Н. Д. Угринович

ИНФОРМАТИКА И ИКТ

Учебник для 9 класса

2-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
2009

УДК 004.9
ББК 32.97
У27

Угринович Н. Д.

У27 Информатика и ИКТ : учебник для 9 класса / Н. Д. Угринович. — 2-е изд., испр. — М. : БИНОМ. Лаборатория знаний, 2009. — 295 с. : илл.

ISBN 978-5-9963-0065-5

Учебник предназначен для продолжения изучения курса «Информатика и ИКТ» в 9-х общеобразовательных классах, а также в классах предпрофильной подготовки по физико-математическому и информационно-технологическому профилям. Учебник полностью соответствует образовательному стандарту по информатике и ИКТ. Большое внимание в учебнике уделяется формированию у учащихся алгоритмического и системного мышления, а также практических умений и навыков в области информационных технологий. Учебник мультисистемный, так как практические работы компьютерного практикума могут выполняться в операционных системах Windows и Linux.

УДК 004.9
ББК 32.97

ISBN 978-5-9963-0065-5

© Угринович Н. Д., 2009
© БИНОМ. Лаборатория знаний, 2009

Оглавление

Рекомендации по использованию учебника **8**

Глава 1. Кодирование и обработка графической и мультимедийной информации **10**

1.1. Кодирование графической информации 10
1.1.1. Пространственная дискретизация 10
1.1.2. Растровые изображения на экране монитора 14
1.1.3. Палитры цветов в системах цветопередачи RGB, CMYK и HSB 15
1.2. Растровая и векторная графика 21
1.2.1. Растровая графика 21
1.2.2. Векторная графика 24
1.3. Интерфейс и основные возможности графических редакторов 28
1.3.1. Рисование графических примитивов в растровых и векторных графических редакторах 28
1.3.2. Инструменты рисования растровых графических редакторов 31
1.3.3. Работа с объектами в векторных графических редакторах 32
1.3.4. Редактирование изображений и рисунков в растровых и векторных графических редакторах 35
1.4. Растровая и векторная анимация 37
1.5. Кодирование и обработка звуковой информации 40
1.6. Цифровое фото и видео 45

Глава 2. Кодирование и обработка текстовой информации **49**

2.1. Кодирование текстовой информации 49
2.2. Создание документов в текстовых редакторах 52

2.3. Ввод и редактирование документа 54
2.4. Сохранение и печать документов 59
2.5. Форматирование документа 61
2.5.1. Форматирование символов 61
2.5.2. Форматирование абзацев 63
2.5.3. Нумерованные и маркированные списки 66
2.6. Таблицы 67
2.7. Компьютерные словари и системы машинного перевода текстов 70
2.8. Системы оптического распознавания документов 71

Глава 3. Кодирование и обработка числовой информации 75

3.1. Кодирование числовой информации 75
3.1.1. Представление числовой информации с помощью систем счисления 75
3.1.2. Арифметические операции в позиционных системах счисления 80
3.1.3. *Двоичное кодирование чисел в компьютере 82
3.2. Электронные таблицы 84
3.2.1. Основные параметры электронных таблиц 84
3.2.2. Основные типы и форматы данных 87
3.2.3. Относительные, абсолютные и смешанные ссылки . . 89
3.2.4. Встроенные функции 91
3.3. Построение диаграмм и графиков в электронных таблицах 93
3.4. Базы данных в электронных таблицах 97
3.4.1. Представление базы данных в виде таблицы и формы 97
3.4.2. Сортировка и поиск данных в электронных таблицах 100

Глава 4. Основы алгоритмизации и объектно-ориентированного программирования 105

4.1. Алгоритм и его формальное исполнение 105
4.1.1. Свойства алгоритма и его исполнители 105
4.1.2. Блок-схемы алгоритмов 108
4.1.3. Выполнение алгоритмов компьютером 109
4.2. Кодирование основных типов алгоритмических структур на языках объектно-ориентированного и процедурного программирования 113

