованием диалогового окна *Открыть*;
- диалог SaveFileDialog1, который позволяет выбрать имя файла при его сохранении с использованием диалогового окна *Сохранить*.



Элементы управления MainMenu1, ImageList1, ColorDialog1, OpenFileDialog1 и SaveFileDialog1 будут видны только в процессе создания проекта (рис. 4.33).

MainMenu1 ImageList1 ColorDialog1 OpenFileDialog1 SaveFileDialog1

Рис. 4.33. Элементы управления, которые не видны в процессе выполнения проекта



Объявление переменных и объектов и создание обработчика событий на языке Visual Basic .NET

2. Объявить переменные, которые будут содержать координаты двух точек, а также перо и кисть, которые будут использоваться для рисования графических примитивов.

```
Dim X1, Y1, X2, Y2 As Integer
Dim Pen1 As New Pen(Color.Black, 5)
Dim Brush1 As New SolidBrush(Color.Red)
```

3. Создать программный код обработчика события, который обеспечивает запоминание и вывод на надписи координат щелчков левой и правой кнопками мыши по графическому полю. Использовать оператор выбора, который в зависимости от условия (значения аргумента обработчика со-

бытия e.Button) запоминает и выводит ту или иную пару координат.

```

Private Sub PictureBox1_Click(ByVal sender As
Object, ByVal e As System.Windows.Forms.
MouseEventArgs) Handles PictureBox1.MouseDown
Select Case e.Button
Case MouseButton.Left
    X1 = e.X
    Y1 = e.Y
    Label1.Text = X1
    Label2.Text = Y1
Case MouseButton.Right
    X2 = e.X
    Y2 = e.Y
    Label3.Text = X2
    Label4.Text = Y2
End Select
End Sub
    
```

Создадим меню графического редактора, для чего используем элемент управления mainMenu1:

Файл	Графические примитивы	Цвет
Открыть	Линия	Перо
Сохранить	Прямоугольник	Кисть
	Закрашенный прямоугольник	
	Окружность	
	Закрашенная окружность	
	Очистить	



Создание меню и обработчиков событий рисования графических примитивов на языке Visual C#

4. Разместить на форме элемент управления mainMenu1.
5. В появившемся в левом верхнем углу формы в редакторе меню создать заголовок первого уровня. В поле *Прототип для текста* внести пункт меню *Графические примитивы*.
6. Для создания пункта меню перейти на следующую строку в редакторе меню. Внести пункты меню *Линия*, *Прямоугольник*, *Закрашенный прямоугольник*, *Окружность*, *Закрашенная окружность* и *Очистить*.

7. Аналогично создать пункты меню верхнего уровня *Файл* и *Цвет* и их подпункты.
8. Использовать второй способ для создания области рисования с использованием растрового изображения `image1`, которое можно сохранить как графический файл. Объявить `image1` как растровое изображение.
`Bitmap image1 = new Bitmap(300,300);`
9. Двойным щелчком по пунктам меню последовательно создать заготовки обработчиков событий. Ввести программный код обработчика события рисования линии.
`private void menuItem2_Click(object sender, System.EventArgs e)
{Graphics Graph1 = Graphics.FromImage(image1);
pictureBox1.Image = image1;
Graph1.DrawLine(Pen1, X1, Y1, X2, Y2);
}`
10. Создать программный код обработчика события рисования прямоугольника. Для вычисления ширины и высоты прямоугольника использовать абсолютное значение разности соответствующих координат.
`private void menuItem2_Click(object sender, System.EventArgs e)
{Graphics Graph1 = Graphics.FromImage(image1);
pictureBox1.Image = image1;
Graph1.DrawRectangle(Pen1, X1, Y1, Math.Abs(X2-X1),
Math.Abs(Y2-Y1));}`
11. Создать программный код обработчика события рисования закрашенного прямоугольника.
`private void menuItem2_Click(object sender, System.EventArgs e)
{Graphics Graph1 = Graphics.FromImage(image1);
pictureBox1.Image = image1;
Graph1.DrawRectangle(Pen1, X1, Y1, Math.Abs(X2-X1),
Math.Abs(Y2-Y1));
Graph1.FillRectangle(Brush1, X1, Y1, Math.Abs(X2-X1),
Math.Abs(Y2-Y1));
}`
12. Создать программный код обработчика события рисования окружности.
`private void menuItem2_Click(object sender, System.EventArgs e)
{Graphics Graph1 = Graphics.FromImage(image1);`

```

pictureBox1.Image = image1;
Graph1.DrawEllipse(Pen1, X1, Y1, Math.Abs(X2 - X1),
Math.Abs(Y2 - Y1));
}

```

13. Создать программный код обработчика события рисования закрашенной окружности.

```

private void menuItem2_Click(object sender, System.
EventArgs e)
{Graphics Graph1 = Graphics.FromImage(image1);
pictureBox1.Image = image1;
Graph1.DrawEllipse(Pen1, X1, Y1, Math.Abs(X2 - X1),
Math.Abs(Y2 - Y1));
Graph1.FillEllipse(Brush1, X1, Y1, Math.Abs(X2-X1),
Math.Abs(Y2 - Y1));
}

```

14. Создать программный код обработчика события очистки области рисования.

```

private void menuItem2_Click(object sender, System.
EventArgs e)
{Graphics Graph1 = Graphics.FromImage(image1);
pictureBox1.Image = image1;
Graph1.Clear(Color.White);
}

```

Создадим панель инструментов графического редактора. В панель инструментов должны входить кнопки *Линия*, *Прямоугольник*, *Закрашенный прямоугольник*, *Окружность*, *Закрашенная окружность* и *Очистить*.

Для создания панели инструментов используем элемент управления `toolBar1`.



Создание панели инструментов и обработчиков событий рисования графических примитивов на языке Visual J#

15. Разместить на форме элемент управления `toolBar1`.
16. Выделить элемент управления `toolBar1` и в окне *Свойства* у свойства *Buttons* активизировать значение (*Коллекция*).
17. В появившемся окне *Редактор коллекции ToolBarButton* (рис. 4.34) создать шесть кнопок на панели инструментов, нажав шесть раз кнопку *Добавить*.

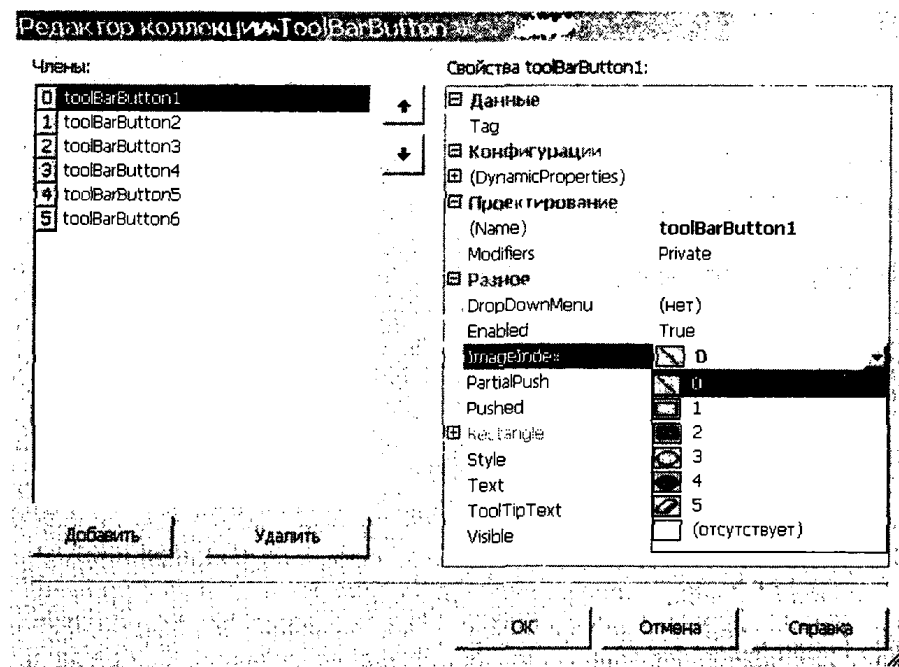


Рис. 4.34. Редактор коллекции кнопок на панели инструментов

На каждой кнопке панели инструментов разместим соответствующее изображение, которое хранится в коллекции изображений в элементе управления `imageList1`.

18. Разместить на форме элемент управления `imageList1`.
19. Выделить элемент управления `imageList1` и в окне *Свойства* у свойства *Images* активизировать значение (*Коллекция*).
20. В появившемся окне *Редактор коллекции Image* (рис. 4.35) добавить шесть изображений для размещения на кнопках панели инструментов, нажав шесть раз кнопку *Добавить*.

Установим соответствие между коллекцией кнопок на панели инструментов `toolBar1` и коллекцией изображений `imageList1`.

21. Выделить элемент управления `toolBar1` и в окне *Свойства* у свойства *ImageList* установить значение *imageList1*.

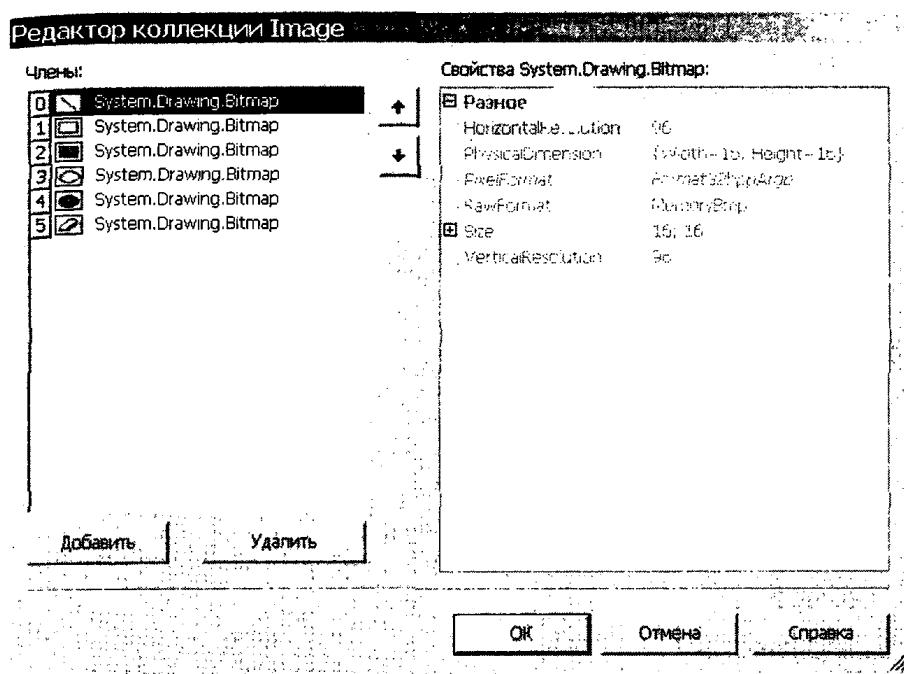


Рис. 4.35. Редактор коллекции изображений для кнопок панели инструментов

22. Использовать для создания области рисования первый способ, т. е. применить метод `CreateGraphics()` к графическому полю. Создать программный код обработчика события щелчка по кнопке панели инструментов. С помощью оператора выбора в зависимости от выбранной кнопки осуществить рисование графических примитивов или очистку графического поля.

```
private void toolBar1_ButtonClick (Object sender,
System.Windows.Forms.ToolBarButtonClickEventArgs e)
{Graphics Graph2 = pictureBox1.CreateGraphics();
switch (toolBar1.get_Buttons().
IndexOf(e.get_Button()))
{case 0:
Graph2.DrawLine(Pen1, X1, Y1, X2, Y2);
break;
case 1:
Graph2.DrawRectangle(Pen1,X1,Y1,System.Math.
Abs(X2-X1), System.Math.Abs(Y2-Y1));
break;
case 2:
```

```

Graph2.DrawRectangle(Pen1, X1, Y1, System.Math.
Abs(X2-X1), System.Math.Abs(Y2 - Y1));
Graph2.FillRectangle(Brush1, X1, Y1, System.
Math.Abs(X2-X1), System.Math.Abs(Y2-Y1));
break;
case 3:
Graph2.DrawEllipse(Pen1, X1, Y1, System.Math.
Abs(X2-X1), System.Math.Abs(Y2 - Y1));
break;
case 4:
Graph2.DrawEllipse(Pen1, X1, Y1, System.Math.
Abs(X2-X1), System.Math.Abs(Y2 - Y1));
Graph2.FillEllipse(Brush1, X1, Y1, System.Math.
Abs(X2-X1), System.Math.Abs(Y2 - Y1));
break;
case 5:
Graph2.Clear(Color.get_White());
break;
}
}

```

Двойным щелчком по пунктам меню *Цвет* последовательно создадим заготовки обработчиков событий и введем их программный код. Для выбора цвета используем диалог `ColorDialog1`, а для вывода диалогового окна *Цвет* — метод `ShowDialog()`.



Создание обработчиков событий установки цвета пера и кисти на языке Visual Basic .NET

23. Создать обработчик события установки цвета пера.


```

Private Sub MenuItem9_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MenuItem9.Click
ColorDialog1.ShowDialog()
Pen1.Color = ColorDialog1.Color
End Sub

```
24. Создать обработчик события установки цвета кисти.


```

Private Sub MenuItem10_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MenuItem10.Click
ColorDialog1.ShowDialog()
Brush1.Color = ColorDialog1.Color
End Sub

```

Двойным щелчком по пунктам меню *Файл* последовательно создадим заготовки обработчиков событий и введем их программный код. Для вывода диалоговых окон используем диалоги `openFileDialog1` и `saveFileDialog1` и метод `ShowDialog()`.



Создание обработчиков событий открытия и сохранения графических файлов на языке Visual C#

25. Создать обработчик события открытия растрового графического файла.

```
private void menuItem2_Click(object sender,
System.EventArgs e)
{openFileDialog1.ShowDialog();
pictureBox1.Image = Image.FromFile(openFileDialog1.
FileName);
}
```

26. Создать обработчик события сохранения растрового графического файла.

```
private void menuItem3_Click(object sender,
System.EventArgs e)
{saveFileDialog1.Filter = "BMP (*.bmp)|*.bmp";
saveFileDialog1.ShowDialog();
image1.Save(saveFileDialog1.FileName, System.
Drawing.Imaging.ImageFormat.Bmp);
}
```



Запуск проекта на языках Visual Basic .NET, Visual C# и Visual J#

27. Запустить проект.

Осуществить щелчки левой и правой кнопками мыши по графическому полю. На надписи будут выведены координаты двух точек (см. рис. 4.36).

С использованием меню *Цвет* установить цвета пера и кисти.

С использованием меню *Графические примитивы* нарисовать графические фигуры.

С использованием меню *Файл* сохранить рисунок как растровый графический файл.

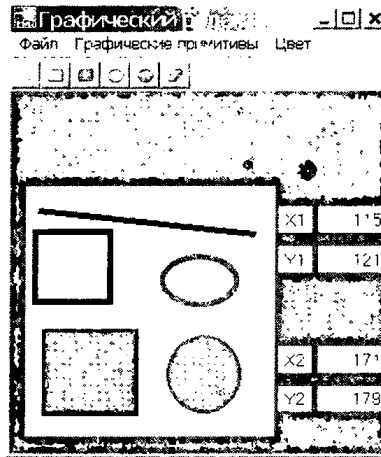
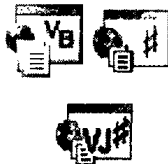


Рис. 4.36. Проект «Графический редактор»

Проект «Треугольник». Создать проект, который позволяет нарисовать по заданным трем точкам треугольник, вычислить его периметр и площадь. Ввод координат вершин треугольника осуществлять щелчками по графическому полю. Вывод координат вершин треугольника осуществлять методом рисования текста в графическом поле.



Создание графического интерфейса проекта на языках Visual Basic .NET, Visual C# и Visual J#

1. Поместить на форму (рис. 4.37):

- графическое поле `PictureBox1`, которое будет использоваться в качестве области рисования;
- две надписи `Label1` и `Label2` для вывода периметра и площади треугольника;
- две надписи для вывода поясняющих текстов;
- меню `MainMenu1` для создания меню проекта.



Объявление переменных и объектов и создание обработчика событий запоминания координат на языке Visual C#

- 2. Объявить переменные, которые будут содержать координаты вершин треугольника, длины его сторон, а также периметр и площадь. Кроме того, объявить область рисования, перо, которым будут рисоваться стороны треугольника,**

```

а также шрифт и кисть, которые будут использоваться
для рисования значений координат в графическом поле.
int X1, X2, Y1, Y2, X3, Y3;
double L1, L2, L3, P, S;
Graphics Graph1;
Pen Pen1 = new Pen(Color.Black, 2);
Font drawFont = new Font("Arial", 12);
Brush drawBrush = new SolidBrush(Color.Blue);

```

3. Создать программный код обработчика события, который обеспечивает запоминание координат щелчков левой, правой и средней кнопками мыши по графическому полю. Использовать оператор выбора, который в зависимости от условия (значения аргумента обработчика события e.Button) запоминает ту или иную пару координат.

```

private void pictureBox1_Click(object sender,
System.Windows.Forms.MouseEventArgs e)
{switch (e.Button)
  {case MouseButton.Left:
    X1 = e.X;
    Y1 = e.Y;
    break;
  case MouseButton.Right:
    X2 = e.X;
    Y2 = e.Y;
    break;
  case MouseButton.Middle:
    X3 = e.X;
    Y3 = e.Y;
    break;
  }
}

```



Создание обработчиков событий рисования треугольника на языке Visual Basic .NET

4. Использовать третий способ для создания области рисования. В программном коде обработчика события Paint графического поля объявим создание объекта Graph1 типа Graphics как свойства аргумента e. Для вызова события Paint обновить графическое поле в обработчике события щелчка по форме.

В обработчике события Paint графического поля вывести значения координат вершин треугольника и нарисовать

стороны треугольника, соединив координаты вершин прямыми линиями.

```
Private Sub Form1_Click(ByVal sender As Object,
ByVal e As System.EventArgs) Handles MyBase.Click
PictureBox1.Refresh()
End Sub
```

```
Private Sub PictureBox1_Paint(ByVal sender As
Object, ByVal e As System.Windows.Forms.Paint-
EventArgs) Handles PictureBox1.Paint
Dim Graph1 As Graphics = e.Graphics
Graph1.DrawLine(Pen1, X1, Y1, X2, Y2)
Graph1.DrawLine(Pen1, X1, Y1, X3, Y3)
Graph1.DrawLine(Pen1, X2, Y2, X3, Y3)
Graph1.DrawString(Convert.ToString(X1)+", "+
Convert.ToString(Y1), drawFont, drawBrush, X1, Y1)
Graph1.DrawString(Convert.ToString(X2)+", "+
Convert.ToString(Y2), drawFont, drawBrush, X2, Y2)
Graph1.DrawString(Convert.ToString(X3)+", "+
Convert.ToString(Y3), drawFont, drawBrush, X3, Y3)
End Sub
```

Создадим меню графического редактора, для чего используем элемент управления mainMenu1:

Треугольник
Периметр
Площадь
Очистить



Создание меню и обработчиков событий вычисления периметра и площади треугольника, а также очистки графического поля на языке Visual J#

5. Разместить на форме элемент управления mainMenu1.
6. В появившемся в левом верхнем углу формы в редакторе меню создать заголовок первого уровня. В поле *Прототип для текста* внести пункт меню *Треугольник*.
7. Для создания пункта меню перейти на следующую строку в редакторе меню. Ввести пункты меню *Периметр*, *Площадь* и *Очистить*.
8. Создать обработчик события, реализующий вычисление периметра треугольника.

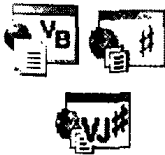
```
private void menuItem2_Click (Object sender,
System.EventArgs e)
{L1 = System.Math.Sqrt(System.Math.Pow((X2 - X1),2)
+ System.Math.Pow((Y2 - Y1), 2));
L2 = System.Math.Sqrt(System.Math.Pow((X3 - X1),
2) + System.Math.Pow((Y3 - Y1), 2));
L3 = System.Math.Sqrt(System.Math.Pow((X2 - X3),
2) + System.Math.Pow((Y2 - Y3), 2));
P = L1 + L2 + L3;
label1.set_Text(System.Convert.ToString(P));
}
```

9. Создать обработчик события, реализующий вычисление площади треугольника.

```
private void menuItem3_Click (Object sender,
System.EventArgs e)
{S = System.Math.Sqrt(P/2 * (P/2 - L1) * (P/2 -
L2) * (P/2 - L3));
label2.set_Text(System.Convert.ToString(S));
}
```

10. Создать обработчик события, реализующий очистку графического поля.

```
private void menuItem4_Click (Object sender,
System.EventArgs e)
{Graph1 = this.pictureBox1.CreateGraphics();
Graph1.Clear(Color.get_White());
}
```



Запуск проекта на языках Visual Basic .NET, Visual C# и Visual J#

11. Запустить проект.

Осуществить щелчки левой, правой и средней кнопками мыши по графическому полю.

Щелкнуть по форме, в графическом поле будет нарисован треугольник и выведены координаты его вершин (см. рис. 4.37).

С использованием меню *Треугольник* вычислить значения периметра и площади треугольника.

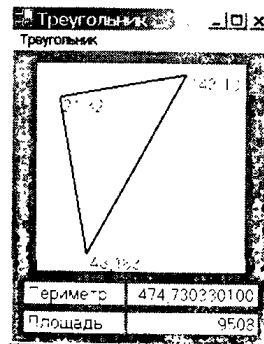



Рис. 4.37. Проект «Треугольник»

Компьютерный практикум



Windows-CD 

4.14. Создать проект «Графический редактор».



...\informatika10\...\GraphRedactor\

4.15. Создать проект «Треугольник».



...\informatika10\...\Triangle\

4.13.2. Графика в языке программирования Turbo Delphi

Холст. Язык программирования Turbo Delphi для создания рисунков, рисования текста и отображения графических изображений на формах и элементах управления использует в качестве области рисования объект Canvas (Холст). Объект Canvas можно создать для формы или элемента управления (чаще всего объект Canvas создается для графического поля Image).



```
Image1.Canvas;
```

Перо. Объект Pen (Перо) определяет цвет (например, красный clRed) и ширину линии в пикселях (например, 3) рисования контуров графических объектов.



```
Image1.Canvas.Pen.Color := clRed;
Image1.Canvas.Pen.Width := 3;
```

Кисть. Объект Brush (Кисть) определяет цвет (например, закраска синего цвета clBlue) и стиль закрашивания прямоугольников, окружностей и других замкнутых фигур (например, сплошная закраска bsSolid).



```
Image1.Canvas.Brush.Color := clBlue;
Image1.Canvas.Brush.Style := bsSolid;
```

Цвет. Цвет устанавливается как значение свойства `Color`, которое можно задать различными способами:

- с помощью одной из нескольких десятков констант, определяющих цвет (`clBlack` — черный, `clBlue` — синий, `clGreen` — зеленый, `clRed` — красный, `clYellow` — желтый, `clWhite` — белый и т. д.);
- с помощью функции `StringToColor()`, реализующей цветовую модель RGB (красный, зеленый, синий). Аргументом этой функции является шестнадцатеричное представление интенсивностей базовых цветов, в котором на каждый цвет отводится два шестнадцатеричных разряда (например, `$00FF0000` — синий, `$0000FF00` — зеленый, `$000000FF` — красный; `$00000000` — черный и `$00FFFFFF` — белый);



```
Image1.Canvas.Pen.Color := StringToColor
('$0000FF00');
```

- цвет пера или кисти можно также установить с использованием элемента управления `ColorDialog`. Для этого данный элемент управления необходимо поместить на форму и ввести в программный код следующие строки.



```
ColorDialog1.Execute;
Image1.Canvas.Pen.Color := ColorDialog1.Color;
```

Графические методы. Графические фигуры рисуются с использованием графических методов.

`MoveTo(X1, Y1)` — метод осуществляет переход (без рисования линии) в точку с требуемыми текущими координатами (например, в точку с координатами `X1, Y1`).



```
Image1.Canvas.MoveTo(X1, Y1);
```

`LineTo(X2, Y2)` — метод рисует прямую линию от точки с текущими координатами до точки с заданными координатами (например, в точку с координатами `X2, Y2`). Цвет и толщина линии определяются свойствами объекта `Pen`.



```
Image1.Canvas.LineTo(X2, Y2);
```

`Rectangle(X1,Y1, X2,Y2)` — метод рисует прямоугольник с координатами левого верхнего и правого нижнего углов. Цвет и толщина линии контура определяются свойствами объекта `Pen`, а цвет и стиль заливки — свойствами объекта `Brush`.



```
Image1.Canvas.Rectangle(X1,Y1,X2,Y2);
```

`Ellipse(X1,Y1, X2,Y2)` — метод рисует окружность, вписанную в прямоугольник с заданными координатами левого верхнего и правого нижнего углов. Цвет и толщина линии контура определяются свойствами объекта `Pen`, а цвет и стиль заливки — свойствами объекта `Brush`.



```
Image1.Canvas.Ellipse(X1,Y1,X2,Y2);
```

`Pixels[X,Y]` — свойство, которое закрашивает точку холста с заданными координатами определенным цветом.



```
Image1.Canvas.Pixels[X,Y] := color;
```

Рисование текста. Метод `TextOut()` позволяет выводить текст в область рисования. Аргументами метода являются координаты начала строки и строка текста. Название шрифта, его размер в пунктах, начертание (полужирное, курсив или подчеркивание) и цвет текста определяются свойствами объекта `Font`. Рисование текста в левом верхнем углу области рисования шрифтом `Arial` размером 12 пунктов можно осуществить так.



```
Image1.Canvas.Font.Name := 'Arial';
Image1.Canvas.Font.Size := 12;
Image1.Canvas.TextOut(0,0,'Текст');
```

Проект «Графический редактор». Создать проект, который позволит рисовать линии, закрашенные прямоугольники и закрашенные окружности. Ввод координат осуществлять путем нажатия и отпускания кнопки мыши на графическом поле. Для рисования фигур использовать меню и панель инструментов. Обеспечить возможность открытия и сохранения графических файлов.



Создание проекта «Графический редактор» на языке Turbo Delphi

1. Поместить на форму (рис. 4.38):

- графическое поле Image1, которое будет использоваться в качестве области рисования;
- четыре надписи Label1, Label2, Label3 и Label4 для вывода координат;
- четыре надписи для вывода имен координат;
- меню MainMenu1 для создания пунктов меню *Файл* и *Графические примитивы*;
- панель инструментов ToolBar1 для создания четырех кнопок, обеспечивающих рисование графических примитивов и очистку поля рисования;
- коллекцию изображений ImageList1 для хранения изображений, которые будут помещены на кнопки панели инструментов;
- диалог ColorDialog1, который позволяет выбрать цвет с использованием диалогового окна *Цвет*;
- диалог OpenFileDialog1, который позволяет выбрать файл для открытия с использованием диалогового окна *Открыть*;
- диалог SaveDialog1, который позволяет выбрать имя файла при его сохранении с использованием диалогового окна *Сохранить*.

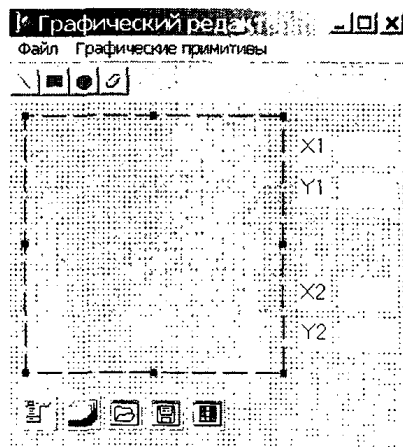


Рис. 4.38. Графический интерфейс проекта «Графический редактор»



Элементы управления `MainMenu1`, `ImageList1`, `OpenDialog1`, `SaveDialog1` и `ColorDialog1` будут видны только в процессе создания проекта.

- Объявить переменные, которые будут содержать координаты двух точек на холсте.

```
var
X1: integer;
Y1: integer;
X2: integer;
Y2: integer;
```

- Создать программные коды обработчиков событий, которые обеспечивают запоминание и вывод на надписи координат после нажатия (событие `ImageMouseDown`) и отпущения (событие `ImageMouseUp`) кнопки мыши на графическом поле.

```
procedure TForm1.ImageMouseDown(Sender: TObject;
Button: TMouseButton; Shift: TShiftState; X, Y:
Integer);
```

```
begin
X1:=X;
Y1:=Y;
Label1.Caption := IntToStr(X1);
Label2.Caption := IntToStr(Y1);
end;
```

```
procedure TForm1.ImageMouseUp(Sender: TObject;
Button: TMouseButton; Shift: TShiftState; X, Y:
Integer);
```

```
begin
X2:=X;
Y2:=Y;
Label3.Caption := IntToStr(X2);
Label4.Caption := IntToStr(Y2);
end;
```

Создадим меню графического редактора, для чего используем элемент управления `MainMenu1`:

Файл	Графические примитивы
Открыть	Линия
Сохранить	Закрашенный прямоугольник
	Закрашенная окружность
	Очистить

4. Разместить на форме элемент управления MainMenu1.
5. Осуществить двойной щелчок по элементу управления MainMenu1. В появившемся диалоговом окне Form1.MainMenu1 редактора меню создать меню проекта (рис. 4.39).

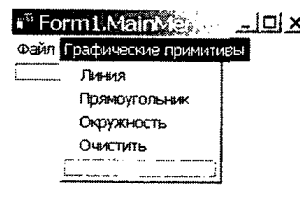


Рис. 4.39. Редактор меню

6. Двойным щелчком по пунктам меню последовательно создать заготовки обработчиков событий рисования графических примитивов. Ввести программный код обработчика события рисования линии.

```
procedure TForm1.N2Click(Sender: TObject);
begin
    Image1.Canvas.Pen.Width := 3;
    ColorDialog1.Execute;
    Image1.Canvas.Pen.Color := ColorDialog1.Color;
    Image1.Canvas.MoveTo(X1, Y1);
    Image1.Canvas.LineTo(X2, Y2);
end;
```

7. Создать программные коды обработчиков событий рисования закрашенного прямоугольника, закрашенной окружности и очистки области рисования. Для установки цвета контура графической фигуры (цвета пера) и цвета ее заливки (цвета кисти) использовать диалоговое окно *Цвет*, которое вызывается с помощью элемента управления ColorDialog1.

Создадим панель инструментов графического редактора. В панель инструментов должны входить кнопки *Линия*, *Закрашенный прямоугольник*, *Закрашенная окружность* и *Очистить*. Для создания панели инструментов используем элемент управления ToolBar1.

8. Разместить на форме элемент управления ToolBar1.

Создадим четыре кнопки на панели инструментов.

9. Выделить элемент управления ToolBar1, щелкнуть правой кнопкой мыши и в контекстном меню выбрать пункт *New Button*. Прodelать эти действия четыре раза.

На каждой кнопке панели инструментов разместим соответствующее изображение, которое хранится в элементе управления ImageList1.

10. Разместить на форме элемент управления `ImageList1`.
11. Выделить элемент управления `ImageList1`, щелкнуть правой кнопкой мыши и в контекстном меню выбрать пункт *ImageList Editor...*
12. В появившемся окне *Form1.ImageList1 ImageList* (рис. 4.40) добавить четыре изображения для размещения на кнопках панели инструментов, нажав четыре раза кнопку *Add...*

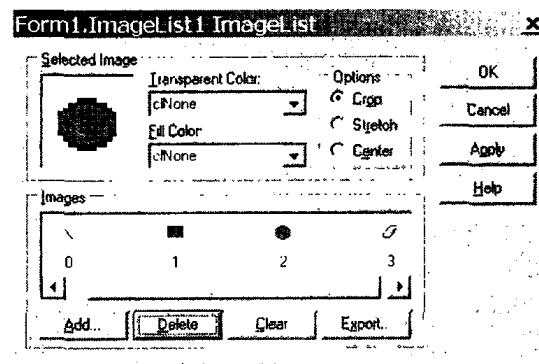


Рис. 4.40. Редактор коллекции изображений для кнопок панели инструментов

Установим соответствие между коллекцией кнопок на панели инструментов `ToolBar1` и коллекцией изображений `ImageList1`.

13. Выделить элемент управления `ToolBar1` и в окне *Object Inspector* для свойства *Image* установить значение *ImageList1*.
14. Последовательно выделить кнопки `ToolButton1`, `ToolButton2`, `ToolButton3` и `ToolButton4` на панели инструментов и в окне *Object Inspector* для свойства *ImageIndex* установить соответствующие порядковые номера изображения в коллекции изображений `ImageList1`.

Создадим программный код обработчиков событий щелчка по кнопке панели инструментов.

15. Ввести программный код обработчика события рисования закрашенного прямоугольника.

```
procedure TForm1.ToolButton2Click(Sender: TObject);
begin
  ColorDialog1.Execute;
```

```

Image1.Canvas.Pen.Color := ColorDialog1.Color;
ColorDialog1.Execute;
Image1.Canvas.Brush.Color := ColorDialog1.Color;
Image1.Canvas.Rectangle(X1, Y1, X2, Y2);
end;

```

16. Создать программные коды обработчиков событий рисования линии, закрашенной окружности и очистки области рисования.

Двойным щелчком по пунктам меню *Файл* последовательно создадим заготовки обработчиков событий и введем их программный код. Для вывода диалоговых окон используем диалоги `OpenDialog1` и `SaveDialog1` и метод `ShowDialog()`.

17. Создать обработчик события открытия растрового графического файла.

```

procedure TForm1.N7Click(Sender: TObject);
begin
  OpenDialog1.Execute;
  Image1.Picture.LoadFromFile(OpenDialog1.FileName);
end;

```

18. Создать обработчик события сохранения растрового графического файла.

```

procedure TForm1.N8Click(Sender: TObject);
begin
  SaveDialog1.Execute;
  Image1.Picture.SaveToFile(SaveDialog1.FileName);
end;

```

19. Запустить проект.

Осуществить нажатие и отпускание кнопки мыши на графическом поле. На надписи будут выведены координаты двух точек.

С использованием меню *Графические примитивы* нарисовать графические фигуры.

С использованием меню *Файл* сохранить рисунок как растровый графический файл.

Проект «Треугольник». Создать проект, который позволяет нарисовать по заданным трем точкам треугольник, вычислить его периметр и площадь. Ввод координат вершин треугольника осуществлять щелчками по графическому полю. Вывод координат вершин треугольника осуществлять методом рисования текста в графическом поле.



Создание графического интерфейса проекта на языке Turbo Delphi

1. Поместить на форму (рис. 4.41):
 - графическое поле Image1, которое будет использоваться в качестве области рисования;
 - две надписи Label1 и Label2 для вывода периметра и площади треугольника;
 - две надписи для вывода поясняющих текстов;
 - меню MainMenu1 для создания меню проекта.
2. Объявить переменные, которые будут содержать координаты вершин треугольника, длины его сторон, а также периметр и площадь.

```
var
  X1, Y1, X2, Y2, X3, Y3: integer;
  L1, L2, L3, P, S: single;
```

3. Создать программный код обработчика события, который обеспечивает запоминание координат щелчков левой, правой и средней кнопками мыши по графическому полю. Использовать три оператора ветвления, которые в зависимости от условия (значения аргумента обработчика события Button) запоминают ту или иную пару координат.

```
procedure TForm1.ImageMouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y:
  Integer);
begin
  if Button = mbLeft
  Then
  begin
    X1:=X;
    Y1:=Y;
  end;
  if Button = mbRight
  Then
  begin
    X2:=X;
    Y2:=Y;
  end;
  if Button = mbMiddle
  Then
  begin
    X3:=X;
    Y3:=Y;
  end;
end;
```

Создадим меню графического редактора, для чего используем элемент управления MainMenu1:

Треугольник
Нарисовать
Периметр
Площадь
Очистить

4. Разместить на форме элемент управления MainMenu1.
5. В появившемся в левом верхнем углу формы в редакторе меню создать заголовок первого уровня. В поле *Прототип для текста* внести пункт меню *Треугольник*.
6. Для создания пункта меню перейти на следующую строку в редакторе меню. Ввести пункты меню *Нарисовать*, *Периметр*, *Площадь* и *Очистить*.

7. В обработчике события пункта меню *Нарисовать* вывести значения координат его вершин и нарисовать стороны треугольника, соединив координаты вершин прямыми линиями.

```

procedure TForm1.N2Click(Sender: TObject);
begin
  With Image1.Canvas Do
  begin
    Pen.Width := 3;
    MoveTo(X1,Y1);
    LineTo(X2,Y2);
    LineTo(X3,Y3);
    LineTo(X1,Y1);
    Image1.Canvas.Font.Name := 'Arial';
    Image1.Canvas.Font.Size := 10;
    Image1.Canvas.TextOut(X1,Y1,IntToStr(X1)+' ','+
    IntToStr(Y1));
    Image1.Canvas.TextOut(X2,Y2,IntToStr(X2)+' ','+
    IntToStr(Y2));
    Image1.Canvas.TextOut(X3,Y3,IntToStr(X3)+' ','+
    IntToStr(Y3));
  end;
end;

```

8. Создать обработчик события, реализующий вычисление периметра треугольника.

```

procedure TForm1.N3Click(Sender: TObject);
begin
  L1 := Sqrt(Sqr(X2 - X1) + Sqr(Y2 - Y1));
  L2 := Sqrt(Sqr(X3 - X1) + Sqr(Y3 - Y1));

```

```

L3 := Sqrt(Sqr(X2 - X3) + Sqr(Y2 - Y3));
P := L1 + L2 + L3;
Label1.Caption := FloatToStr(P);
end;

```

9. Создать обработчик события, реализующий вычисление площади треугольника.

```

procedure TForm1.N4Click(Sender: TObject);
begin
S := Sqrt(P / 2 * (P / 2 - L1) * (P / 2 - L2) *
(P / 2 - L3));
Label2.Caption := FloatToStr(S);
end;

```

10. Создать обработчик события, реализующий очистку графического поля.

```

procedure TForm1.N7Click(Sender: TObject);
begin
With Image1.Canvas Do
begin
Brush.Color := clWhite;
FillRect(Rect(0,0,200,200));
end;
end;

```

11. Запустить проект.
Осуществить щелчки левой, правой и средней кнопками мыши по графическому полю.

С использованием меню *Треугольник* нарисовать треугольник с координатами вершин, вычислить значения периметра и площади треугольника (рис. 4.41).

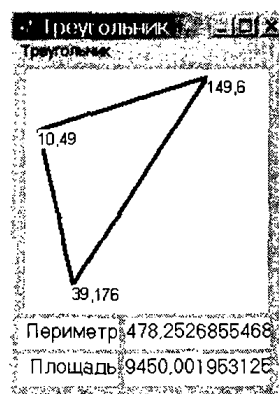



Рис. 4.41. Проект «Треугольник»

Компьютерный практикум



TurboDelphi-CD 

- 4.16. Создать проект «Графический редактор».



...Projects\GraphRedactor\

- 4.17. Создать проект «Треугольник».



...Projects\Triangle\

4.13.3. Компьютерная и математическая системы координат

Компьютерная система координат. Рисование линий, прямоугольников и других фигур производится в компьютерной системе координат, начало которой расположено в верхнем левом углу области рисования. Ось X направлена вправо, а ось Y направлена вниз. Единицей измерения по умолчанию является точка (пиксель). Компьютерная система координат графического поля шириной 300 точек и высотой 200 точек приведена на рис. 4.42.

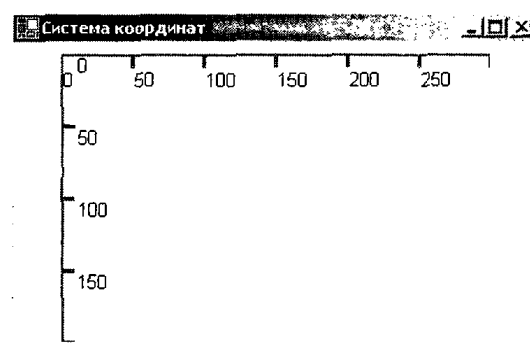


Рис. 4.42. Компьютерная система координат области рисования

Математическая система координат. При геометрических построениях и построении графиков функций удобнее использовать математическую систему координат, начало которой обычно находится в центре области рисования. Ось X направлена вправо, а ось Y направлена вверх. Математическая система координат области рисования шириной 300 точек и высотой 200 точек приведена на рис. 4.43.

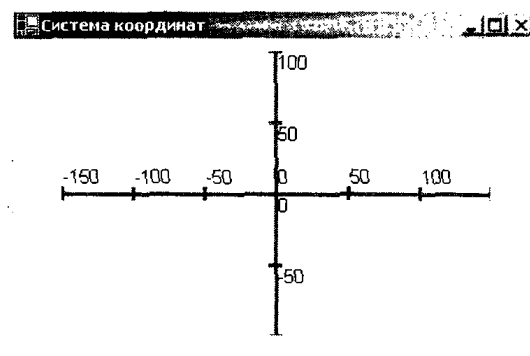


Рис. 4.43. Математическая система координат области рисования

Преобразование компьютерной системы координат в математическую. На языках Visual Basic .NET, Visual C# и Visual J# для преобразования компьютерной системы координат области рисования в математическую систему координат используется метод масштабирования и поворота осей `ScaleTransform()` и метод сдвига начала координат `TranslateTransform()`. Поворот оси Y:



```
Graph1.ScaleTransform(1, -1)
```

Сдвиг по оси X на 150 точек вправо и сдвиг по оси Y на 100 точек вниз:



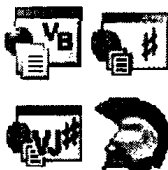
```
Graph1.TranslateTransform(150, -100)
```

В языке Turbo Delphi компьютерная система координат холста остается неизменной, но можно рисовать графики в математической системе координат путем масштабирования и поворота осей, а также сдвига начала координат путем математических преобразований самих координат.