4.2.1. Линейный алгоритм 113
4.2.2. Алгоритмическая структура «ветвление» 114
4.2.3. Алгоритмическая структура «выбор» 115
4.2.4. Алгоритмическая структура «цикл» 117
4.3. Переменные: тип, имя, значение 119
4.4. Арифметические, строковые и логические выражения 123
4.5. Функции в языках объектно-ориентированного и алгоритмического программирования 124
4.6. Основы объектно-ориентированного визуального программирования 128
4.7. *Графические возможности объектно-ориентированного языка программирования Visual Basic 2005 133

Глава 5. Моделирование и формализация 138

5.1. Окружающий мир как иерархическая система 138
5.2. Моделирование, формализация, визуализация 142
5.2.1. Моделирование как метод познания 142
5.2.2. Материальные и информационные модели 145
5.2.3. Формализация и визуализация информационных моделей 148
5.3. Основные этапы разработки и исследования моделей на компьютере 152
5.4. Построение и исследование физических моделей 154
5.5. Приближенное решение уравнений 157
5.6. Экспертные системы распознавания химических веществ 157
5.7. Информационные модели управления объектами 161

Глава 6. Информатизация общества 164

6.1. Информационное общество 164
6.2. Информационная культура 169
6.3. Перспективы развития информационных и коммуникационных технологий (ИКТ) 171

Компьютерный практикум 174

Практические работы к главе 1 «Кодирование и обработка графической и мультимедийной информации» 174
Практическая работа 1.1. Кодирование графической информации 175

Практическая работа 1.2. Редактирование изображений в растровом графическом редакторе 177
Практическая работа 1.3. Создание рисунков в векторном графическом редакторе 179
Практическая работа 1.4. Анимация 183
Практическая работа 1.5. Кодирование и обработка звуковой информации 188
Практическая работа 1.6. Захват цифрового фото и создание слайд-шоу 191
Практическая работа 1.7. Захват и редактирование цифрового видео с использованием системы нелинейного видеомонтажа 193

Практические работы к главе 2 «Кодирование и обработка текстовой информации» 196
Практическая работа 2.1. Кодирование текстовой информации 196
Практическая работа 2.2. Вставка в документ формул 199
Практическая работа 2.3. Форматирование символов и абзацев 201
Практическая работа 2.4. Создание и форматирование списков 204
Практическая работа 2.5. Вставка в документ таблицы, ее форматирование и заполнение данными 207
Практическая работа 2.6. Перевод текста с помощью компьютерного словаря 211
Практическая работа 2.7. Сканирование и распознавание «бумажного» текстового документа 212

Практические работы к главе 3 «Кодирование и обработка числовой информации» 214
Практическая работа 3.1. Перевод чисел из одной системы счисления в другую с помощью калькулятора ... 214
Практическая работа 3.2. Относительные, абсолютные и смешанные ссылки в электронных таблицах 216
Практическая работа 3.3. Создание таблиц значений функций в электронных таблицах 218
Практическая работа 3.4. Построение диаграмм различных типов 220
Практическая работа 3.5. Сортировка и поиск данных в электронных таблицах 228

Практические работы к главе 4 «Алгоритмизация и основы объектно-ориентированного программирования» .. 233
Практическая работа 4.1. Знакомство с системами объектно-ориентированного и алгоритмического программирования 233
Практическая работа 4.2. Проект «Переменные» 239
Практическая работа 4.3. Проект «Калькулятор» 242
Практическая работа 4.4. Проект «Строковый калькулятор» 246
Практическая работа 4.5. Проект «Даты и время» 249

Практическая работа 4.6. Проект «Сравнение кодов символов» 252
Практическая работа 4.7. Проект «Отметка» 255
Практическая работа 4.8. Проект «Коды символов» 258
Практическая работа 4.9. Проект «Слово-перевертыш» ... 261
*Практическая работа 4.10. Проект «Графический редактор» 263
*Практическая работа 4.11. Проект «Системы координат» 267
*Практическая работа 4.12. Проект «Анимация» 270

Практические работы к главе 5 «Моделирование и формализация» 272
*Практическая работа 5.1. Проект «Бросание мячика в площадку» 273
Практическая работа 5.2. Проект «Графическое решение уравнения» 279
Практическая работа 5.3. Проект «Распознавание удобрений» 283
Практическая работа 5.4. Проект «Модели систем управления» 286

Ответы и решения к заданиям для самостоятельного выполнения 292

Рекомендации по использованию учебника

1. Учебник «Информатика и ИКТ-9» входит в состав учебно-программного комплекса, который обеспечивает изучение курса «Информатика и ИКТ» в соответствии с образовательным стандартом. В состав комплекса входят:
   - учебники для основной школы: «Информатика и ИКТ-8» и «Информатика и ИКТ-9»;
   - учебники по профильному курсу в старшей школе на базовом уровне: «Информатика и ИКТ-10. Базовый уровень» и «Информатика и ИКТ-11. Базовый уровень»;
   - учебники по профильному курсу в старшей школе 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», включающее диски:
     - Windows-CD, содержащий свободно распространяемую программную поддержку курса, готовые компьютерные проекты, рассмотренные в учебниках, тесты и методические материалы для учителей;
     - Linux-DVD (выпускается по лицензии компании AltLinux), содержащий операционную систему Linux и программную поддержку курса;
     - VisualStudio-CD (выпускается по лицензии корпорации Microsoft), содержащий системы объектно-ориентированного программирования Visual Basic 2005, Visual C# и Visual J#;
     - TurboDelphi-CD (выпускается по лицензии компании Borland), содержащий систему объектно-ориентированного программирования Turbo Delphi.
2. Компьютерный практикум может проводиться в операционных системах Windows и Linux. Для каждой главы указано необходимое для выполнения работ компьютерного практикума программное обеспечение и его источник.

3. В случае выделения на предмет «Информатика и ИКТ» количества часов, не большего, чем указано в Федеральном базисном учебном плане, рекомендуется выполнять практические задания компьютерного практикума в одной операционной системе (Windows или Linux).
4. Начало каждой работы компьютерного практикума обозначается значком операционной системы и приложений, для которых приведена подробная пошаговая инструкция выполнения работы.
5. В учебнике используются ссылки на внешние источники информации (учебники, CD-диски и Интернет), а также на параграфы и пункты самого учебника:

6. Параграфы, пункты и задания повышенного предпрофильного уровня обозначены звездочками *.
7. В тексте пособия приняты следующие шрифтовые выделения:
   - шрифтом Arial выделены имена программ, файлов и Интернет-адреса;
   - шрифтом Courier New выделены программы на языках программирования;
   - *курсивом* выделены названия диалоговых окон, вкладок и управляющих элементов графического интерфейса операционных систем и приложений;
   - **полужирным** начертанием шрифта выделены важные термины и понятия;
8. Важная информация и формулы выделены в тексте восклицательным знаком:

9. Абзацы, содержащие дополнительную интересную информацию, выделены значком .
10. Дополнительные материалы и интерактивные тесты для проверки усвоения материала находятся в Интернете по адресу: http://iit.metodist.ru

Глава 1

Кодирование и обработка графической и мультимедийной информации

1.1. Кодирование графической информации

1.1.1. Пространственная дискретизация

Графическая информация может быть представлена в **аналоговой** или **дискретной** форме. Примером аналогового представления графической информации может служить живописное полотно, цвет которого изменяется непрерывно, а дискретного — изображение, напечатанное с помощью струйного принтера, состоящее из отдельных точек разного цвета.

Графические изображения из аналоговой (непрерывной) формы в цифровую (дискретную) преобразуются путем **пространственной дискретизации**. Пространственную дискретизацию изображения можно сравнить с построением изображения из мозаики (большого количества маленьких разноцветных стекол). Изображение разбивается на отдельные маленькие фрагменты (точки, или **пиксели**), причем каждый элемент имеет свой цвет (красный, зеленый, синий и т. д.).

Пиксель — минимальный участок изображения, для которого 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 (рис. 1.1).

Рис. 1.1. Растровое изображение эмблемы операционной системы Linux

Разрешающая способность. Важнейшей характеристикой качества растрового изображения является разрешающая способность.

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

Чем меньше размер точки, тем больше разрешающая способность (так как больше количество строк 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).

Пространственная дискретизация непрерывных изображений, хранящихся на бумаге, фото- и кинопленке, может быть осуществлена путем сканирования. В настоящее время все большее распространение получают цифровые фото- и видеокамеры, которые фиксируют изображения сразу в дискретной форме.