Проект «Система координат». Проект должен обеспечить рисование осей и вывод шкал в компьютерной системе координат (см. рис. 4.42) и математической системе координат (см. рис. 4.43).

2.7.1 — проект «Система координат»
на языке Visual Basic .NET

Информатика-9



Создание графического интерфейса проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

1. Поместить на форму:
 - графическое поле `PictureBox1` (в Turbo Delphi `Image1`), которое будет использоваться в качестве области рисования (холста);
 - кнопки `Button1` и `Button2` для создания обработчиков событий рисования осей и вывода шкал в компьютерной и математической системах координат.
2. Установить размеры графического поля `PictureBox1` (`Image1`): присвоить свойству `Width` значение 300, а свойству `Height` — значение 200.



Объявление переменных и создание обработчика событий рисования осей компьютерной системы координат на языке Visual C#

3. *//Объявление переменных*

```
Graphics Graph1;
Pen Pen1 = new Pen(Color.Red,3);
Font drawFont = new Font("Arial", 12);
Brush drawBrush = new SolidBrush(Color.Black);
```
4. Создать обработчик события рисования осей и вывода шкал в компьютерной системе координат.

```
private void button1_Click(object sender, System.
EventArgs e)
{Graph1 = this.pictureBox1.CreateGraphics();
Graph1.Clear(Color.White);
//Ось X
Graph1.DrawLine(Pen1, 0, 0, 300, 0);
//Ось Y
Graph1.DrawLine(Pen1, 0, 0, 0, 200);
for (int X = 0; X < 300; X+=50)
{//Засечки на оси X
Graph1.DrawLine(Pen1, X, 0, X, 10);
//Вывод шкалы оси X
Graph1.DrawString(Convert.ToString(X), drawFont,
drawBrush, X, 10);
}
for (int Y = 0; Y < 200; Y+=50)
{//Засечки на оси Y
Graph1.DrawLine(Pen1, 0, Y, 10, Y);
//Вывод шкалы оси Y
Graph1.DrawString(Convert.ToString(Y), drawFont,
drawBrush, 10, Y);
}
}
```



Создание обработчика событий рисования осей математической системы координат на языке Visual J#

5.

```
private void button2_Click (Object sender, System.
EventArgs e)
{Graph1 = this.pictureBox1.CreateGraphics();
Graph1.Clear(Color.get_White());
//Вывод шкал математической системы координат в
//компьютерной системе координат
```

```

for (int X = -150; X < 150; X+=50)
{Graph1.DrawString(System.Convert.ToString(X),
drawFont, drawBrush, X + 150, 80);
}
for (int Y = 0; Y <= 200; Y+=50)
{Graph1.DrawString(System.Convert.ToString(Y - 100),
drawFont, drawBrush, 150, 200 - Y);
}
//Преобразование компьютерной системы координат
//в математическую систему координат
Graph1.ScaleTransform(1, -1); //Поворот оси Y
Graph1.TranslateTransform(150, -100); //Сдвиг по
//осям X и Y
//Рисование осей в математической системе
//координат
Graph1.DrawLine(Pen1, -150, 0, 150, 0); //Ось X
Graph1.DrawLine(Pen1, 0, -100, 0, 100); //Ось Y
//Засечки на оси X
for (int X = -150; X < 150; X+=50)
{Graph1.DrawLine(Pen1, X, -5, X, 5);
}
//Засечки на оси Y
for (int Y = -100; Y < 100; Y+=50)
{Graph1.DrawLine(Pen1, -5, Y, 5, Y);
}
}

```



Создание событийной процедуры рисования осей математической системы координат на языке Turbo Delphi

```

5. procedure TForm1.Button2Click(Sender: TObject);
begin
X:=0;
Y:=0;
Image1.Canvas.Pen.Color := clWhite;
Image1.Canvas.Brush.Color := clWhite;
Image1.Canvas.Rectangle(0,0,300,200);
Image1.Canvas.Pen.Color := clRed;
Image1.Canvas.Pen.Width := 3;
//Ось X
Image1.Canvas.MoveTo(0,100);
Image1.Canvas.LineTo(300,100);
//Ось Y
Image1.Canvas.MoveTo(150,0);

```

```

Image1.Canvas.LineTo(150,200);
//Шкала оси X
while X <= 300 Do
begin
Image1.Canvas.MoveTo(X,95);
Image1.Canvas.LineTo(X,105);
Image1.Canvas.TextOut(X,105,IntToStr(X-150));
X := X + 50;
end;
//Шкала оси Y
while Y <= 200 Do
begin
Y := Y + 50;
Image1.Canvas.MoveTo(145,Y);
Image1.Canvas.LineTo(155,Y);
Image1.Canvas.TextOut(155,Y-15,IntToStr(100-Y));
end;
end;

```



Запуск проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

6. Запустить проект.

Осуществить щелчок по первой кнопке, появятся оси со шкалами компьютерной системы координат (см. рис. 4.42). Осуществить щелчок по второй кнопке, появятся оси со шкалами математической системы координат (см. рис. 4.43).

Компьютерный практикум

Windows-CD

4.18. Создать проект «Система координат».



...\informatika10\...\Coordinates\

TurboDelphi-CD



...\Projects\Coordinates\

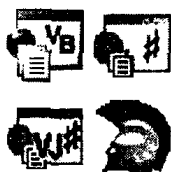
4.13.4. Анимация

Для создания анимации (иллюзии движения на экране какого-либо объекта) применяется принцип смены кадров (изображений), как это делается в мультипликации. Программа, имитирующая движение, должна реализовывать следующие этапы:

- создание изображения в области рисования;
- реализация временной паузы для того, чтобы глаз зафиксировал изображение;
- проведение коррекции изображения.

Анимация часто используется для изображения движения объектов. Для регулирования скорости движения объекта используют пустой цикл, чем большее количество раз он будет выполняться, тем медленнее будет двигаться объект.

Проект «Часы». Создать проект, реализующий работу стрелочных и электронных часов. Стрелки (часовая, минутная и секундная) должны вращаться с использованием эффекта анимации синхронно с системным временем компьютера.



Создание графического интерфейса проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

1. Поместить на форму (рис. 4.44):
 - графическое поле `PictureBox1` (в Turbo Delphi `Image1`), которое будет использоваться в качестве области рисования (холста) для рисования стрелочных часов;
 - надпись `Label1` для вывода времени цифровых часов;
 - таймер `Timer1` для создания обработчика события рисования вывода стрелочных и электронных часов.
2. Установить размеры графического поля `PictureBox1` (`Image1`): присвоить свойствам `Width` и `Height` значение 250.

Для периодического обновления значения времени используем объект `Timer1`. Объект `Timer1` не отображается на форме в процессе выполнения программы и выполняет всего одну функцию — проверяет показания системных часов по событию `Tick`.

Периодичность события `Tick` может быть задана в свойстве `Interval`, измеряемом в миллисекундах (может изменяться от 0 до 65 535). Для того чтобы событие `Tick` происходило

каждую секунду, необходимо свойству Interval присвоить значение 1000.

3. Выделить объект Timer и присвоить свойству Interval значение 1000.



Объявление переменных и создание обработчика события вывода показаний цифровых и стрелочных часов на языке Visual Basic .NET

```
4. Dim XT, YT As Integer 'Координаты цифр
   'на стрелочных часах
   Dim X1, Y1 As Integer 'Координаты секундной 'стрелки
   Dim X2, Y2 As Integer 'Координаты минутной 'стрелки
   Dim X3, Y3 As Integer 'Координаты часовой стрелки
   Dim N As Double
   Dim T As Double, I As Byte 'Счетчики циклов
   Dim Graph1 As Graphics
   Dim Pen1 As New Pen(Color.Red, 1) 'Перо, рисующее
   'секундную стрелку
   Dim Pen2 As New Pen(Color.Green, 2) 'Перо, рисующее
   минутную стрелку
   Dim Pen3 As New Pen(Color.Blue, 3) 'Перо, рисующее
   'часовую стрелку
   Dim drawFont As New Font("Arial", 16)
   Dim drawBrush As New SolidBrush(Color.Black)
```

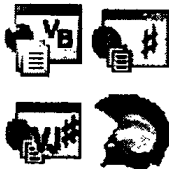
5. В обработчике события Timer1_Tick каждую секунду:
 - выводить значение текущей даты и времени на надпись с использованием объекта DateTime и его свойства Now;
 - рисовать циферблат стрелочных часов с использованием цикла со счетчиком I;
 - рисовать в часовой системе координат секундную, минутную и часовую стрелки, угол поворота которых увеличивается с каждым «тиком» (значение переменной N делится соответственно на 60, 3600 и 21 600);
 - делать паузу с использованием цикла со счетчиком T (пауза должна быть достаточной, чтобы глаз зафиксировал положение стрелок, но не должна превышать времени одного «тика», т. е. 1 секунды);
 - осуществлять стирание рисунка стрелочных часов, чтобы подготовиться к рисованию их положения в следующем «тике» (в следующую секунду).

```
Private Sub Timer1_Tick(ByVal sender As System.
Object, ByVal e As System.EventArgs) Handles
Timer1.Tick
```

```

'Цифровые часы
Label1.Text = DateTime.Now
'Циферблат стрелочных часов
Graph1 = Me.PictureBox1.CreateGraphics()
For I = 1 To 12
XT = 115 + Round(100 * Sin(6.28 * 30 * I / 360))
YT = 115 - Round(100 * Cos(6.28 * 30 * I / 360))
Graph1.DrawString(I, drawFont, drawBrush, XT, YT)
Next I
'Преобразование компьютерной системы координат в
'"часовую" систему координат
Graph1.ScaleTransform(1, -1)
Graph1.TranslateTransform(125, -125)
'Рисование стрелок
N = N + 1
'Секундная стрелка
X1 = Round(90 * Sin(6.28 * N / 60))
Y1 = Round(90 * Cos(6.28 * N / 60))
Graph1.DrawLine(Pen1, 0, 0, X1, Y1)
'Минутная стрелка
X2 = Round(60 * Sin(6.28 * N / 3600))
Y2 = Round(60 * Cos(6.28 * N / 3600))
Graph1.DrawLine(Pen2, 0, 0, X2, Y2)
'Часовая стрелка
X3 = Round(40 * Sin(6.28 * N / 216000))
Y3 = Round(40 * Cos(6.28 * N / 216000))
Graph1.DrawLine(Pen3, 0, 0, X3, Y3)
'Пауза
For T = 1 To 20000000
Next T
'Стирание циферблата и стрелок
Graph1.Clear(Color.White)
End Sub

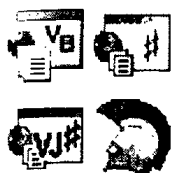
```



Дополнение графического интерфейса
проекта на языках Visual Basic .NET,
Visual C#, Visual J# и Turbo Delphi

6. Поместить на форму (см. рис 4.44):

- элемент управления MonthCalendar1, который будет выводить календарь с выделенной текущей датой;
- элемент управления DateTimePicker1, который будет выводить дату, день недели и время запуска проекта.



Запуск проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

7. Запустить проект (см. рис. 4.44).

- Цифровые часы будут показывать текущие дату и время.
- Стрелочные часы будут показывать время, прошедшее с момента запуска проекта.
- Календарь будет показывать текущую дату.
- Поле вывода даты и времени будет показывать дату и время запуска проекта.

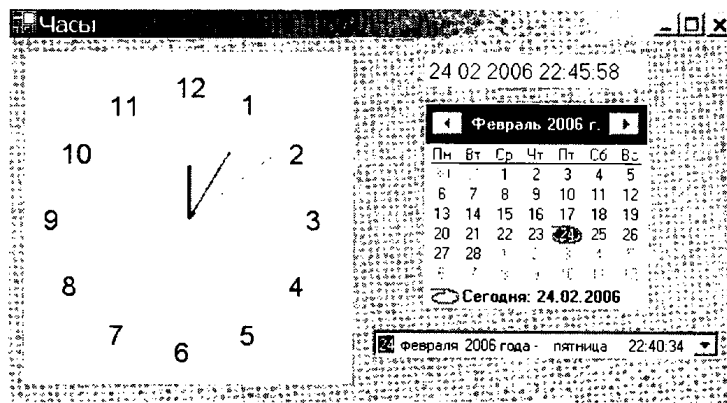


Рис. 4.44. Проект «Часы»

Компьютерный практикум

4.19. Создать проект «Часы».

Windows-CD



...\informatika10...\Clock\

TurboDelphi-CD



...\Projects\Clock\

4.14. Модульный принцип построения решений (групп) и проектов

В системах объектно-ориентированного программирования проекты включают в себя программные модули (программные модули форм и самостоятельные программные модули) и другие файлы. В системе объектно-ориентированного программирования Visual Studio проекты, в свою очередь, входят в состав решений, а в системе Turbo Delphi — в состав групп.

4.3.3. Проекты и приложения

Решения (группы) и проекты. В системе Visual Studio состав решения содержится в файле с расширением sln (по умолчанию в файле WindowsApplication1.sln).

Состав проекта содержится:

- в системе программирования Visual Basic .NET в файле с расширением vbproj (по умолчанию в файле WindowsApplication1.vbproj);
- в системе программирования Visual C# в файле с расширением csproj (по умолчанию в файле WindowsApplication1.csproj);
- в системе программирования Visual J# в файле с расширением vjsproj (по умолчанию в файле WindowsApplication1.vjsproj);

В системе Turbo Delphi состав группы содержится в файле с расширением bdsgroup (по умолчанию в файле ProjectGroup1.bdsgroup).

Состав проекта содержится:

- в системе программирования Turbo Delphi в файле с расширением dpr (по умолчанию в файле Project1.dpr).

Программные модули форм. В системе Visual Studio программный модуль формы содержит в одном файле и описание графического интерфейса (свойства формы и элементов управления), и программный код (обработчики событий, процедуры и функции). Программные модули форм хранятся:

- в системе программирования Visual Basic .NET в файлах с расширением vb (Form1.vb, Form2.vb и т. д.);
- в системе программирования Visual C# в файлах с расширением cs (Form1.cs, Form2.cs и т. д.);
- в системе программирования Visual J# в файлах с расширением jsf (Form1.jsf, Form2.jsf и т. д.);

В системе Turbo Delphi описание графического интерфейса (свойства формы и элементов управления) и программный код (обработчики событий, процедуры и функции) хранятся в двух разных файлах. Описание графического интерфейса хранится в файле с расширением `dfm` (`Unit1.dfm`, `Unit2.dfm` и т. д.), а программный код хранится в файлах с расширением `pas` (`Unit1.pas`, `Unit2.pas` и т. д.).

Программные модули. Самостоятельные программные модули обычно содержат глобальные процедуры и функции, которые многократно вызываются из других модулей проекта. Программные модули хранятся:

- в системе программирования Visual Basic .NET в файлах `Module1.vb`, `Module2.vb` и т. д.;
- в системе программирования Visual C# в файлах `Component1.cs`, `Component2.cs` и т. д.;
- в системе программирования Visual J# в файлах `Component1.jsl`, `Component2.jsl` и т. д.);
- в системе программирования Turbo Delphi в файлах `Unit3.pas`, `Unit4.pas` и т. д.).

Область видимости процедур. Проект может содержать несколько форм и, соответственно, несколько программных модулей форм, а также самостоятельные программные модули. Программный модуль формы, в свою очередь, может состоять из процедур нескольких типов:

- процедур-обработчиков событий, которые выполняются в ответ на событие, вызванное действием пользователя или определенными условиями в программе;
- процедур, которые представляют собой подпрограммы, не возвращающие значение и начинающие выполняться после их вызова из других процедур;
- функций, которые представляют собой подпрограммы, возвращающие значение и входящие в состав выражений.

Процедуры могут быть **локальными** и **глобальными**. Локальная процедура доступна только внутри данного программного модуля и не может быть вызвана из другого модуля. Например, локальная процедура, размещенная в программном модуле одной формы, не может быть вызвана из программного модуля другой формы.

Локальная процедура задается с помощью спецификатора доступа **Private**, который записывается перед именем процедуры-обработчика события, процедуры или функции (в языке Turbo Delphi спецификатор доступа к процедуре указывается

в разделе **type** программного модуля). По умолчанию, если спецификатор доступа отсутствует, процедура является локальной.

Глобальные процедуры доступны, т. е. могут быть вызваны, из всех программных модулей проекта. Глобальная процедура задается с помощью спецификатора доступа **Public**, который записывается перед именем процедуры-обработчика события, процедуры или функции.

Файлы ресурсов. Файл ресурсов проекта (в Visual Studio по умолчанию файл *Form1.resx*, в Turbo Delphi файл *Project1.res*) содержит информацию об использовании внешних ресурсов (например, изображений на форме). Файл ресурсов состоит из записей в формате расширенного языка разметки XML (eXtensible Markup Language), и поэтому его можно открыть в обычном текстовом редакторе. При просмотре файла ресурсов внедренный объект (например, рисунок) отображается в двоичном виде.



Файлы, входящие в решения (группы) и проекты, являются текстовыми файлами, поэтому их просмотр и редактирование можно выполнять в текстовом редакторе (например, в Блокноте).

Компиляция проекта в приложение (рис. 4.45). Решение (группа) или проект, состоящие из текстовых файлов, могут выполняться с использованием интерпретатора системы программирования. В процессе выполнения проекта производится его отладка, информация о которой записывается в базу данных программы (в Visual Studio по умолчанию файл *WindowsApplication1.pdb*).

Для выполнения непосредственно в операционной системе решение или проект должны быть преобразованы путем компиляции в приложение, т. е. в исполнимый двоичный файл. Компиляция решения или проекта выполняется системой программирования автоматически после их запуска на выполнение. В процессе компиляции производится сборка проекта, информация о которой содержится в файле (в Visual Studio по умолчанию файл *AssemblyInfo.**).

Результатом компиляции по умолчанию является файл приложения:

- в системе Visual Studio в папке проекта *WindowsApplication1\bin\WindowsApplication1.exe*;
- в системе Turbo Delphi в папке проекта *Project1\Project1.exe*.

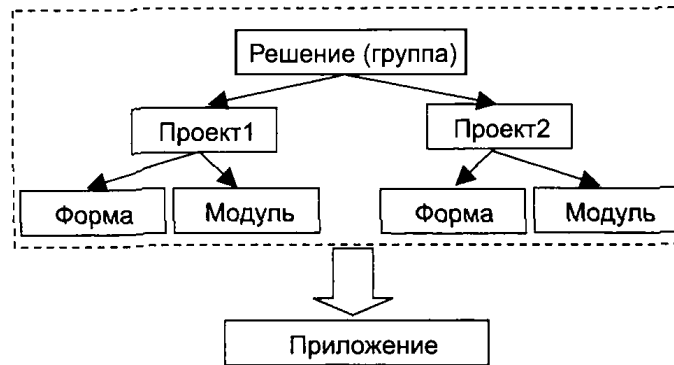


Рис. 4.45. Компиляция решения в приложение

Проект «Домики». Создать проект, состоящий из двух форм и двух программных модулей. На каждую из двух форм должны выводиться рисунки трех домиков путем вызова процедур рисования домиков. Первая процедура рисования домика должна находиться в программном коде первой формы, а вторая процедура — в самостоятельном втором программном модуле. Открытие форм обеспечить из первого самостоятельного программного модуля.



Добавление в проект второй формы и двух программных модулей на языке Visual Basic .NET

1. Ввести команду [Проект-Добавить форму Windows...]. В появившемся диалоговом окне (рис. 4.46) выбрать шаб-

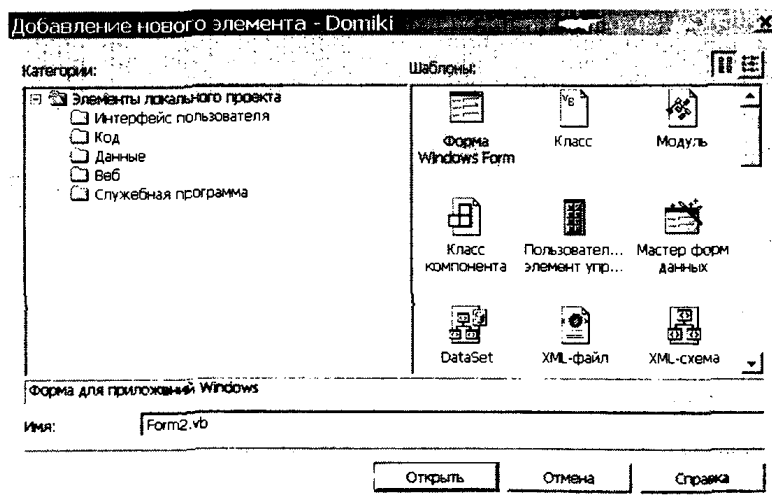


Рис. 4.46. Добавление элемента в проект

лон *Форма Windows Form* и щелкнуть по кнопке *Открыть*.