Качество растровых изображений, полученных в результате сканирования, зависит от разрешающей способности сканера, которую производители указывают двумя числами (например, 1200 × 2400 dpi).

Сканирование производится путем перемещения полоски светочувствительных элементов вдоль изображения (рис. 1.2). Первое число является **оптическим разрешением** сканера и определяется количеством светочувствительных элементов на одном дюйме полоски. Второе число является **аппаратным разрешением** и

определяется количеством «микрошагов», которое может сделать полоска светочувствительных элементов, перемещаясь на один дюйм вдоль изображения.

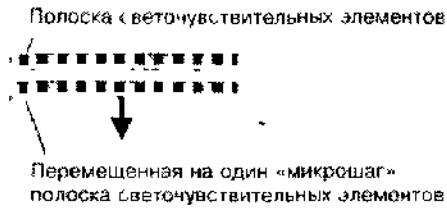


Рис. 1.2. Оптическое и аппаратное разрешение сканера

Глубина цвета. В процессе дискретизации могут использоваться различные **палитры цветов**, т. е. наборы тех 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:

$$N = 2^I. \qquad (1.1)$$

Информатика и ИКТ-8

В простейшем случае (черно-белое изображение без градаций серого цвета) палитра цветов состоит всего из двух цветов (черного и белого). Каждая точка экрана может принимать одно из двух состояний («черная» или «белая»). По формуле (1.1) можно вычислить, какое количество информации необходимо, чтобы закодировать цвет каждой точки:

$$2 = 2^I \quad \Rightarrow \quad 2^1 = 2^I \quad \Rightarrow \quad I = 1 \text{ бит}.$$

Количество информации, которое используется для кодирования цвета точки изображения, называется **глубиной цвета**.

Наиболее распространенными значениями глубины цвета при кодировании цветных изображений являются 8, 16 или 24 бита на точку. Зная глубину цвета, по формуле (1.1) можно вычислить количество цветов в палитре (табл. 1.1).

Таблица 1.1. **Глубина цвета и количество цветов в палитре**

Глубина цвета, *I* (битов)	Количество цветов в палитре, *N*
8	$2^8 = 256$
16	$2^{15} = 65\,536$
24	$2^{24} = 16\,777\,216$

Контрольные вопросы

1. Объясните, как с помощью пространственной дискретизации происходит формирование растрового изображения.
2. В каких единицах выражается разрешающая способность растровых изображений?
3. Как связаны между собой количество цветов в палитре и глубина цвета?

Задания для самостоятельного выполнения

1.1. *Задание с выборочным ответом.* В процессе преобразования растрового графического изображения количество цветов уменьшилось с 65 536 до 16. Во сколько раз уменьшился его информационный объем?

1) в 2 раза; 2) в 4 раза; 3) в 8 раз; 4) в 16 раз.

1.2. *Задание с кратким ответом.* Черно-белое (без градаций серого) растровое графическое изображение имеет размер 10 × 10 точек. Какой информационный объем имеет изображение?

1.3. *Задание с кратким ответом.* Цветное с палитрой из 256 цветов растровое графическое изображение имеет размер 10 × 10 точек. Какой информационный объем имеет изображение?

1.4. **Задание с развернутым ответом.* Сканируется цветное изображение размером 10 × 10 см. Разрешающая способность сканера 1200 × 1200 dpi, глубина цвета 24 бита. Какой информационный объем будет иметь полученный графический файл?

1.1.2. Растровые изображения на экране монитора

Графические режимы экрана монитора. Качество изображения на экране монитора зависит от величины **пространственного разрешения** и **глубины цвета**. Эти два параметра задают графический режим экрана монитора.

Пространственное разрешение экрана монитора определяется как произведение количества строк изображения на количество точек в строке. Монитор может отображать информацию с различными пространственными разрешениями (800 × 600, 1024 × 768, 1400 × 1050 и выше).

Глубина цвета измеряется в битах на точку и характеризует количество цветов, которое могут принимать точки изображения. Количество отображаемых цветов может изменяться в широком диапазоне, от 256 (глубина цвета 8 битов) до более чем 16 миллионов (глубина цвета 24 бита).

Чем больше пространственное разрешение и глубина цвета, тем выше качество изображения. В операционных системах предусмотрена возможность выбора необходимого пользователю и технически возможного графического режима.

Рассмотрим формирование на экране монитора растрового изображения, состоящего из 600 строк по 800 точек в каждой строке (всего 480 000 точек), с глубиной цвета 8 битов (рис. 1.3). Двоичные коды цветов всех точек хранятся в видеопамяти компьютера, которая находится на видеокарте.

Видеопамять	
Номер точки	**Двоичный код цвета точки**
1	01010101
2	10101010
..	
800	11110000
480000	11111111

1 2 3 4 ... 800

1 .....

2 .....

3 .....

...

600 .....

Рис. 1.3. Формирование растрового изображения на экране монитора

Периодически, с определенной частотой, коды цветов точек считываются из видеопамяти и точки отображаются на экране монитора. Частота считывания изображения вли-

яет на стабильность изображения на экране. В современных мониторах обновление изображения происходит с частотой 75 и более раз в секунду, что обеспечивает комфортность восприятия изображения пользователем компьютера (человек не замечает мерцания изображения). Для сравнения можно напомнить, что частота смены кадров в кино составляет 24 кадра в секунду.

Качество отображения информации на экране монитора *зависит от размера экрана и размера пикселя*. Зная размер диагонали экрана в дюймах (15", 17" и т. д.) и размер пикселя экрана (0,28, 0,24 мм или 0,20 мм), можно оценить максимально возможное пространственное разрешение экрана монитора.

Контрольные вопросы

1. С помощью каких параметров задается графический режим экрана монитора?
2. Как вы думаете, почему частота обновления изображения на экране монитора должна быть больше, чем частота кадров в кино?

Задания для самостоятельного выполнения

1.5. **Задание с развернутым ответом.* Определить максимально возможную разрешающую способность экрана для монитора с диагональю 17" и размером точки экрана 0,28 мм.

1.1.3. Палитры цветов в системах цветопередачи RGB, CMYK и HSB

Белый свет может быть разложен с помощью оптических приборов (например, призмы) или природных явлений (радуги) на различные цвета спектра: *красный*, *оранжевый*, *желтый*, *зеленый*, *голубой*, *синий* и *фиолетовый* (рис. 1.4).

Оптика Физика-8

Хорошо известна фраза, которая помогает легко запомнить последовательность цветов в спектре видимого света: «Каждый охотник желает знать, где сидит фазан».

Рис. 1.4. Разложение белого света в спектр

Человек воспринимает свет с помощью цветовых рецепторов, так называемых колбочек, находящихся на сетчатке глаза. Наибольшая чувствительность колбочек приходится на красный, зеленый и синий цвета, которые являются базовыми для человеческого восприятия. Сумма красного, зеленого и синего цветов воспринимается человеком как белый цвет, их отсутствие — как черный, а различные их сочетания — как многочисленные оттенки цветов.

Палитра цветов в системе цветопередачи RGB. С экрана монитора человек воспринимает цвет как сумму излучения трех базовых цветов: красного, зеленого и синего. Такая система цветопередачи называется RGB, по первым буквам английских названий цветов (*Red* — красный, *Green* — зеленый, *Blue* — синий).

Цвета в палитре RGB формируются путем сложения базовых цветов, каждый из которых может иметь различную интенсивность. Цвет палитры *Color* можно определить с помощью формулы (1.2):

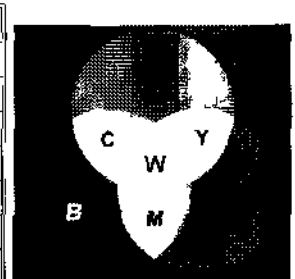
$$Color = R + G + B,$$
$$\text{где } 0 \le R \le Rmax,\ 0 \le G \le Gmax,\ 0 \le B \le Bmax. \qquad (1.2)$$

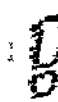
При минимальных интенсивностях всех базовых цветов получается черный цвет, при максимальных интенсивностях — белый цвет. При максимальной интенсивности одного цвета и минимальной двух других — красный, зеленый и синий цвета. Наложение зеленого и синего цветов образует голубой цвет (*Cyan*), наложение красного и зеленого цве-

тов — желтый цвет (*Yellow*), наложение красного и синего цветов — пурпурный цвет (*Magenta*) (табл. 1.2).