2. Дважды ввести команду [*Проект-Добавить модуль...*]. В появившемся диалоговом окне выбрать шаблон *Модуль* и щелкнуть по кнопке *Открыть*.
3. В окне *Обозреватель решений* (рис. 4.47) ознакомиться с файлами и ссылками, которые входят в состав решения и проекта *Domiki*:
 - программные модули форм *Form1.vb* и *Form2.vb*;
 - самостоятельные программные модули *Module1.vb* и *Module2.vb*;
 - приложение *Domiki.exe*;
 - база данных отладочной информации *Domiki.pdb*;
 - ресурсы форм *Form1.resx* и *Form2.resx*;
 - сборка *AssemblyInfo.vb*;
 - ссылки на пространства имен, используемые в проекте.

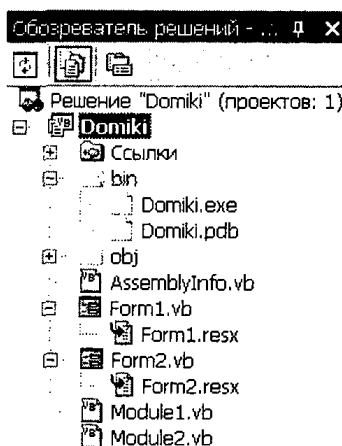


Рис. 4.47. Состав решения *Domiki*

По умолчанию выполнение проекта начинается с загрузки первой формы, но можно начать выполнение проекта с любой формы. В Visual Studio можно начать выполнение проекта и с самостоятельного программного модуля, если в него поместить процедуру (метод) `Main()`.

В проект входят четыре программных модуля (два программных модуля форм и два самостоятельных программных модуля), поэтому необходимо выбрать модуль, с которого начинается выполнение проекта.

4. В окне *Обозреватель решений* выделить имя проекта *Domiki* и в контекстном меню выбрать пункт *Свойства*. В появившемся диалоговом окне (рис. 4.48) из раскрывающегося списка *Начальный объект*: выбрать *Sub Main* и щелкнуть по кнопке *OK*.

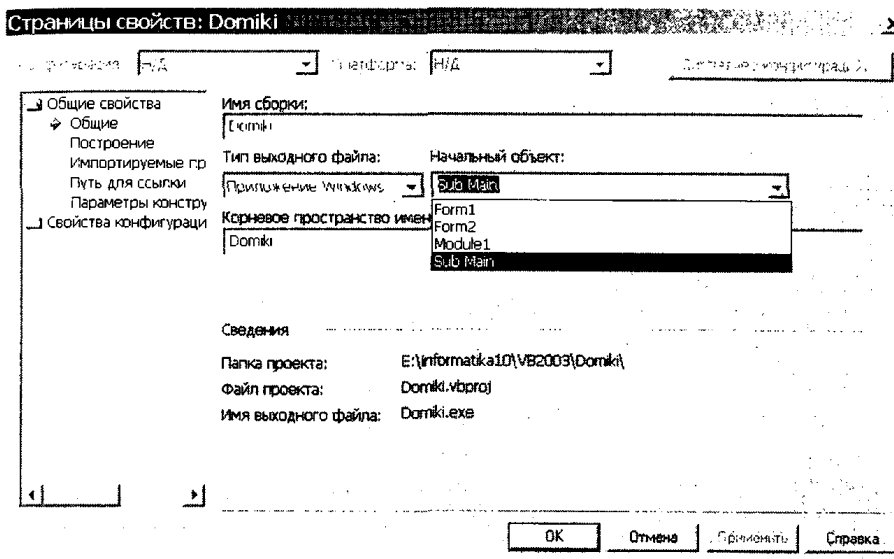
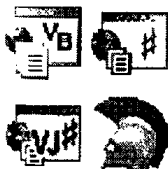


Рис. 4.48. Выбор начального модуля выполнения проекта



Создание графического интерфейса проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

5. Поместить на первую форму (рис. 4.49):
 - графическое поле PictureBox1 (Image1 в Turbo Delphi), которое будет использоваться в качестве области рисования (холста) для рисования домиков;
 - кнопки Button1 и Button2 для создания обработчиков событий.
6. Поместить на вторую форму (см. рис 4.49):
 - графическое поле PictureBox2 (Image2 в Turbo Delphi), которое будет использоваться в качестве области рисования (холста) для рисования домиков;
 - кнопки Button1 и Button2 для создания обработчиков событий.

7. Установить размеры графических полей PictureBox1 (Image1) и PictureBox2 (Image2): присвоить свойствам Width и Height значение 200.

В первом программном модуле объявим глобальные переменные, в которых будут храниться координаты, ширина и высота, а также две области рисования и два пера для рисования домиков на первой и второй формах.

Создадим процедуру Main(), в которой объявим две переменные для идентификации форм, а затем обеспечим их вызов с использованием метода ShowDialog().



Создание программного кода проекта на языке Visual Basic .NET

```
8. Module Module1
  Public X, Y, W, H As Single
  Public Graph1, Graph2 As Graphics
  Public Pen1 As New Pen(Color.Red, 1)
  Public Pen2 As New Pen(Color.Blue, 1)

  Sub Main()
    Dim F1 As New Form1
    Dim F2 As New Form2
    F1.ShowDialog()
    F2.ShowDialog()
  End Sub
End Module
```

Создадим программный модуль первой формы. В процедуре описывается рисование домика из прямоугольника (стена) и двух линий (крыша), а в обработчике события она вызывается три раза с разными наборами параметров.

Для перехода на вторую форму закроем первую форму. Осуществится переход на первый самостоятельный программный модуль, в котором будет выполнена строка программного кода, открывающая вторую форму.

```
9. 'Первая процедура рисования домика
Sub Domik1(ByVal X, ByVal Y, ByVal W, ByVal H)
  Graph1.DrawRectangle(Pen1, X, Y, W, H)
  Graph1.DrawLine(Pen1, X, Y + H, X + W \ 2, Y + H +
  H \ 2)
  Graph1.DrawLine(Pen1, X + W \ 2, Y + H + H \ 2, X
  + W, Y + H)
End Sub
```

```

'Обработчик события, в котором рисуются три домика
Private Sub Button1_Click(ByVal sender As System.
Object, ByVal e As System.EventArgs) Handles
Button1.Click
Graph1 = Me.PictureBox1.CreateGraphics()
Graph1.ScaleTransform(1, -1)
Graph1.TranslateTransform(0, -200)
Call Domik1(10, 20, 60, 50)
Call Domik1(120, 20, 50, 70)
Call Domik1(30, 110, 90, 50)
End Sub

```

```

'Обработчик события, в котором закрывается
'первая форма
Private Sub Button2_Click(ByVal sender As System.
Object, ByVal e As System.EventArgs) Handles
Button2.Click
Me.Close()
End Sub

```

Во втором самостоятельном программном модуле создадим вторую процедуру рисования домика из прямоугольника (стена) и двух линий (крыша).

```

10. 'Вторая процедура рисования домика
Module Module2
Sub Domik2(ByVal X, ByVal Y, ByVal W, ByVal H)
Graph2.DrawRectangle(Pen2, X, Y, W, H)
Graph2.DrawLine(Pen2, X, Y + H, X + W \ 2, Y + H +
H \ 2)
Graph2.DrawLine(Pen2, X + W \ 2, Y + H + H \ 2, X
+ W, Y + H)
End Sub
End Module

```

Создадим программный модуль второй формы. В обработчике события вторая процедура рисования домика, которая находится во втором самостоятельном программном модуле, вызывается три раза с разными наборами параметров.

```

11. 'Обработчик события, в котором рисуются три домика
Private Sub Button1_Click(ByVal sender As System.
Object, ByVal e As System.EventArgs) Handles
Button1.Click
Graph2 = Me.PictureBox2.CreateGraphics()
Graph2.ScaleTransform(1, -1)

```

```

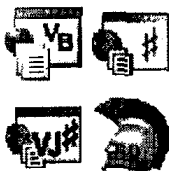
Graph2.TranslateTransform(0, -200)
Call Module2.Domik2(10, 20, 100, 80)
Call Module2.Domik2(120, 20, 50, 50)
Call Module2.Domik2(80, 130, 80, 40)
End Sub

```

```

'Обработчик события, в котором осуществляется
'выход из проекта
Private Sub Button2_Click(ByVal sender As System.
Object, ByVal e As System.EventArgs) Handles
Button2.Click
End
End Sub

```



Запуск проекта на языках Visual Basic
.NET, Visual C#, Visual J# и Turbo Delphi

12. Запустить проект (см. рис. 4.49).

На появившейся первой форме щелкнуть по кнопке *Нарисовать домики*, в области рисования будут нарисованы три разных домика.

Щелкнуть по кнопке *Закреть форму*, на появившейся второй форме щелкнуть по кнопке *Нарисовать домики*, в области рисования будут нарисованы три других разных домика.

Щелкнуть по кнопке *Закреть проект*.

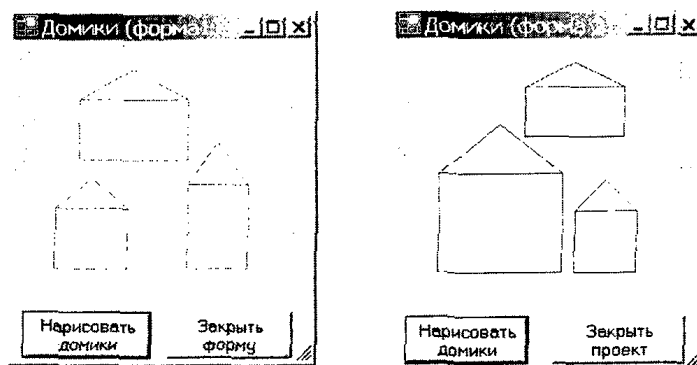


Рис. 4.49. Проект «Домики»


Контрольные вопросы

1. В чем состоит различие между локальными и глобальными процедурами?

Компьютерный практикум




- 4.20. Создать проект «Домики».

Windows-CD 



...informatika10\...\Domiki\

TurboDelphi-CD 



...Projects\Domiki\

4.15. Чтение и запись данных в файлы

По способу доступа к данным различаются файлы **последовательного доступа** и **произвольного доступа**. В файлах последовательного доступа запись или чтение данных осуществляется последовательно от начала до конца файла, а в файлах произвольного доступа — в произвольном порядке. Файлы последовательного доступа состоят из последовательности символов (букв, цифр или других знаков) и часто называются текстовыми файлами.

Языки программирования **Visual Basic .NET**, **Visual C#** и **Visual J#**. В языках программирования **Visual Basic .NET**, **Visual C#** и **Visual J#** для записи последовательности символов (ее часто называют **поток символов** или просто **поток**) в файл последовательного доступа предназначен класс объектов **StreamWriter**. Для чтения последовательности символов из файла последовательного доступа предназначен класс объектов **StreamReader**.

Прежде чем создать объект доступа к данным, необходимо объявить переменную, хранящую в памяти его адрес. Обь-

екты `StreamWriter` и `StreamReader` входят в пространство имен `System.IO`, поэтому необходимо это указать при использовании данных объектов. Это можно сделать непосредственно в строке программного кода `System.IO.StreamWriter` или в начале программного модуля с использованием директив **Imports** `System.IO` в языке Visual Basic .NET, **using** `System.IO` в языке Visual C# и **import** `System.IO.*` в языке Visual J#.

Язык Visual Basic .NET:

```
Dim sw As System.IO.StreamWriter
Dim sr As System.IO.StreamReader
```

Язык Visual C#:

```
using System.IO;
```

```
StreamWriter sw;
StreamReader sr;
```

Язык Visual J#:

```
import System.IO.*;
```

```
StreamWriter sw;
StreamReader sr;
```

Для создания объектов `StreamWriter` и `StreamReader` достаточно в обработчике события присвоить значение соответствующей переменной. Для создания файла используется метод `CreateText()`, а для открытия файла — метод `OpenText()`, аргументом которых является имя создаваемого или открываемого файла. Если файл не хранится в папке проекта, необходимо указать к нему полный путь.

Языки Visual Basic .NET:

```
sw = System.IO.File.CreateText("file_name")
sr = System.IO.File.OpenText("file_name")
```

Язык Visual C# и Visual J#:

```
sw = File.CreateText("file_name");
sr = File.OpenText("file_name");
```

Для записи данных в файл используются методы `Write()` и `WriteLine()`. При использовании метода `Write()` файл запи-

сывается целиком, а при использовании метода `WriteLine()` — построчно (например, из переменной `Line`).

Для чтения данных из файла используются методы `Read()` и `ReadLine()`. При использовании метода `Read()` файл считывается целиком, а при использовании метода `ReadLine()` — построчно (например, в переменную `Line`).



Языки Visual Basic .NET, Visual C# и Visual J#:
`sw.WriteLine(Line)`
`Line = sr.ReadLine()`

После записи или чтения данных из файла необходимо эти файлы закрыть.



Языки Visual Basic .NET, Visual C# и Visual J#:
`sw.Close()`
`sr.Close()`

Язык программирования Turbo Delphi. Файл для записи или чтения должен быть объявлен в разделе описания переменных. Необходимо объявить файловую переменную, хранящую в памяти адрес текстового файла.



`var`
`f: TextFile;`

Объявление файловой переменной задает только тип файла. Для того чтобы можно было выводить данные в файл или считывать данные из файла, необходимо связать имя файловой переменной с именем файла. Если файл не хранится в папке проекта, необходимо указать к нему полный путь, включая имя диска и папок.

Для создания файла используется метод `Rewrite()`, а для открытия файла — метод `Reset()`, аргументом которых является имя файловой переменной, связанной с создаваемым или открываемым файлом.



`AssignFile(f, 'file_name');`
`Rewrite(f);`
`Reset(f);`

Для записи данных в файл используются методы `Write()` и `Writeln()`. При использовании метода `Write()` файл записывается целиком, а при использовании метода `Writeln()` — построчно (например, из переменной `Line`).

Для чтения данных из файла используются методы `Read()` и `Readln()`. При использовании метода `Read()` файл считывается целиком, а при использовании метода `Readln()` — построчно (например, в переменную `Line`).



```
writeln(f, Line);
Readln(f, Line);
```

После записи или чтения данных из файла необходимо эти файлы закрыть.



```
CloseFile(f);
```

4.16.4. Сортировка строковых массивов

Контрольные вопросы

1. Какими способами в языках программирования .NET можно задать пространство имен `System.IO`?
2. В чем состоит различие в использовании методов записи данных в файлы `Write()` и `WriteLine()`, а также методов чтения данных из файлов `Read()` и `ReadLine()`?

4.16. Массивы

4.16.1. Заполнение массивов

Типы массивов и объявление массива. Массив является набором переменных одного типа, объединенных одним именем. Массивы, как и переменные, могут быть различных типов: **числовые, строковые** и т. д.

Массив состоит из пронумерованной последовательности элементов. Номера в этой последовательности называются **индексом**; индекс может принимать целочисленные значения.

Каждый из этих элементов является переменной, т. е. обладает именем и значением, и поэтому массив можно назвать **переменной с индексом**.

Массив может быть одномерным или многомерным. **Размерность** массива соответствует числу индексов, необходимых для идентификации отдельного элемента. Можно задать до 32 индексов, хотя случаи использования более трех индексов очень редки.

Например, одномерный строковый массив, содержащий 33 буквы русского алфавита, можно представить себе в виде табл. 4.5.

Таблица 4.5. Одномерный строковый массив

Индекс	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Элементы массива	а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н	о	п
Индекс	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	
Элементы массива	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я	

Индекс может принимать целочисленные значения от минимального до максимального индекса (в данном массиве от 0 до 32). Каждый элемент массива обладает собственным значением (в данном массиве значением элемента с индексом 4 является строка «д»). Размер массива определяется количеством элементов в массиве (в данном массиве 33 элемента).

Объявление массива производится аналогично объявлению переменных, необходимо только дополнительно указать максимальный индекс или диапазон изменения индекса. Объявим одномерный целочисленный массив, содержащий 10 элементов, и одномерный строковый массив, содержащий 33 элемента.

Язык Visual Basic .NET
Dim N(9) As Byte, B(32) As String



Языки Visual C# и Visual J#
int[] N = new int[10];
String[] B = new String[33];

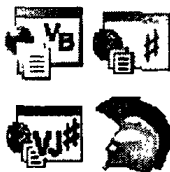
Язык Turbo Delphi
N: array[0..9] of integer;
B: array[0..32] of string;

Обращение к элементу массива производится по его имени, состоящему из имени массива и значения индекса, например $N(5)$.

Заполнение массива. Для начала работы с массивом необходимо его предварительно заполнить, т. е. присвоить элементам массива определенные значения. Это можно сделать различными способами:

- заполнить массив случайными числами;
- заполнить массив символами с клавиатуры;
- заполнить массив из файла.

Проект «Заполнение массива». Заполнить числовой массив десятью случайными числами, строковый массив пятью символами, введенными с клавиатуры, и строковый массив буквами русского алфавита из текстового файла.



Создание графического интерфейса проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

1. Поместить на форму (рис. 4.51):

- список `ListBox1`, который будет использоваться для вывода целочисленного массива, заполненного случайными числами;
- список `ListBox2`, который будет использоваться для вывода строкового массива, заполненного символами с клавиатуры;
- список `ListBox3`, который будет использоваться для вывода строкового массива, заполненного символами алфавита из файла;
- кнопки `Button1`, `Button2` и `Button3` для создания обработчиков событий.

Заполним числовой массив $N(I)$ целыми случайными числами в интервале от 1 до 100.

На языке Visual Basic .NET для генерации последовательности случайных чисел используем функцию `Rnd()`. При запуске программы функция `Rnd()` дает псевдослучайную (т. е. каждый раз повторяющуюся) последовательность чисел в интервале $0 \leq X < 1$.

Для получения последовательности случайных чисел в заданном интервале $A \leq X < B$ необходимо использовать следующую формулу:

$$(B - A) * \text{Rnd}() + A.$$

Тогда получение целочисленной последовательности случайных чисел на интервале $0 \leq X < 100$ достигается использованием функции выделения целой части числа:

```
Int (Rnd() * 100) .
```

Для генерации различающихся между собой последовательностей случайных чисел рекомендуется использовать оператор `Randomize()`. Оператор `Randomize()` использует аргумент для инициализации генератора случайных чисел функции `Rnd()`, задавая его новое начальное значение. Если аргумент опущен, то в качестве нового значения используется значение, возвращаемое системным таймером.



Создание обработчика события заполнения массива случайными числами на языке Visual Basic .NET

```
2. Dim I, N(9) As Byte
```

```
Private Sub Button1_Click(ByVal sender As System.
Object, ByVal e As System.EventArgs) Handles
Button1.Click
Randomize()
ListBox1.Items.Clear()
For I = 0 To 9
N(I) = Int(Rnd() * 100)
ListBox1.Items.Add(Str(N(I)))
Next I
End Sub
```

На языках Visual C# и Visual J# для генерации псевдо-случайных последовательностей используется экземпляр класса `Random` (например, `rnd`). Для генерации случайного числа можно использовать метод `rnd.Next(100)`, который генерирует случайные числа, большие или равные нулю, но меньшие или равные значению аргумента (в данном случае 100). Общий размер массива можно получить с помощью свойства `Length`, которое возвращает количество элементов в массиве.



Создание обработчика события заполнения массива случайными числами на языках Visual C# и Visual J#

```
2. Random rnd = new Random();
int[] N = new int[10];
int I;
```

```

private void button1_Click(object sender,
System.EventArgs e)
{listBox1.Items.Clear();
  for (I = 0; I < N.Length; I++)
  {N[I]= rnd.Next(100);
  listBox1.Items.Add(Convert.ToString(N[I]));
  }
}

```

На языке Turbo Delphi для генерации псевдослучайных последовательностей используется функция `Random()`. Эта функция генерирует случайные числа, большие или равные нулю, но меньшие или равные значению аргумента. Для генерации различающихся между собой последовательностей случайных чисел можно использовать оператор `Randomize`.



Создание обработчика события заполнения массива случайными числами на языке Turbo Delphi

2. var

```

N: array[0..9] of integer;
I: integer;

```

```

procedure TForm1.Button1Click(Sender: TObject);
begin
ListBox1.Items.Clear();
Randomize;
For I := 0 To 9 Do
begin
N[I] := Random(100);
ListBox1.Items.Add(IntToStr(N[I]));
end;
end;

```

Для ввода элементов массива с клавиатуры можно использовать функцию `InputBox()`. Аргументами этой функции являются две строки, которые отображаются в диалоговом окне ввода, а значением функции — символ, введенный пользователем.





Создание обработчика события заполнения массива с клавиатуры на языке Visual Basic .NET

3. Dim A(4) As String

```

Private Sub Button2_Click(ByVal sender As System.
Object, ByVal e As System.EventArgs) Handles
Button2.Click
For I = 0 To 4
A(I) = InputBox("Введите элемент массива", "Ввод
элементов массива с клавиатуры")
ListBox2.Items.Add(A(I))
Next I
End Sub

```



Создание обработчика события заполнения массива с клавиатуры на языке Turbo Delphi

3. var

```

A: array[0..4] of string;

procedure TForm1.Button2Click(Sender: TObject);
begin
ListBox2.Items.Clear();
For I := 0 To 4 Do
begin
A[I] := InputBox('Ввод элементов массива с клави-
атуры', 'Введите элемент массива', '');
ListBox2.Items.Add(A[I]);
end;
end;

```

На языках Visual C# и Visual J# для ввода элементов массива с клавиатуры с использованием функции InputBox() необходимо подключить и использовать пространство имен Microsoft.VisualBasic.



Создание обработчика события заполнения массива с клавиатуры на языках Visual C# и Visual J#

3.1. Ввести команду [Проект-Добавить ссылку...].

В появившемся диалоговом окне *Add Reference* на вкладке *.NET* из списка выбрать имя компонента *Microsoft.VisualBasic.NET Runtime* и щелкнуть по кнопке *Выбрать*. Выбранный компонент добавится в поле *Выбранные компоненты*:

Щелкнуть по кнопке *ОК*.

3.2. В *Обозревателе решений* в проекте *array1* в разделе ссылок *References* добавится ссылка *Microsoft.VisualBasic* на соответствующее пространство имен.

Теперь можно добавить ссылку на пространство имен и обработчик события в программный код. Необходимо учесть, что параметрами функции `InputBox()` также являются значение функции по умолчанию и координаты верхнего левого угла диалогового окна функции.

3.3. `using Microsoft.VisualBasic;`

```
...
String[] A = new String[5];

private void button2_Click(object sender,
System.EventArgs e)
{for (I = 0; I < 5; I++)
  {A[I] = Interaction.InputBox("Введите элемент
массива", "Ввод элементов массива с клавиатуры",
"", 500, 500);
  listBox2.Items.Add(A[I]);
}
}
```

Заполним строковый массив буквами русского алфавита из текстового файла. Создадим текстовый файл, содержащий буквы русского алфавита. Такие текстовые файлы должны содержать только коды самих символов (не должны содержать управляющие коды форматирования текста, тэги языка HTML и т. д.) и, следовательно, должны создаваться в простейших текстовых редакторах типа Блокнот (рис. 4.50).

4.15. Чтение и запись данных в файлы

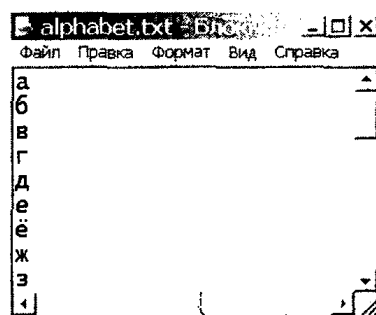


Рис. 4.50. Текстовый файл, содержащий русский алфавит

4. Создать в текстовом редакторе файл и ввести буквы русского алфавита по одной букве в строке (рис. 4.50). Сохранить файл `alphabet.txt` в кодировке *Unicode*.

На языках Visual Basic .NET, Visual C# и Visual J# объявим массив и переменную, хранящую в памяти адрес объекта `StreamReader`.

В обработчике события присвоим переменной значение с использованием метода открытия файла `OpenText()`, аргументом которого является имя открываемого файла `alphabet.txt`. В цикле со счетчиком произведем чтение букв алфавита из файла с использованием метода `ReadLine()` и выведем буквы в список.



Создание обработчика события заполнения массива из файла на языке Visual Basic .NET

```
5. Dim I As Byte, B(32) As String
   Dim sr As System.IO.StreamReader

   Private Sub Button3_Click(ByVal sender As System.
Object, ByVal e As System.EventArgs) Handles
Button3.Click
   sr = System.IO.File.OpenText("alphabet.txt")
   For I = 0 To 32
   B(I) = sr.ReadLine()
   ListBox3.Items.Add(B(I))
   Next I
   End Sub
```



Создание обработчика события заполнения массива из файла на языках Visual C# и Visual J#

```
5. String[] B = new String[33];
   StreamReader sr;

   private void button2_Click(object sender, System.
EventArgs e)
   {sr = File.OpenText("alphabet.txt");
   for (I = 0; I < 33; I++)
   {B[I] = sr.ReadLine();
   ListBox3.Items.Add(B[I]);
   }
   }
```


На языке Turbo Delphi объявим массив и файловую переменную.

В событийной процедуре с помощью метода AssignFile() свяжем имя файловой переменной с именем файла. Для открытия файла используем метод Reset(), аргументом которого является имя файловой переменной. В цикле со счетчиком произведем чтение букв алфавита из файла с использованием метода Readln() и выведем буквы в список.



Создание событийной процедуры заполнения массива из файла на языке Turbo Delphi

5. var

```
B: array[0..32] of string;
f: TextFile;
```

```
procedure TForm1.Button3Click(Sender: TObject);
begin
  ListBox3.Items.Clear();
  AssignFile(f, 'alphabet.txt');
  Reset(f);
  For I := 0 To 32 Do
  begin
    Readln(f, B[I]);
    ListBox3.Items.Add(B[I]);
  end;
end;
```



Запуск проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi



6. Запустить проект (см. рис. 4.51).

Щелкнуть по кнопке *Заполнение массива случайными числами*, в первый список будут выведены элементы массива, заполненного случайными числами.

Щелкнуть по кнопке *Заполнение массива с клавиатуры*, во второй список будут выведены элементы массива, заполненного символами, введенными с клавиатуры.

Щелкнуть по кнопке *Заполнение массива из файла*, в третий список будут выведены элементы массива, заполненного буквами русского алфавита из файла. Так как не все элементы массива помещаются в список, появляется вертикальная полоса прокрутки.

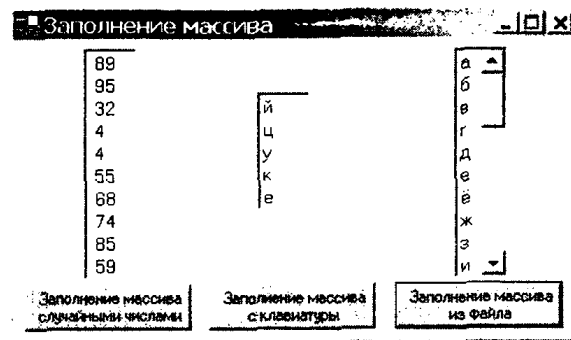


Рис. 4.51. Проект «Заполнение массива»

Компьютерный практикум




Windows-CD 

4.21. Создать проект «Заполнение массива».



...\informatika10\...\Array1\

TurboDelphi-CD 



...\Projects\Array1\

4.16.2. Поиск элемента в массивах

Поиск в строковых массивах обычно реализуется в форме поиска индекса элемента массива, для которого значение элемента совпадает с заданным. В числовых массивах обычно производится поиск минимального или максимального элемента.

Проект «Поиск в массиве». Разработать проект, в котором числовой массив, содержащий 10 элементов, заполняется случайными числами в диапазоне от 0 до 100. Осуществить в этом числовом массиве поиск максимального элемента и его индекса.



Создание графического интерфейса проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

1. Поместить на форму (рис. 4.52):

- список `ListBox1` для вывода значений индекса элементов массива;
- список `ListBox2` для вывода значений элементов массива;
- надпись `Label1` для вывода индекса максимального элемента;
- надпись `Label2` для вывода значения максимального элемента;
- кнопку `Button1` для создания обработчика события;
- две надписи для вывода поясняющих текстов.

Создадим обработчик события для заполнения целочисленного массива случайными числами и поиска максимального элемента и его индекса:

- 1) Объявим целочисленные переменные `I` и `Max`, а также целочисленный массив `A(9)`, содержащий десять элементов.
- 2) Заполним массив целыми случайными числами от 0 до 100.
- 3) Будем считать, что сначала максимальный элемент равен первому элементу массива, т. е. $Max = 0$ и $A(Max) = A(0)$.
- 4) Затем в цикле со счетчиком `I` с использованием оператора ветвления последовательно сравним максимальный элемент массива (элемент массива с индексом 0) со всеми остальными элементами. Если какой-либо элемент окажется больше, присвоим переменной `Max` индекс этого элемента.
- 5) Результат поиска, т. е. индекс максимального элемента и сам максимальный элемент, выведем на надписи.



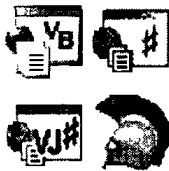
Создание обработчика события на языке Visual Basic .NET

2. `Dim A(9), I, Max As Byte`

```
Private Sub Button1_Click(ByVal sender As System.
Object, ByVal e As System.EventArgs) Handles
Button1. Click
ListBox1.Items.Clear()
ListBox2.Items.Clear()
'Заполнение массива
Randomize()
```

```

For I = 0 To 9
A(I) = Int(Rnd() * 100)
ListBox1.Items.Add(I)
ListBox2.Items.Add(A(I))
Next I
'Поиск максимального элемента
Max = 0
A(Max) = A(0)
For I = 1 To 9
If A(I) > A(Max) Then A(Max) = A(I) : Max = I
Next I
Label1.Text = Max
Label2.Text = A(Max)
End Sub
    
```



Запуск проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

3. Запустить проект (см. рис. 4.52). Щелкнуть по кнопке *Заполнить и найти* несколько раз. На надписи будут выводиться результаты поиска индекса максимального элемента и сам максимальный элемент для различных вариантов заполнения массива.

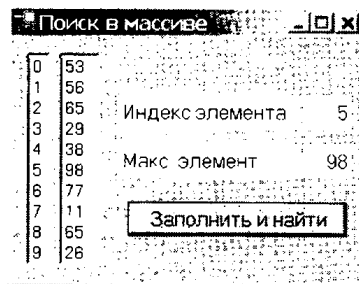


Рис. 4.52. Проект «Поиск в массиве»

Компьютерный практикум



4.22. Создать проект «Поиск в массиве».

Windows-CD



...\Informatika10\...larrayMax\

TurboDelphi-CD



...\Projects\larrayMax\

В таблице каждый шаг сортировки массива разместим в вертикальном столбце, отображающем порядок элементов массива. Перестановку элементов массива отобразим стрелками.

Проект «Сортировка числового массива». Заполнить числовой массив, содержащий 10 элементов, случайными числами в диапазоне от 0 до 100. Осуществить сортировку числового массива по убыванию. Поиск максимального элемента осуществить различными способами:

- с помощью процедуры;
- с помощью функции;
- во вложенном цикле.



Создание графического интерфейса проекта на языках Visual Basic .NET, Visual C# и Visual J#



1. Поместить на форму (рис. 4.53):

- список ListBox1, который будет использоваться для вывода значений индекса целочисленного массива;
- список ListBox2, который будет использоваться для вывода числового массива, заполненного случайными числами;
- списки ListBox3 — ListBox11, которые будут использоваться для вывода шагов сортировки массива;
- кнопки Button1 и Button2 для создания обработчиков событий заполнения массива и сортировки;
- надпись Label1 для вывода нумерации шагов сортировки.

На языке Visual Basic .NET будем искать максимальный элемент с использованием процедуры. Программный модуль проекта будет содержать:

- обработчик события заполнения массива случайными числами;
- обработчик события сортировки массива;
- процедуру поиска максимального элемента.



Создание программного кода проекта на языке Visual Basic .NET

Объявим переменные для использования в программном модуле и создадим обработчик события заполнения целочисленного массива случайными числами.

2. Dim A(9), I, J, K, R, Max As Byte

```

Private Sub Button1_Click(ByVal sender As System.
Object, ByVal e As System.EventArgs) Handles
Button1.Click
'Очистка полей списков индексов элементов
'и значений элементов массива
ListBox1.Items.Clear()
ListBox2.Items.Clear()
'Заполнение массива случайными числами
Randomize()
For K = 0 To 9
A(K) = Int(Rnd() * 100)
ListBox1.Items.Add(K)
ListBox2.Items.Add(Str(A(K)))
Next K
End Sub

```

Процедура поиска максимального элемента МаксЭлемент (**ByVal** I, **ByRef** Max) будет многократно вызываться из обработчика события сортировки и будет содержать входной и выходной параметры:

I — входной параметр (индекс максимального элемента массива после предыдущего шага);

Max — выходной параметр (индекс максимального элемента массива после данного шага).

```

3. 'Процедура поиска индекса максимального элемента
'массива
Sub МаксЭлемент(ByVal I, ByRef Max)
Max = I
For J = I + 1 To 9
If A(J) > A(Max) Then Max = J
Next J
End Sub

```

Создадим обработчик события сортировки массива, который будет включать:

- очистку полей списков шагов сортировки;
- цикл со счетчиком I, в котором осуществляется пошаговая сортировка массива по убыванию и который, в свою очередь, будет включать:
 - вызов процедуры поиска индекса максимального элемента;
 - перестановку элементов массива;

вывод массива для каждого шага сортировки в списке с использованием цикла со счетчиком и оператора выбора.

```

4. Private Sub Button2_Click_1(ByVal sender As System.
Object, ByVal e As System.EventArgs) Handles
Button2.Click
'Очистка полей списков шагов сортировки
ListBox3.Items.Clear()
...
ListBox11.Items.Clear()
Label1.Text = " "
'Сортировка
For I = 0 To 8
Label1.Text = Label1.Text + " " + Str(I + 1) + " "
'Вызов процедуры поиска индекса максимального
'элемента МаксЭлемент(I, Max)
'Перестановка элементов массива
R = A(I)
A(I) = A(Max)
A(Max) = R
'Вывод массива для каждого шага сортировки
For K = 0 To 9
Select Case I
Case 0
ListBox3.Items.Add(Str(A(K)))
...
Case 8
ListBox11.Items.Add(Str(A(K)))
End Select
Next K
Next I
End Sub

```

На языке Visual J# максимальный элемент будем искать с использованием функции. Программный модуль проекта будет содержать:

- обработчик события заполнения массива случайными числами;
- обработчик события сортировки массива;
- функцию поиска максимального элемента.



Создание программного кода проекта на языке Visual J#

Объявим переменные для использования в программном модуле и создадим обработчик события заполнения целочисленного массива случайными числами.


```

2. Random rnd = new Random();
   int[] A = new int[10];
   int I, J, Max, K, R;

   private void button1_Click (Object sender, System.
   EventArgs e)
   {listBox1.get_Items().Clear();
     listBox2.get_Items().Clear();
     //Заполнение массива
     for (K = 0; K <= 9; K++)
     {A[K]=rnd.Next(100);
      listBox1.get_Items().Add(System.Convert.
        ToString(K));
      listBox2.get_Items().Add(System.Convert.
        ToString(A[K]));
     }
   }

```

Функция поиска максимального элемента `MaxEl(int I)` будет многократно вызываться из обработчика события сортировки. Функция содержит входной параметр `I` (индекс максимального элемента массива после предыдущего шага). Возвращаемое функцией значение — `result` (индекс максимального элемента массива после данного шага).

```

3. //Поиск максимального элемента
   int MaxEl(int I)
   {int result;
     result = I;
     for (J = I+1; J <= 9; J++)
     {if (A[J] > A[result])
      {result = J;
      }
     }
   }
   return result;
}

```

Создадим обработчик события сортировки массива.

```

4. private void button2_Click (Object sender, System.
   EventArgs e)
   {//Очистка полей списков шагов сортировки
     listBox3.get_Items().Clear();
     ...
     listBox11.get_Items().Clear();
     label1.set_Text("");
   }
   //Сортировка

```

```

for (I = 0; I <= 8; I++)
{label1.setText(label1.getText() + " " + System.
Convert.ToString(I+1) + " ");
//Вызов процедуры поиска номера максимального
//элемента
Max = MaxEl(I);
//Перестановка элементов массива
R = A[I];
A[I] = A[Max];
A[Max] = R;
//Вывод массива для каждого шага сортировки
for (K = 0; K <= 9; K++)
{switch (I)
{case
0: listBox3.getItems().Add(System.Convert.
ToString(A[K]));
break;
...
case 8: listBox11.getItems().Add(System.Convert.
ToString(A[K]));
break;
}
}
}
}

```

На языке Visual C# максимальный элемент будем искать с использованием вложенного цикла. Программный модуль проекта будет содержать:

- обработчик события заполнения массива случайными числами;
- обработчик события сортировки массива, в котором поиск максимального элемента осуществляется во вложенном цикле.



Создание программного кода проекта на языке Visual C#

Объявим переменные для использования в программном модуле и создадим обработчик события заполнения целочисленного массива случайными числами.

```

2. Random rnd = new Random();
int[] A = new int[10];
int I, J, Max, K, R;

```

```

private void button1_Click(object sender, System.
EventArgs e)

```

```

{label1.Text ="";
 listBox1.Items.Clear();
 listBox2.Items.Clear();
//Заполнение массива
for (K = 0; K < A.Length; K++)
{A[K]=rnd.Next(100);
 listBox1.Items.Add(Convert.ToString(K));
 listBox2.Items.Add(Convert.ToString(A[K]));
}
}

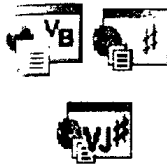
```

Создадим обработчик события сортировки массива.

```

4. private void button2_Click(object sender, System.
EventArgs e)
{//Очистка полей списков шагов сортировки
listBox3.Items.Clear();
...
listBox11.Items.Clear();
//Сортировка
for (I = 0; I < A.Length-1; I++)
{label1.Text = label1.Text + " " + Convert.
ToString(I+1) + " ";
//Поиск максимального элемента
Max = I;
for (J = I+1; J < A.Length; J++)
{if (A[J] > A[Max])
{Max = J;
}
}
//Перестановка элементов массива
R = A[I];
A[I] = A[Max];
A[Max] = R;
//Вывод массива для каждого шага сортировки
for (K = 0; K < A.Length; K++)
{switch (I)
{case 0: listBox3.Items.Add(Convert.ToString(A[K]));
break;
...
case 8: listBox11.Items.Add(Convert.ToString(A[K]));
break;
}
}

```



Запуск проекта на языках Visual Basic .NET, Visual C# и Visual J#

5. Запустить проект (см. рис. 4.53). Щелкнуть по кнопке *Заполнить массив*. В первый список будут выведены значения индекса массива, а во второй список — значения элементов массива — случайные числа. Щелкнуть по кнопке *Сортировать*. В списках с третьего по одиннадцатый будет реализована пошаговая визуализация процесса сортировки числового массива. На надпись будут выведены номера шагов сортировки.



Рис. 4.53. Проект «Сортировка числового массива»



Создание графического интерфейса проекта на языке Turbo Delphi

1. Поместить на форму (рис. 4.54):
 - таблицу StringGrid1, которая будет использоваться для вывода значений индекса целочисленного массива, номеров шагов сортировки, числового массива, заполненного случайными числами, а также состояния массива после каждого шага сортировки;
 - кнопки Button1 и Button2 для создания обработчиков событий заполнения массива и сортировки.

Программный модуль проекта будет содержать:

- обработчик события заполнения массива случайными числами;

- обработчик события сортировки массива, в котором поиск максимального элемента осуществляется во вложенном цикле.

Объявим переменные для использования в программном модуле и создадим обработчик события заполнения целочисленного массива случайными числами.

2. var

```
A: array[1..10] of integer;
I: integer;
J: integer;
K: integer;
R: integer;
Max: integer;
```

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  //Заполнение массива
  Randomize;
  for i:=0 to 9 do
    begin
      for j:=0 to 9 do
        begin
          StringGrid1.Cells[i+2,0]:= IntToStr(i+1);
          StringGrid1.Cells[0,j+1]:= IntToStr(j);
          A[j] := Random(100);
          StringGrid1.Cells[1,j+1]:= IntToStr(A[j]);
        end;
      end;
    end;
end;
```

Создадим обработчик события сортировки массива.