Таблица 1.2. **Формирование цветов в системе цветопередачи RGB**

Цвет	Формирование цвета
Черный	Black = 0 + 0 + 0
Белый	White = *Rmax* + *Gmax* + *Bmax*
Красный	Red = *Rmax* + 0 + 0
Зеленый	Green = 0 + *Gmax* + 0
Синий	Blue = 0 + 0 + *Bmax*
Голубой	Cyan = 0 + *Gmax* + *Bmax*
Пурпурный	Magenta = *Rmax* + 0 + *Bmax*
Желтый	Yellow = *Rmax* + *Gmax* + 0





В **системе цветопередачи RGB** палитра цветов формируется путем сложения красного, зеленого и синего цветов

При глубине цвета в 24 бита на кодирование каждого из базовых цветов выделяется по 8 битов. В этом случае для каждого из цветов возможны $N = 2^8 = 256$ уровней интенсивности. Уровни интенсивности задаются десятичными (от минимального — 0 до максимального — 255) или двоичными (от 00000000 до 11111111) кодами (табл. 1.3).

Таблица 1.3. **Кодировка цветов при глубине цвета 24 бита**

Цвет	Двоичный и десятичный коды интенсивности базовых цветов					
	Красный		Зеленый		Синий	
Черный	00000000	0	00000000	0	00000000	0
Красный	11111111	255	00000000	0	00000000	0
Зеленый	00000000	0	11111111	255	00000000	0
Синий	00000000	0	00000000	0	11111111	255
Голубой	00000000	0	11111111	255	11111111	255
Пурпурный	11111111	255	00000000	0	11111111	255
Желтый	11111111	255	11111111	255	00000000	0
Белый	11111111	255	11111111	255	11111111	255

Палитра цветов в системе цветопередачи CMYK. При печати изображений на принтерах используется палитра цветов в системе CMY. Основными красками в ней являются *Cyan* — голубая, *Magenta* — пурпурная и *Yellow* — желтая.

Цвета в палитре CMY формируются путем *наложения красок базовых цветов*. Цвет палитры *Color* можно определить с помощью формулы (1.3), в которой интенсивность каждой краски задается в процентах:

$$Color = C + M + Y, \qquad (1.3)$$
где $0\% \le C \le 100\%$, $0\% \le M \le 100\%$, $0\% \le Y \le 100\%$.

Напечатанное на бумаге изображение человек воспринимает в отраженном свете. Если на бумагу краски не нанесены, то падающий белый свет полностью отражается и мы видим белый лист бумаги. Если краски нанесены, то они поглощают определенные цвета спектра. Цвета в палитре CMY формируются путем *вычитания из белого света определенных цветов*.

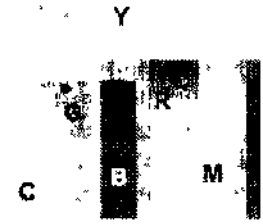
Нанесенная на бумагу голубая краска поглощает красный свет и отражает зеленый и синий свет, и мы видим голубой цвет. Нанесенная на бумагу пурпурная краска поглощает зеленый свет и отражает красный и синий свет, и мы видим пурпурный цвет. Нанесенная на бумагу желтая краска поглощает синий свет и отражает красный и зеленый свет, и мы видим желтый цвет.

Смешав две краски системы CMY, мы получим базовый цвет в системе цветопередачи RGB. Если нанести на бумагу пурпурную и желтую краски, то будет поглощаться зеленый и синий свет, и мы увидим красный цвет. Если нанести на бумагу голубую и желтую краски, то будет поглощаться красный и синий свет, и мы увидим зеленый цвет. Если нанести на бумагу пурпурную и голубую краски, то будет поглощаться зеленый и красный свет, и мы увидим синий цвет (табл. 1.4).