```
4. procedure TForm1.Button2Click(Sender: TObject);
begin
  //Сортировка
  For I := 0 To 8 Do
    begin
      //Поиск индекса максимального элемента
      Max := I;
      For J := I + 1 To 10 Do
        begin
          If A[J] > A[Max]
          Then Max := J;
        end;
      //Перестановка
```

```

R := A[I];
A[I] := A[Max];
A[Max] := R;
//Вывод массива в таблицу для каждого шага
//сортировки
For K := 0 To 9 Do
StringGrid1.Cells[i+2,k+1]:= IntToStr(A[k]);
end;
end;

```

5. Запустить проект (см. рис. 4.54).

Щелкнуть по кнопке *Заполнить массив*. В нулевой столбец таблицы будут выведены значения индекса массива, в нулевую строку — нумерация шагов сортировки, а в первый столбец — значения элементов массива (случайные числа). Щелкнуть по кнопке *Сортировать массив*. В столбцах таблицы будет реализована пошаговая визуализация процесса сортировки числового массива.



Рис. 4.54. Проект «Сортировка числового массива» на языке Turbo Delphi


Компьютерный практикум

4.23. Создать проект «Сортировка числового массива».

Windows-CD



...\\informatika10\...\arrayByteSort*

TurboDelphi-CD 

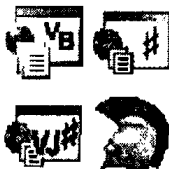
...\Projects\arrayByteSort\

4.16.4. Сортировка строковых массивов

Под сортировкой строкового массива понимается процесс перестановки значений элементов массива, целью которого является размещение строковых значений элементов массива по возрастанию (или по убыванию) (сначала по первым символам, затем по вторым и т. д.).

При сортировке строкового массива по возрастанию его элементы располагаются в прямом порядке (числа, латинский алфавит от А до Z, русский алфавит от А до Я), а при сортировке по убыванию — в обратном порядке (русский алфавит от Я до А, латинский алфавит от Z до А, числа).

Проект «Сортировка строкового массива». Заполнить строковые массивы буквами русского алфавита и компьютерными терминами из текстовых файлов. Отсортировать массив, содержащий компьютерные термины, по возрастанию и сохранить его в текстовом файле.



Создание графического интерфейса проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

1. Поместить на форму (рис. 4.55):

- надпись `Label1`, которая будет использоваться для вывода элементов строкового массива, заполненного из текстового файла буквами русского алфавита;
- список `ListBox1`, который будет использоваться для вывода строкового массива, заполненного из текстового файла компьютерными терминами;
- список `ListBox2`, который будет использоваться для вывода отсортированного строкового массива, содержащего компьютерные термины;
- кнопки `Button1` — `Button5` для создания обработчиков событий заполнения строковых массивов, сортировки, вывода и сохранения отсортированного массива компьютерных терминов.

Создадим текстовые файлы, содержащие буквы русского алфавита и компьютерные термины.

2. Создать в текстовом редакторе файл и ввести буквы русского алфавита по одной букве в строке. Сохранить файл `alphabet.txt` в кодировке *Unicode*.
3. Создать в текстовом редакторе файл и ввести компьютерные термины по одному в строке. Сохранить файл `term.txt` в кодировке *Unicode*.



Создание программного кода проекта на языке Visual Basic .NET

Объявим переменные для использования в программном модуле:

- счетчики циклов;
- строковый массив, состоящий из 33 элементов, значениями которых будут буквы русского алфавита;
- строковый массив, состоящий из 10 элементов, значениями которых будут компьютерные термины до сортировки;
- строковый массив, состоящий из 10 элементов, значениями которых будут компьютерные термины после сортировки.

4. `Dim I, K, J As Byte, A(32), B(9), C(9) As String`

Объявим переменную, хранящую в памяти адрес объекта `StreamReader`. Создадим обработчик события заполнения строкового массива из текстового файла, содержащего буквы русского алфавита. ~

5. `Dim sra As StreamReader`

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    TextBox1.Text = ""
    sra = File.OpenText("alphabet.txt")
    For I = 0 To 32
        A(I) = sra.ReadLine()
        TextBox1.Text = TextBox1.Text + A(I)
    Next I
    sra.Close()
End Sub
```

Объявим переменную, хранящую в памяти адрес объекта `StreamReader`. Создадим обработчик события заполнения строкового массива из текстового файла, содержащего компьютерные термины.


```

6. Dim srt As StreamReader
   Private Sub Button2_Click(ByVal sender As System.
Object, ByVal e As System.EventArgs) Handles
Button2.Click
   srt = File.OpenText("term.txt")
   For K = 0 To 9
   B(K) = srt.ReadLine()
   ListBox1.Items.Add(B(K))
   Next K
   srt.Close()
End Sub

```

Создадим обработчик события сортировки массива. В исходном строковом массиве B[] компьютерные термины хранятся в неупорядоченном виде. Сравним последовательно во внутреннем цикле со счетчиком K букву «а» алфавита с первыми символами терминов, в случае их совпадения присвоим первому элементу результирующего строкового массива C[] значение, равное этому термину. Повторим эту процедуру последовательно для всех букв алфавита с записью в очередной элемент массива C[]. Получим упорядоченный строковый массив C[].



Создание обработчика события сортировки строкового массива на языках Visual C# и Visual J#

```

7. private void button3_Click(object sender, System.
EventArgs e)
{J = 0;
 for (I = 0; I < 33; I++)
 {for (K = 0; K < 10; K++)
 {if (A[I] == B[K].Substring(0,1))
 {C[J] = B[K];
 J = J + 1;
 }
 }
 }
}
}

```

Создадим обработчик события вывода отсортированного строкового массива, содержащего компьютерные термины, в список.

```

8. private void button4_Click(object sender, System.
EventArgs e)
{for (K = 0; K < 10; K++)

```

```

    {
      listBox2.Items.Add(C[K]);
    }
  }

```

Создадим обработчик события сохранения отсортированного строкового массива, содержащего компьютерные термины, в текстовом файле.



Создание обработчика события сохранения отсортированного строкового массива на языке Turbo Delphi

9. var

```
f3: TextFile;
```

```

procedure TForm1.Button5Click(Sender: TObject);
begin
  AssignFile(f3, 'term1.txt');
  Rewrite(f3);
  For I := 1 To 10 Do
  begin
    writeln(f3, C[I]);
  end;
  CloseFile(f3);
end;

```



Запуск проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi



10. Запустить проект (см. рис. 4.55).

Щелкнуть по кнопке *Считать алфавит*. На надпись будут выведены буквы русского алфавита.

Щелкнуть по кнопке *Считать термины*. В первый список будут выведены неотсортированные компьютерные термины.

Щелкнуть по кнопке *Сортировать*. Компьютерные термины будут отсортированы.

Щелкнуть по кнопке *Показать термины*. Во второй список будут выведены отсортированные компьютерные термины.

Щелкнуть по кнопке *Записать в файл*. В текстовом файле будут сохранены отсортированные компьютерные термины.

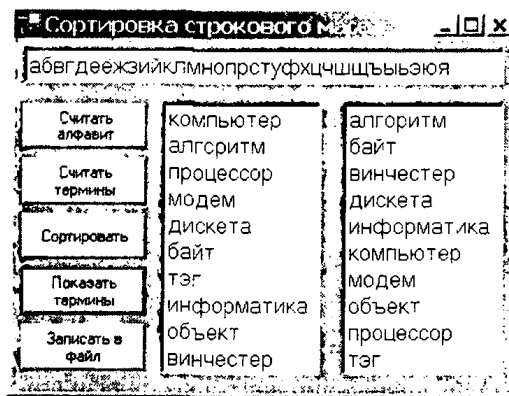


Рис. 4.55. Проект «Сортировка строкового массива»

Компьютерный практикум

Windows-CD

4.24. Создать проект «Сортировка строкового массива».



...\informatika10\...\arrayStringSort\

TurboDelphi-CD



...\Projects\arrayStringSort\

Глава 5

Практические задания для тематического и итогового контроля

Тема «Архитектура компьютера и защита информации»

Практическая работа 1.1

Создание логического диска и его форматирование

Задание. В операционной системе Windows произвести создание логического диска и его форматирование в определенной файловой системе.

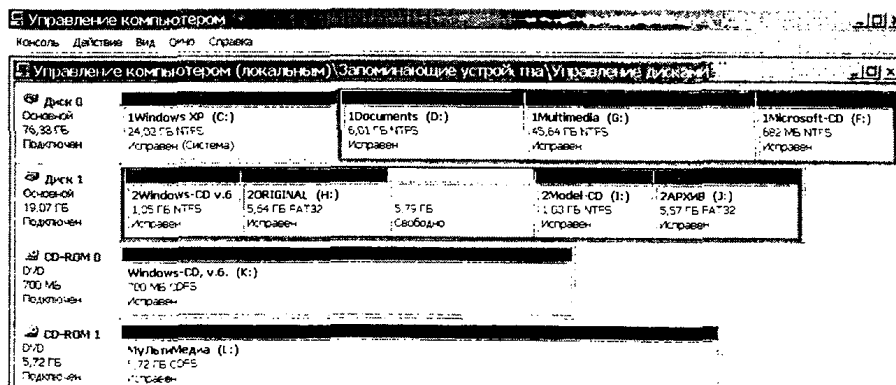
Варианты выполнения работы:

- создание логического диска с использованием различных файловых систем (FAT32 или NTFS) и различных размеров кластера.



Создание логического диска и его форматирование

1. Ввести команду [Программы-Администрирование-Управление компьютером]. В появившемся диалоговом окне *Управление компьютером* в меню консоли выбрать пункт *Управление дисками*.

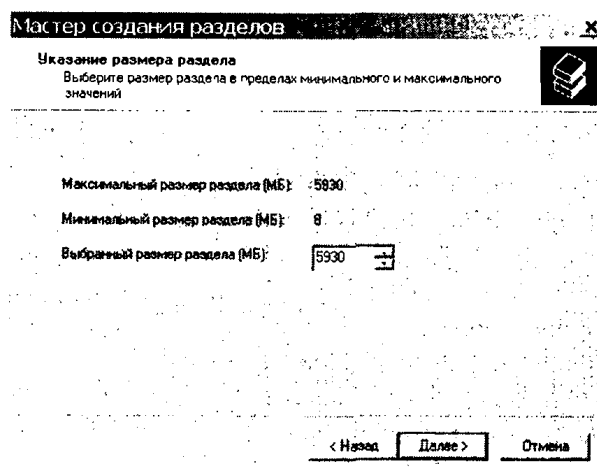


В окне отобразятся жесткие диски и оптические дисководы и существующие на них логические диски.

Выберем свободный раздел диска (отмечен зеленым цветом) и создадим на нем логический диск.

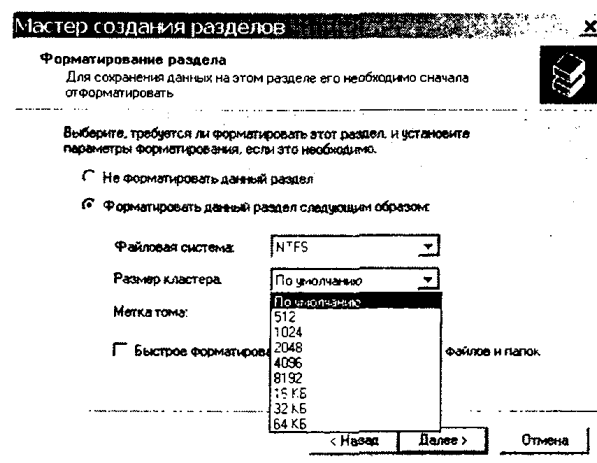
2. В контекстном меню свободного раздела диска выбрать пункт *Создать логический диск...* Будет вызван *Мастер создания разделов*.

В третьем диалоговом окне указать размер раздела. Щелкнуть по кнопке *Далее*.



3. В четвертом диалоговом окне указать букву для обозначения логического диска.

4. В пятом диалоговом окне указать тип файловой системы, размер кластера и метку тома.



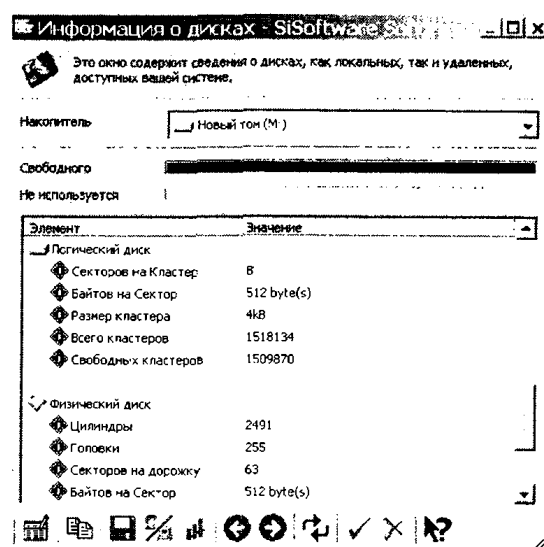
5. В шестом диалоговом окне проверить правильность введенных параметров и щелкнуть по кнопке *Готово*.
В окне *Управление компьютером* отобразится процесс форматирования раздела.

Для получения информации о дисках и логических разделах дисков можно воспользоваться программой тестирования компьютера SiSoft Sandra.



Получение информации о дисках

6. Запустить информационный модуль *Информация о дисках*.
В списке *Накопитель* выбрать созданный логический диск.
В окне появится информация об объеме кластера, количестве кластеров на логическом диске и др.



Практическая работа 1.2

Запись CD- или DVD-диска

Задание. Произвести запись CD- или DVD-диска (например, с помощью приложения *DeerBurner*). При необходимости произвести предварительное стирание CD-RW или DVD±RW диска.

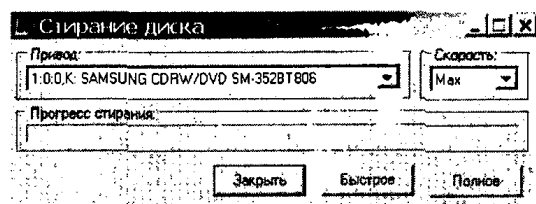
Варианты выполнения работы:

- запись CD-R или CD-RW или DVD±R или DVD±RW диска;
- создание мультисессийного или односессийного диска;
- запись на диск различных наборов файлов.

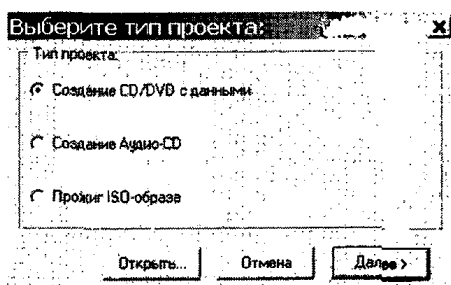


Запись CD- или DVD-диска

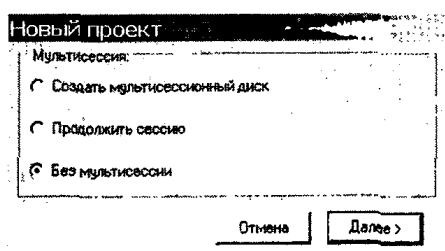
1. Вставить в оптический дисковод перезаписываемый диск. Ввести команду [Запись-Стереть]. В появившемся диалоговом окне *Стирание диска* щелкнуть по кнопке *Быстрое* и наблюдать прогресс стирания.



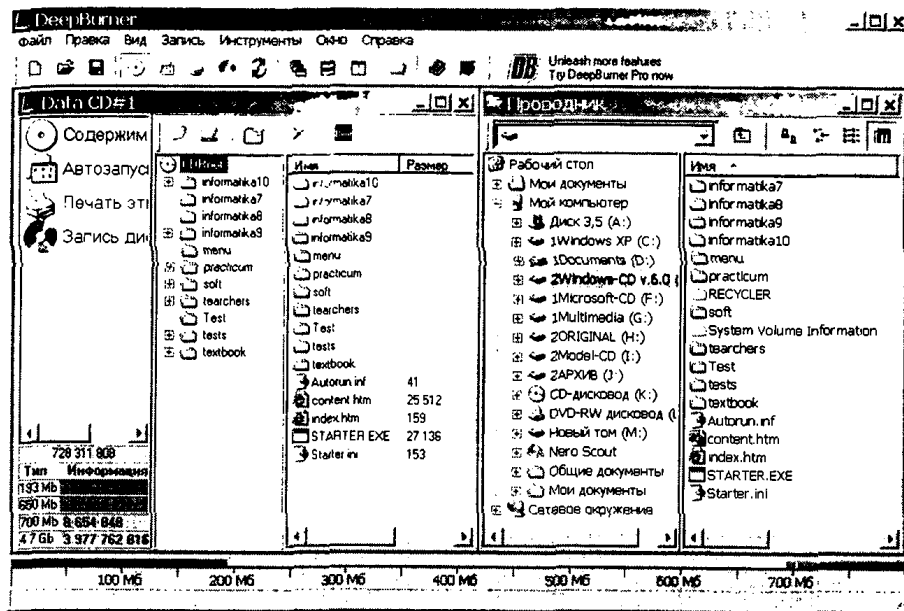
2. Для записи оптического диска выбрать ввести команду [Файл-Новый], в появившемся диалоговом окне выбрать тип проекта и щелкнуть по кнопке *Далее>*.



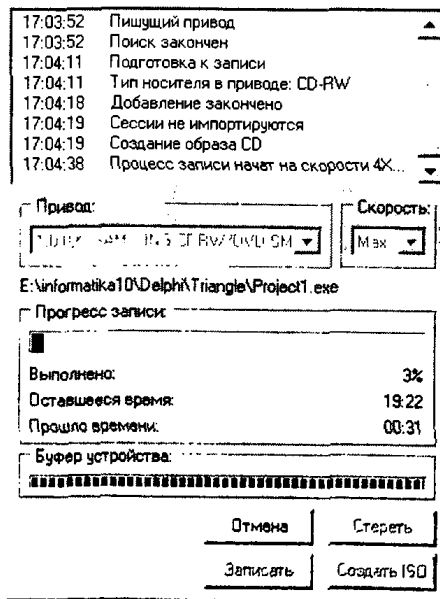
3. В появившемся диалоговом окне *Новый проект* выбрать тип мультисессии.



4. В окне *Проводник* выбрать файлы для записи и скопировать их в папку оптического диска. На нижней шкале будет показан объем, который файлы займут на оптическом диске.



5. Для начала записи ввести команду [Вид-Запись диска].
 В появившемся диалоговом окне будут отображены:
- время этапов создания диска;
 - прогресс записи;
 - процент выполнения;
 - оставшееся время записи;
 - прошедшее время с начала записи.



Практическая работа 1.3

Установка параметров BIOS

Задание. Включить компьютер. С помощью программы BIOS Setup произвести установку новых параметров конфигурации компьютера.

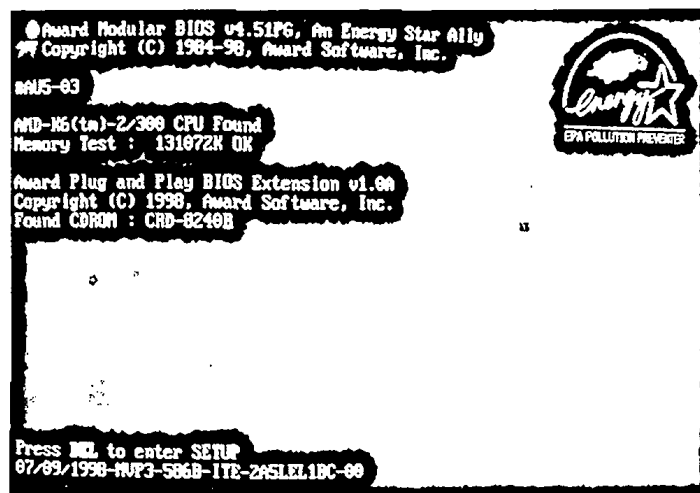
Варианты выполнения работы:

- различные версии BIOS;
- различные значения конфигурационных параметров.



BIOS и загрузка операционной системы

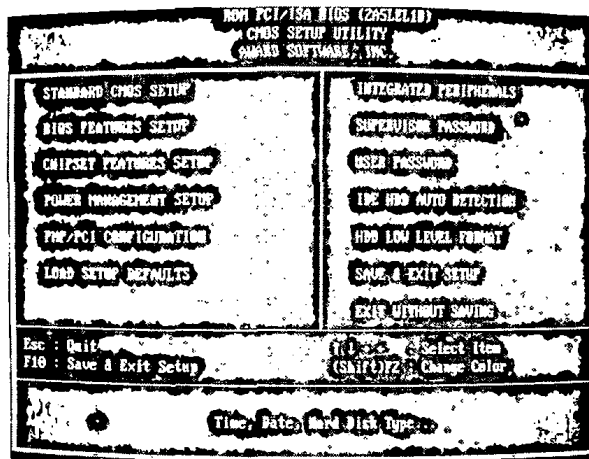
1. Включить компьютер. Наблюдать процесс тестирования компьютера. В случае возникновения звуковых сигналов или сообщений об ошибке устранить неисправность.



В некоторых случаях появление сообщения об ошибке связано с забывчивостью пользователя. Например, в случае появления сообщения «Invalid Boot Diskette» (пер. с англ. «Несистемная дискета») для продолжения загрузки необходимо просто извлечь несистемную дискету из дисковода.

В процессе тестирования BIOS сравнивает получаемые данные о конфигурации компьютера с информацией, хранящейся в CMOS — специальной микросхеме памяти, расположенной на системной плате. Если данные не совпадают, то появляется сообщение: «System Option Not Set» (пер. англ. «Системные параметры не установлены»). В этом случае необходимо с помощью утилиты BIOS Setup установить новые конфигурационные параметры.

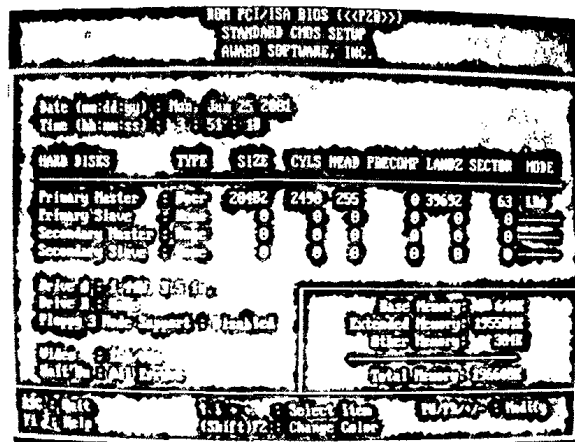
- Для входа в BIOS Setup нажать клавишу, указанную на экране загрузки (чаще всего {Del}), в процессе тестирования. Утилита имеет интерфейс в виде системы иерархического меню, перемещение по которому производится с помощью клавиш управления курсором.



Для правильной установки даты, времени и параметров жестких и гибких дисков необходимо использовать панель *STANDARD CMOS SETUP* (Стандартная установка).

- Установить курсор на пункт меню *STANDARD CMOS SETUP* и нажать клавишу {Enter}.

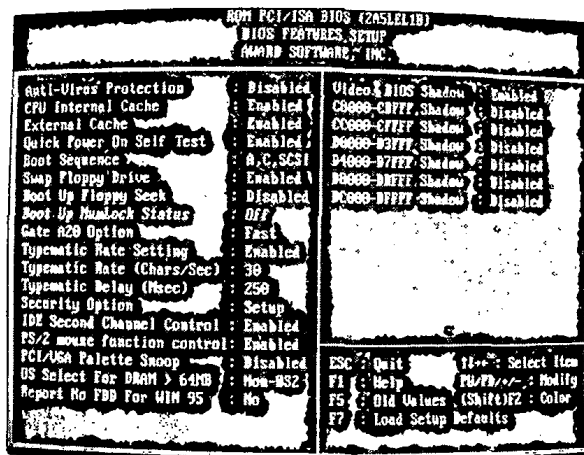
В появившемся окне установить курсор на элемент конфигурационных данных (выделяется цветом) и с помощью клавиш {PageUp} и {PageDown} установить требуемое значение параметра.



С помощью панели *BIOS FEATURES SETUP* (Возможности BIOS) можно установить защиту от заражения вирусом загрузочного сектора системного диска. В этом случае будет заблокировано любое изменение загрузочного сектора, и поэтому при переустановке или установке операционной системы защиту необходимо снять.

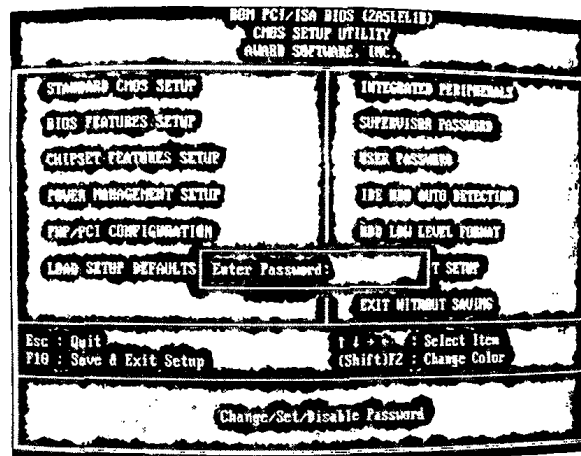
4. Установить курсор на пункт меню *BIOS FEATURES SETUP* и нажать клавишу {Enter}.

На появившейся панели установить курсор на элемент конфигурационных данных *Anti-Virus Protection* и установить значение *Enable*.



Для предохранения данных от несанкционированного доступа можно с помощью BIOS Setup установить пароль для входа в систему или в установку конфигурационных параметров. Обязательно хорошо запомните или запишите пароль, так как в случае его утери вход в систему становится невозможным (сброс пароля возможен только аппаратно, путем отключения микросхемы BIOS от источника питания — батарейки).

5. На панели *BIOS FEATURES SETUP* выбрать пункт *Security Option* и установить значение *System* или *Setup*.
6. Открыть панель *SUPERVISOR PASSWORD*.
На появившейся панели *Enter Password* ввести пароль и нажать клавишу {Enter}.
Повторно ввести пароль для подтверждения его правильности.



Практическая работа 1.4

Защита информации

Задание. Произвести комплексную защиту компьютера от вредоносных программ. Установить и настроить следующие программы защиты информации:

- антивирусные программы: avast! или Antivir Personal Edition;
- программу удаления рекламных и шпионских программ: Ad-Aware;
- программу восстановления системы: CCleaner;
- межсетевой экран: Outpost Firewall.

Тема «Информация. Системы счисления»

Практическая работа 2.1

Кодирование текстовой, графической и звуковой информации

Задание. Определить, сколько поместится на гибком диске, флэш-диске, жестком диске, CD-диске и DVD-диске страниц, каждая из которых содержит:

- 40 строк текста по 50 символов в строке в кодировке Unicode;
- изображение 200 на 100 точек с палитрой 16 777 216 цветов;
- звукозапись среднего качества с частотой дискретизации 24 000 Гц и глубиной кодирования 16 битов.

Варианты выполнения работы:

- различная емкость гибкого диска, флэш-диска, жесткого диска, CD-диска и DVD-диска;
- различное количество строк и символов в строке и кодировка текста;
- различный размер и палитра цветов изображения;
- различная частота дискретизации и глубина кодирования звукозаписи.

Практическая работа 2.2

Системы счисления

Задание. Перевести число, содержащее целую и дробную части, из десятичной системы счисления в двоичную, восьмеричную и шестнадцатеричную системы.

Варианты выполнения работы:

- различные десятичные числа.

Тема «Логика и логические основы компьютера»

Практическая работа 3.1

Равносильность логических выражений


Задание. Докажите, что логические выражения равносильны. Для доказательства постройте таблицы истинности логических выражений в электронных таблицах.

Варианты логических выражений:

- $A \& B \vee A \& B \& (C \vee D) = A \& B$;
- $(A \vee B \vee C) \& \overline{(A \vee B \vee C)} = B \& \overline{A} \& \overline{C}$;
- $(A \vee B) \& (\overline{B} \vee A) \& (\overline{C} \vee B) = A \& (\overline{C} \vee B)$;
- $(A \Rightarrow B) \& (A \vee \overline{B}) = (A \Leftrightarrow B) \& (A \& B) \vee (\overline{A} \& \overline{B})$.



Таблицы истинности логических выражений в электронных таблицах Microsoft Excel и OpenOffice.org Calc

Windows-CD 

Вариант логических выражений:

$$(A \vee B \vee C) \& \overline{(A \vee B \vee C)} = B \& \overline{A} \& \overline{C}$$



\\informatika10\logic\log.*

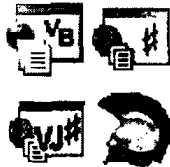
	A	B	C	D	E	F	G	H
1	A	B	C	\bar{B}	$A \vee B \vee C$	$A \vee \bar{B} \vee C$	$A \vee \bar{B} \vee C$	$(A \vee B \vee C) \& (A \vee \bar{B} \vee C)$
2	0	0	0	ИСТИНА	ЛОЖЬ	ИСТИНА	ЛОЖЬ	ЛОЖЬ
3	0	0	1	ИСТИНА	ИСТИНА	ИСТИНА	ЛОЖЬ	ЛОЖЬ
4	0	1	0	ЛОЖЬ	ИСТИНА	ЛОЖЬ	ИСТИНА	ИСТИНА
5	0	1	1	ЛОЖЬ	ИСТИНА	ИСТИНА	ЛОЖЬ	ЛОЖЬ
6	1	0	0	ИСТИНА	ИСТИНА	ИСТИНА	ЛОЖЬ	ЛОЖЬ
7	1	0	1	ИСТИНА	ИСТИНА	ИСТИНА	ЛОЖЬ	ЛОЖЬ
8	1	1	0	ЛОЖЬ	ИСТИНА	ИСТИНА	ЛОЖЬ	ЛОЖЬ
9	1	1	1	ЛОЖЬ	ИСТИНА	ИСТИНА	ЛОЖЬ	ЛОЖЬ
10								
11	A	B	C	\bar{A}	\bar{C}	$B \& \bar{A} \& \bar{C}$		
12	0	0	0	ИСТИНА	ИСТИНА	ЛОЖЬ		
13	0	0	1	ИСТИНА	ЛОЖЬ	ЛОЖЬ		
14	0	1	0	ИСТИНА	ИСТИНА	ИСТИНА		
15	0	1	1	ИСТИНА	ЛОЖЬ	ЛОЖЬ		
16	1	0	0	ЛОЖЬ	ИСТИНА	ЛОЖЬ		
17	1	0	1	ЛОЖЬ	ЛОЖЬ	ЛОЖЬ		
18	1	1	0	ЛОЖЬ	ИСТИНА	ЛОЖЬ		
19	1	1	1	ЛОЖЬ	ЛОЖЬ	ЛОЖЬ		

Тема «Алгоритмизация и основы объектно-ориентированного программирования»

Практическая работа 4.1

Проект «Визуализация сортировки числового массива»

Задание. Заполнить числовой массив, содержащий 10 элементов, числами в диапазоне от 0 до 100. Произвести сортировку массива с отображением шагов сортировки: в списки вывести значения элементов, а в графические поля — линии, пропорциональные значениям элементов. Перед выводом каждого шага сортировки сделать паузу.



Создание графического интерфейса проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

Варианты элементов управления для создания и запуска обработчиков событий (событийных процедур):

- кнопки;
- пункты меню;
- кнопки на панели инструментов.



Создание программного кода проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

Варианты заполнения числового массива:

- случайными числами;
- с клавиатуры;
- из файла.

Варианты сортировки:

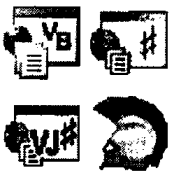
- по возрастанию с поиском максимального (минимального) элемента;
- по убыванию с поиском максимального (минимального) элемента.

Варианты поиска максимального (минимального) элемента:


- с помощью процедуры;
- с помощью функции;
- во вложенном цикле.

Варианты создания области рисования:

- метод `CreateGraphics()`;
- создание растрового изображения `Image1`;
- событие `Paint`.



Запуск проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

Windows-CD 

Вариант проекта:

- кнопки;
- случайными числами;
- по убыванию с поиском максимального элемента; `...\informatika10\...\arrayStringSort3\`
- во вложенном цикле;
- метод `CreateGraphics()`.



TurboDelphi-CD 



...\Projects\arrayStringSort3\

Сортировка числового массива 3

	1	2	3	4	5	6	7	8	9
0	29	79	79	79	79	79	79	79	79
1	19	19	78	78	78	78	78	78	78
2	64	64	64	64	64	64	64	64	64
3	79	29	29	29	62	62	62	62	62
4	7	7	7	7	7	53	53	53	53
5	50	50	50	50	50	50	50	50	50
6	6	6	6	6	6	6	29	29	29
7	78	78	19	19	19	19	19	19	19
8	62	62	62	62	29	29	6	6	7
9	53	53	53	53	53	7	7	7	6

Заполнить массив Сортировать

Практическая работа 4.2

Проект «Тест»

Задание. Считать из текстовых файлов вопросы и правильные ответы в строковые массивы. В цикле:

- задать вопросы;
- получить и запомнить ответ в строковом массиве;
- проверить правильность введенного ответа (сравнить введенный ответ с правильным), если ответ правильный, вывести сообщение «Правильно» на надпись, иначе вывести сообщение «Неправильно» на надпись и прибавить 1 к счетчику ошибок.

В зависимости от количества сделанных ошибок вывести на надпись отметку.

Сохранить введенные ответы в текстовом файле.



Создание графического интерфейса проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

Варианты элементов управления для создания и запуска обработчиков событий (событийных процедур):

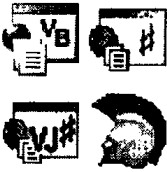
- кнопки;
- пункты меню;
- кнопки на панели инструментов.



Создание программного кода проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

Количество вопросов и соответствующих правильных ответов может изменяться в широких пределах.

Содержание вопросов и соответствующих правильных ответов может быть из различных предметов (информатики, физики, математики, химии, истории, географии и других предметов).



Запуск проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

Вариант проекта:

- пункты меню;
- количество вопросов — 10;
- предмет — информатика.

Windows-CD



...\informatika10...\Test\

TurboDelphi-CD



...\Projects\Test\

Сообщение		Правильно!
Введенный ответ	<HTML>	
Правильный ответ	<HTML>	
Количество ошибок	-	
Отметка	Хорошо	

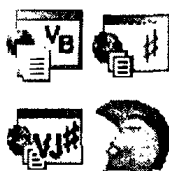
№ 10

С какого тэга начинается Web-страница?

Практическая работа 4.3

Проект «Шифровка и дешифровка»

Задание. Проект должен шифровать и дешифровать текстовые сообщения. Используем алгоритм шифрования, который базируется на использовании ключевой фразы (ключа шифрования). В процессе шифровки секретного текста необходимо заменить каждый символ секретного текста на порядковый номер этого символа в ключевой фразе. В процессе дешифровки последовательность чисел должна преобразовываться обратно в секретный текст с помощью той же ключевой фразы. Ключевая фраза должна содержать все символы, которые будут использоваться в сообщениях, и должна быть известна только шифровальщику и получателю сообщений.



Создание графического интерфейса проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

Варианты состава проекта:

- одна форма, на которой происходит шифровка и дешифровка;
- две формы, на первой форме происходит шифровка, а на второй — дешифровка;
- две формы и независимый программный модуль, на первой форме происходит шифровка, на второй — дешифровка, а в программном модуле содержится программный код поочередного открытия форм.

Варианты для создания и запуска обработчиков событий (событийных процедур):

- кнопки;
- пункты меню;
- кнопки на панели инструментов.



Создание программного кода проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

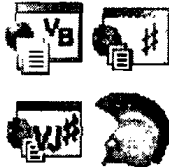
Указания по шифровке. Будем последовательно вырезать в цикле символы из секретного текста, а затем во вложенном цикле будем искать такой же символ в ключевой фразе и запоминать его порядковый номер в целочисленном массиве.

Таким образом, секретный текст будет зашифрован последовательностью чисел, хранящейся в целочисленном массиве. Секретный текст вводится с использованием текстового поля, а шифровка выводится с использованием надписи.

Образец ключевой фразы:

«электронно-вычислительные машины, компьютеры (железо) позволяют находить значения функций, а ещё свойства готовых объектов».

Указания по дешифровке. Шифровка в форме целочисленного массива выводится на надпись. С помощью ключевой фразы необходимо превратить эту числовую последовательность в секретный текст. Считывая по очереди числа шифровки (элементы целочисленного массива), необходимо вырезать из ключевой фразы символ, номер которого по порядку равен значению числа шифровки. Полученные символы последовательно выводятся на надпись, на которой в результате отображается исходный секретный текст.



Запуск проекта на языках Visual Basic .NET, Visual C#, Visual J# и Turbo Delphi

Вариант проекта:

- все варианты состава проекта на различных языках программирования;
- кнопки.

Windows-CD

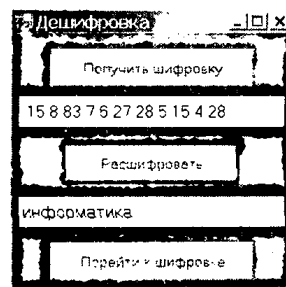
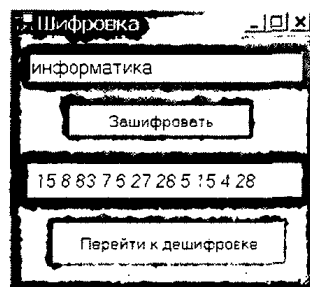


...\\informatika10\...\\Shifr\

TurboDelphi-CD



...\\Projects\\Shifr\




Ответы и решения

Глава 1. Архитектура компьютера и защита информации

Windows-CD 

`\practicum\otvet\`

Глава 2. Информация. Системы счисления

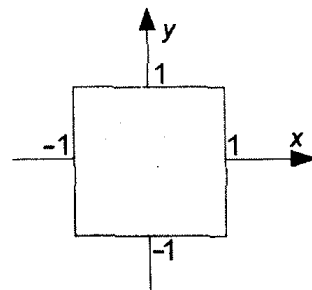
Windows-CD 

`\practicum\otvet\`

Глава 3. Основы логики и логические основы компьютера

- 3.4. а) $x < -3$ или $x > 8$.
б) $-3 \leq x \leq 8$.

3.5.



- 3.6. $A = \langle 2 \times 2 = 4 \rangle$; $B = \langle 3 \times 3 = 9 \rangle$;
 $(A \& B) \vee (\overline{A} \& \overline{B}) = (1 \& 1) \vee (0 \& 0) = 1 \vee 0 = 1$.

- 3.7. Таблица истинности логического выражения $\overline{\overline{A \vee B}}$ совпадает с таблицей истинности логического выражения $A \& B$.

A	B	\bar{A}	\bar{B}	$\bar{A} \& \bar{B}$	$\overline{\bar{A} \vee \bar{B}}$
0	0	1	1	0	0
0	1	1	0	0	0
1	0	0	1	0	0
1	1	0	0	1	1

3.9. Первый закон де Моргана:

A	B	\bar{A}	\bar{B}	$A \vee B$	$\overline{\bar{A} \vee \bar{B}}$	$\bar{A} \& \bar{B}$
0	0	1	1	0	1	1
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	1	0	0	1	0	0

Второй закон де Моргана:

A	B	\bar{A}	\bar{B}	$A \& B$	$\overline{\bar{A} \& \bar{B}}$	$A \vee \bar{B}$
0	0	1	1	0	1	1
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	0	0

3.10. $(A \vee B) \& (A \vee \bar{B}) = A \vee (B \& \bar{B}) = A \vee 0 = A$

3.11. $\overline{\bar{X} \& \bar{B}} \& \overline{X \& \bar{B}} = A$

$$(\bar{X} \vee \bar{B}) \& (\bar{X} \vee B) = A$$

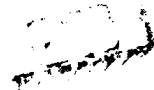



$$\bar{X} \vee (\bar{B} \& B) = A$$

$$\bar{X} \vee 0 = A$$

$$X = \bar{A}$$

Плакаты

Таблица 1. Процессоры

Тип	Внешний вид	Технология	Кол-во элементов	Год выпуска	Частота (МГц)	Шина данных	Шина адреса	Адресуемая память
4004		10 мк	2300	1971	0,1	4	12	4 Кб
8086		3 мк	29000	1978	5-10	16	20	1 Мб
80286		1,5 мк	120000	1982	4-25	16	24	16 Мб
80386		1,5 мк	275000	1985	16-40	32	32	4 Гб







Тип	Внешний вид	Технология	Кол-во элементов	Год выпуска	Частота (МГц)	Шина данных	Шина адреса	Адресуемая память
80486		1 мк	1180000	1989	25–133	32	32	4 Гб
Pentium		0,8 мк	3100000	1993	60–266	64	32	4 Гб
Pentium II		0,35 мк	7500000	1997	266–333	64	36	64 Гб
Pentium III		0,18 мк	24000000	1999	400–1000	64	36	64 Гб
Pentium 4		0,13 мк	42000000	2000	1000–3200	64	36	64 Гб
Pentium Extreme Edition		0,065 мк = 65 нм	376000000	2006	3300–3700	64	64	2 ²⁴ Тб

Таблица 2. Элементы управления для ввода и вывода данных

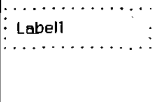
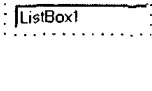
Элемент управления	Внешний вид	Назначение	Свойства	Методы	Событие по умолчанию
Надпись (Label)		Отображает данные, которые нельзя изменить в процессе выполнения проекта	Name — имя Text — надпись Location (X; Y) — координаты верхнего левого угла элемента управления на форме	Show — показывает элемент управления Hide — скрывает элемент управления	Label_Click — происходит при щелчке по надписи
Текстовое поле (TextBox)		Используется для ввода или отображения данных	Size (Width; Height) — размер (ширина; высота) BackColor — цвет фона ForeColor — цвет текста	Focus — делает активным выбранный элемент управления Scale — масштабирует форму или элемент управления	TextBox1_TextChanged — происходит при изменении текста в текстовом поле
Улучшенное текстовое поле (RichTextBox)		Используется для ввода и отображения форматированного текста	Font (Name; Size; Bold и др.) — шрифт (название шрифта, размер и начертание)	Refresh — перерисовывает форму или элемент управления	RichTextBox1_TextChanged — происходит при изменении текста в текстовом поле
Графическое поле (PictureBox)		Предназначен для отображения графических объектов	Image — путь к графическому файлу, который отображается в графическом поле	Draw — рисует графические примитивы	PictureBox1_Click — происходит при щелчке в графическом поле
Список (ListBox)		Отображает список, в котором можно выбрать один или несколько элементов	Items — позволяет задать или получить значения элементов списка Sorted — позволяет сортировать элементы списка	Add — добавляет элемент в список Remove — удаляет элемент из списка	ListBox1_SelectedIndexChanged — происходит при выборе элемента списка

Таблица 3. Элементы управления для организации интерактивного диалога

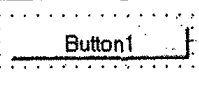
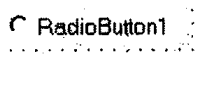
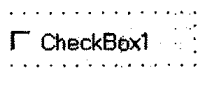
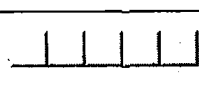
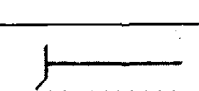
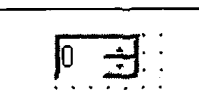
Элемент управления	Внешний вид	Назначение	Свойства	Событие по умолчанию
Кнопка (Button)		Щелчок по кнопке вызывает событийную процедуру	Смотри в таблице 2 свойства надписи	Button1_Click — происходит при щелчке по кнопке
Переключатель (RadioButton)		Представляет собой набор из двух или более взаимоисключающих вариантов	Checked — указывает, выбран или нет данный переключатель или флажок AutoCheck — вызывает автоматическое изменение состояния переключателя или флажка при его выборе	RadioButton1_CheckedChanged — происходит при выборе переключателя
Флажок (CheckBox)		Используется для представления выбора «Да/Нет» или «Истина/Ложь»		CheckBox1_CheckedChanged — происходит при выборе флажка
Панель инструментов (ToolBar)		Используется для создания панели инструментов проекта	Buttons — набор кнопок	ToolBar1_ButtonClick — щелчок по кнопке на панели инструментов
Ползунок (TrackBar)		Используется для визуальной настройки числовых параметров	Value — содержит число, соответствующее положению ползунка, счетчика или движка	TrackBar1_Scroll — происходит при установке ползунка
Счетчик (NumericUpDown)		Выводит и задает числовое значение в списке	Minimum — минимальное значение ползунка или счетчика Maximum — максимальное значение ползунка или счетчика	NumericUpDown1_ValueChanged — происходит при установке значения счетчика

Таблица 4. Элементы управления, не отображаемые на форме в процессе выполнения проекта

Элемент управления	Внешний вид	Назначение	Свойства	Методы
Шрифт (FontDialog)	 FontDialog1	Позволяет задать параметры шрифта	Font — параметры шрифта, установленные по умолчанию	ShowDialog — выводит диалоговое окно <i>Шрифт</i>
Цвет (ColorDialog)	 ColorDialog1	Позволяет задать цвет	Color — цвет, установленный по умолчанию	ShowDialog ~ выводит диалоговое окно <i>Цвет</i>
Открытие файла (OpenFileDialog)	 OpenFileDialog1	Позволяет в иерархической файловой системе выбрать файл	InitialDirectory — папка, открываемая в диалоговом окне	ShowDialog — выводит диалоговое окно <i>Открыть</i>
Сохранение файла (SaveFileDialog)	 SaveFileDialog1	Позволяет в иерархической файловой системе сохранить файл	InitialDirectory — папка, открываемая в диалоговом окне	ShowDialog — выводит диалоговое окно <i>Сохранить как</i>
Меню (MainMenu)	 MainMenu1	Позволяет создать иерархическое меню проекта	Text — позволяет вводить названия пунктов иерархического меню	
Список рисунков (ImageList)	 ImageList1	Используется для хранения рисунков, которые могут отображаться другими элементами управления	Images — позволяет создать коллекцию рисунков	
Таймер (Timer)	 Timer1	Вызывает событие через определенные интервалы времени	Interval -- интервал времени между событиями в миллисекундах	Start — запускает таймер

Таблица 5. Типы переменных

Visual Basic .NET	Visual C#	Visual J#	Turbo Delphi	Занимаемая память	Диапазон значений
Целочисленные типы					
Byte	byte	ubyte	Byte	1 байт	от 0 до 255
Short	short	short	Smallint	2 байта	от -32 768 до 32 767
Integer	int	int	Integer	4 байта	от -2 147 483 648 до 2 147 483 647
Long	long	long	Int64	8 байтов	от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807
Типы с плавающей запятой					
Single	float	float	Single	4 байта	от $-1.5 \cdot 10^{-45}$ до $3.4 \cdot 10^{38}$, 7-8 значащих цифр
Double	double	double	Double	8 байтов	от $-5.0 \cdot 10^{-324}$ до $1.7 \cdot 10^{308}$, 15-16 значащих цифр
Decimal	decimal	Нет	Нет	16 байтов	от $\pm 1.0 \times 10^{-28}$ до $\pm 7.9 \times 10^{28}$, 28-29 значащих цифр
Строковые типы					
Char	char	char	Char	2 байта	знак Unicode
String	string	Нет	String	2 байта × количество символов	от 0 до приблизительно 2 миллиардов знаков Unicode
Логический тип					
Boolean	bool	boolean	Boolean	2 байта	True или False

Таблица 6. Встроенные функции (методы)

Функция (метод)	Visual Basic .NET	Visual C#	Visual J#	Turbo Delphi
Математические функции (методы)				
Квадратный корень	Math.Sqrt()	Math.Sqrt()	System.Math.Sqrt()	Sqrt()
Степень	Math.Pow()	Math.Pow()	System.Math.Pow()	Power()
Целая часть числа	Math.Int()	Math.Int()	System.Math.Int()	Int()
Округление до целого	Math.Round()	Math.Round()	System.Math.Round()	Round()
Синус	Math.Sin()	Math.Sin()	System.Math.Sin()	Sin()
Косинус	Math.Cos()	Math.Cos()	System.Math.Cos()	Cos()
Случайное число	Rnd()	Microsoft.VisualBasic.Rnd()	Microsoft.VisualBasic.Rnd()	Random()
Функции (методы) преобразования типов данных				
В число с плавающей запятой	System.Convert.ToDouble() System.Convert.ToSingle()	System.Convert.ToDouble() System.Convert.ToSingle()	System.Convert.ToDouble() System.Convert.ToSingle()	StrToFloat()
В целое число	System.Convert.ToByte() System.Convert.ToInt32()	System.Convert.ToByte() System.Convert.ToInt32()	System.Convert.ToByte() System.Convert.ToInt32()	StrToInt() StrToInt64()
В строку	System.Convert.ToString()	System.Convert.ToString()	System.Convert.ToString()	IntToStr() FloatToStr()
Функции (методы) ввода и вывода				
Ввод	InputBox()	Microsoft.VisualBasic.InputBox()	Microsoft.VisualBasic.InputBox()	InputBox()
Вывод	MsgBox()	Microsoft.VisualBasic.MsgBox()	Microsoft.VisualBasic.MsgBox()	MessageBox()

Таблица 7. Кодирование алгоритмических структур «ветвление» и «выбор»
Кодирование алгоритмической структуры «ветвление»

Visual Basic .NET	Visual C#	Visual J#	Turbo Delphi
If Условие Then Серия 1 Else Серия 2 End If	if (Условие) { Серия 1; } [else { Серия 2; }]	if (Условие) { Серия 1; } [else { Серия 2; }]	If Условие Then begin Серия 1 end [Else begin Серия 2 end];

Кодирование алгоритмической структуры «выбор»

Visual Basic .NET	Visual C#	Visual J#	Turbo Delphi
Select Case Выражение Case Условие1 Серия 1 Case Условие2 Серия 2 [Case Else Серия] End Select	switch (Выражение) { case список1_констант: Серия 1; break; case список2_констант: Серия 2; break; [default: Серия; break; }]	switch (Выражение) { case список1_констант: Серия 1; break; case список2_констант: Серия 2; break; [default: Серия; break; }]	Case Выражение Of список1_констант: begin Серия 1 end; список2_констант: begin Серия 2 end; [Else begin Серия end;] end;

**Таблица 8. Кодирование алгоритмической структуры «цикл»
Кодирование алгоритмической структуры «цикл со счетчиком»**

Visual Basic .NET	Visual C#	Visual J#	Turbo Delphi
For Счетчик=НачЗнач To КонЗнач [Step шаг] Тело цикла Next [Счетчик]	for (int Счетчик = 1; Счетчик < НачЗнач; Счетчик++) { Тело цикла; }	for (int Счетчик = 1; Счетчик < НачЗнач; Счетчик++) { Тело цикла; }	For Счетчик:=НачЗнач to КонЗнач Do begin Тело цикла end ;

Кодирование алгоритмической структуры «цикл с предусловием»

Visual Basic .NET	Visual C#	Visual J#	Turbo Delphi
Do While Условие Тело цикла Loop Do Until Условие Тело цикла Loop	while (Условие) { Тело цикла; }	while (Условие) { Тело цикла; }	while Условие do begin Тело цикла; end ;

Кодирование алгоритмической структуры «цикл с постусловием»

Visual Basic .NET	Visual C#	Visual J#	Turbo Delphi
Do Тело цикла Loop While Условие Do Тело цикла Loop Until Условие	do { Тело цикла; } while (Условие);	do { Тело цикла; } while (Условие);	Repeat Тело цикла; Until Условие;

Словарь компьютерных терминов

BIOS	Программы начальной загрузки, находящиеся в микросхеме памяти (англ. «Basic Input/Output System» — базовая система ввода/вывода)
Blu-Ray	Оптический диск повышенной информационной емкости (англ. «Blue-Ray» — голубой луч)
CD	Компакт-диск (англ. «Compact Disk»)
CDFS	Файловая система для работы с оптическими дисками, предназначенная для использования под управлением различных операционных систем (англ. «Compact Disk File System» — файловая система для компакт-дисков)
DDR	Тип модулей (англ. «Double Data Rate» — удвоенная скорость передачи данных). Модули памяти DDR имеют 184 разъема
DDR2	Тип модулей оперативной памяти, которые обладают меньшим энергопотреблением, чем модули DDR, и поэтому могут иметь большую частоту. Модули памяти DDR2 имеют 240 разъемов
DVD	Цифровой универсальный диск (англ. «Digital Versatile Disk»)
ext3	Журналируемая файловая система для операционной системы Linux (англ. «ext3» — 3-я расширенная файловая система)
FAT	Таблица размещения файлов (англ. «File Allocation Table»)
HD DVD	Цифровой универсальный диск повышенной информационной емкости (англ. «High Definition DVD»)
HFS	Журналируемая файловая система для операционной системы Mac OS (англ. «Hierarchical File System» — иерархическая файловая система)

NTFS	Журналируемая файловая система для операционной системы Windows (англ. «New Technology File System» – файловая система новой технологии)
POST	Система самотестирования компьютера при включении (англ. «Power On Self Test»)
ReiserFS	Журналируемая файловая система для операционной системы Linux (англ. «ReiserFS» – файловая система Райзера).
SFX	<i>Самораспаковывающийся архив</i> (англ. «Self-eXtracting»)
UDF	Файловая система для работы с файлами на оптических дисках (англ. «Universal Disk Format» – универсальный дисковый формат)
Кластер	Логическая единица хранения данных в таблице размещения файлов, объединяющая группу секторов (англ. «cluster»)
Мультисессия	При записи CD- или DVD-диска можно записывать файлы в несколько сессий (приемов)
Полевой транзистор	Транзистор, в котором ток изменяется в результате действия перпендикулярного току электрического поля, создаваемого входным сигналом
Синергетика	Наука о самоорганизующихся системах в неживой и живой природе, обменивающихся веществом, энергией и информацией с окружающей средой
Хакер	Компьютерный злоумышленник, который организует сетевые атаки на Интернет-серверы, а также проникает на них с неблагоприятными целями

Учебное издание

Угринович Николай Дмитриевич

ИНФОРМАТИКА и ИКТ
Учебник для 10 класса
Профильный уровень

Ведущий редактор *О. Полежаева*
Художник *С. Инфантэ*
Художественный редактор *О. Лапко*
Корректор *Е. Клитина*
Компьютерная верстка *Л. Катуркина*

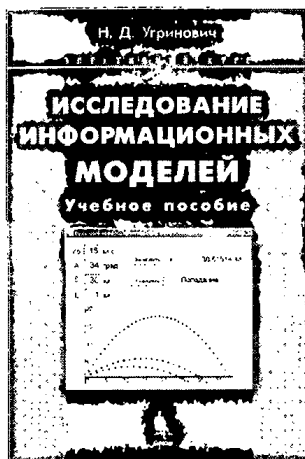
Подписано в печать 22.01.08. Формат 60×90 $\frac{1}{16}$
Бумага офсетная. Усл. печ. л. 24,5
Тираж 25 000 экз. Заказ 400

Издательство «БИНОМ. Лаборатория знаний»
Адрес для переписки: 125167, Москва, проезд Аэропорта, 3
Телефон: (499) 157-5272. E-mail: Lbz@aha.ru
<http://www.Lbz.ru>

При участии ООО «ЭМПРЕЗА»

Отпечатано с готовых файлов заказчика в ОАО «ИПК
«Ульяновский Дом печати». 432980, г. Ульяновск, ул. Гончарова, 14

ИМЕЕТСЯ В ПРОДАЖЕ



Угринович Н. Д. Исследование информационных моделей. Элективный курс : учебное пособие / Н. Д. Угринович. – 2-е изд., испр. – 2006. – 200 с. : ил.

Данное учебное пособие является частью УМК наряду с компьютерным практикумом на CD-ROM. Курс учит создавать и исследовать информационные модели из различных предметных областей с использованием систем объектно-ориентированного программирования и электронных таблиц.

На CD-ROM представлены: практикум по построению и исследованию моделей в форме проектов на языках Visual Basic и Delphi, а также компьютерных моделей в электронных таблицах; дистрибутивы систем объектно-ориентированного программирования Visual Basic и Delphi, а также электронных таблиц StarOffice Calc и OpenOffice Calc.

Для учащихся старших классов информационно-технологического, физико-математического и естественно-научного профилей.



Угринович Н. Д. Исследование информационных моделей. Элективный курс : компьютерный практикум на CD-ROM / Н. Д. Угринович. – 2-е изд., испр. – 2006. – 200 с. : ил.

CD-ROM включает интерактивный практикум, содержащий указания по выполнению практических заданий и ответы на них, т. е. готовые проекты на языках Visual Basic и Delphi. На CD-ROM размещено программное обеспечение, необходимое для реализации компьютерного практикума, а именно свободно распространяемые версии систем программирования Visual Basic и Delphi, а также файлы электронных таблиц.



ИЗДАТЕЛЬСТВО
«БИНОМ
Лаборатория знаний»

125167, Москва, проезд Аэропорта, д. 3
Телефон: (499) 157-5272
e-mail: Lbz@aha.ru. <http://www.Lbz.ru>
Оптовые поставки:
(495) 174-7616. 171-1954, 170-6674

Э Л Е К Т И В Н Ы Е К У Р С Ы

■ ИНФОРМАТИКА

ИМЕЕТСЯ В ПРОДАЖЕ



Монахов М. Ю. Учимся проектировать на компьютере. Элективный курс : практикум / М. Ю. Монахов, С. Л. Солодов, Г. Е. Монахова. — 2-е изд., испр. — 2006. — 172 с. : ил.

Практикум обеспечивает преподавание курса компьютерного проектирования в старших классах.

Практикум позволяет освоить основы современных компьютерных технологий проектирования и дизайна. Рассмотрены компьютерные системы проектирования AutoCAD и 3D Studio MAX. Главы практикума представляют собой законченные учебные модули, каждый из которых включает в себя краткую теорию по теме, типовые практические работы, вопросы для самоконтроля и проверочные задания.

Для учащихся старших классов физико-математического, естественно-научного, технологического профилей и универсального обучения.



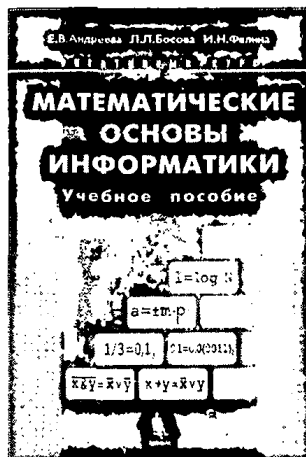
ИЗДАТЕЛЬСТВО
«БИНОМ
Лаборатория знаний»

125167, Москва, проезд Аэропорта, д. 3
Телефон: (499) 157-5272
e-mail: lbz@aha.ru; <http://www.lbz.ru>
Оптовые поставки:
(495) 174-7616; 171-1954; 170-6674

Э Л Е К Т И В Н Ы Е К У Р С Ы

■ МАТЕМАТИКА

ИМЕЕТСЯ В ПРОДАЖЕ



Андреева Е. В. *Математические основы информатики. Элективный курс : учебное пособие / Е. В. Андреева, Л. Л. Босова, И. Н. Фалина. — 2-е изд., испр. — 2007. — 328 с. : ил.*

Учебное пособие входит в УМК для старших классов наряду с методическим пособием и хрестоматией.

Материал раскрывает взаимосвязь математики и информатики, показывает, как развитие одной из этих научных областей стимулировало развитие другой. Дается углубленное представление о математическом аппарате, используемом в информатике, демонстрируются, как результаты, полученные в математике, послужили источником новых идей и результатов в теории алгоритмов, программировании и в других разделах информатики.

Для учащихся старших классов информационно-технологического, физико-математического и естественно-научного профилей, желающих расширить свои теоретические представления о математике в информатике и информатике в математике.



ИЗДАТЕЛЬСТВО
«БИНОМ
Лаборатория знаний»

125167, Москва, проезд Аэропорта, д. 3
Телефон: (499) 157-5272
e-mail: Lbz@aha.ru, <http://www.Lbz.ru>
Оптовые поставки:
(495) 174-7616, 171-1954, 170-6674

ЛИТЕРАТУРА ДЛ Я ШКОЛ

■ ИНФОРМАТИКА

ИМЕЕТС Я В ПРОДАЖЕ



Залогова Л. А. Компьютерная графика. Элективный курс : учебное пособие / Л. А. Залогова. – 2-е изд. – 2007. – 212 с. : 16 с. ил. : ил.

Учебное пособие входит в УМК для старших классов наряду с практикумом.

В учебном пособии рассмотрены вопросы представления графических изображений, описания цветовых оттенков на мониторе и принтере, форматы графических файлов, описаны основные возможности редакторов векторной графики CorelDRAW и растровой графики Adobe Photoshop.

Для учащихся старших классов физико-математического, естественно-научного, социально-гуманитарного и технологического профилей.



ИЗДАТЕЛЬСТВО
«БИНОМ
Лаборатория знаний»

125167, Москва, проезд Аэропорта, д. 3
Телефон: (499) 157-5272
e-mail: Lbz@aha.ru, <http://www.lbz.ru>
Оптовые поставки:
(495) 174-7616, 171-1954, 170-6674



Николай Дмитриевич Угринович — ученый и методист, заведующий лабораторией информатики Московского института открытого образования, кандидат педагогических наук, автор учебного и программно-методического комплекса по курсу «Информатика и ИКТ», который включает учебники для 7–11 классов, элективный курс, методическое пособие для учителей, электронные материалы.

Данный учебник для 10 класса предназначен для обучения информатике и ИКТ на профильном уровне.

Вы получите прочные знания по:

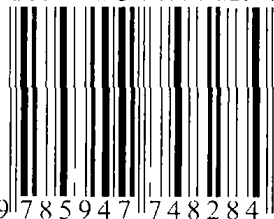
- архитектуре компьютера и методам защиты информации
- системам счисления и понятию информации
- логике и логическим основам компьютера

Вы научитесь программировать на четырех языках объектно-ориентированного программирования:

- Visual Basic
- Delphi
- Visual C#
- Visual J#

Учебник рекомендуется для изучения в 10 классах информационно-технологического и физико-математического профилей.

ISBN 978-5-94774-828-4



9 785947 748284