Смешение трех красок — голубой, желтой и пурпурной — должно приводить к полному поглощению света, и мы должны увидеть черный цвет. Однако на практике вместо черного цвета получается грязно-бурый цвет. Поэтому в цветовую модель добавляют еще один, истинно черный цвет. Так как буква B уже используется для обозначения синего цвета, для обозначения черного цвета принята последняя буква в английском названии черного цвета *Black*, т. е. K. Расширенная палитра получила название CMYK (табл. 1.4).

Таблица 1.4. **Формирование цветов в системе цветопередачи CMYK**

Цвет	Формирование цвета
Черный	Black = $K = C + M + Y = W - G - B - R$
Белый	White = W = ($C = 0$, $M = 0$, $Y = 0$)
Красный	Red = $R = Y + M = W - B - G$
Зеленый	Green = $G = Y + C = W - B - R$
Синий	Blue = $B = M + C = W - G - R$
Голубой	Cyan = $C = W - R = G + B$
Пурпурный	Magenta = $M = W - G = R + B$
Желтый	Yellow = $Y = W - B = R + G$



В системе цветопередачи CMYK палитра цветов формируется путем наложения голубой, пурпурной, желтой и черной красок.

Система цветопередачи RGB применяется в мониторах компьютеров, в телевизорах и других излучающих свет технических устройствах.

Система цветопередачи CMYK применяется в полиграфии, так как напечатанные документы воспринимаются человеком в отраженном свете. В струйных принтерах для получения изображений высокого качества используются четыре картриджа, содержащие базовые краски системы цветопередачи CMYK (рис. 1.5).

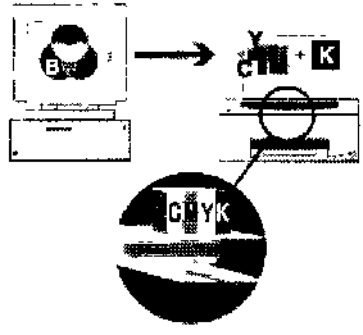
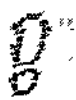


Рис. 1.5. Использование систем цветопередачи RGB и CMYK в технике

Палитра цветов в системе цветопередачи HSB. Система цветопередачи HSB использует в качестве базовых параметров *Hue* (оттенок цвета), *Saturation* (насыщенность) и *Brightness* (яркость). Параметр *Hue* позволяет выбрать оттенок цвета из всех цветов оптического спектра: от красного цвета до фиолетового (H = 0 — красный цвет, H = 120 — зеленый цвет, H = 240 — синий цвет, H = 360 — фиолетовый цвет). Параметр *Saturation* определяет процент «чистого» оттенка и белого цвета (S = 0% — белый цвет, S = 100% — «чистый» оттенок). Параметр *Brightness* определяет интенсивность цвета (минимальное значение B = 0 соответствует черному цвету, максимальное значение B = 100 соответствует максимальной яркости выбранного оттенка цвета).



В системе цветопередачи HSB палитра цветов формируется путем установки значений оттенка цвета, насыщенности и яркости.

В графических редакторах обычно имеется возможность перехода от одной модели цветопередачи к другой. Это можно сделать как с помощью мыши, перемещая указатель по цветовому полю, так и вводя параметры цветовых моделей с клавиатуры в соответствующие текстовые поля.

Контрольные вопросы

1. В каких природных явлениях и физических экспериментах можно наблюдать разложение белого света в спектр?
2. Как формируется палитра цветов в системе цветопередачи RGB? В системе цветопередачи CMYK? В системе цветопередачи HSB?

Задания для самостоятельного выполнения

1.6. *Задание с кратким ответом.* Определить цвета, если заданы интенсивности базовых цветов в системе цветопередачи RGB. Заполнить таблицу.

Цвет	Интенсивность базовых цветов		
	Красный	*Зеленый*	*Синий*
	00000000	00000000	00000000
	11111111	00000000	00000000
	00000000	11111111	00000000
	00000000	00000000	11111111
	00000000	11111111	11111111
	11111111	00000000	11111111
	11111111	11111111	00000000
	11111111	11111111	11111111

1.7. *Задание с кратким ответом.* Определить цвета, если на бумагу нанесены краски в системе цветопередачи CMYK. Заполнить таблицу.

Цвет	Формирование цвета
	$C = 0, M = 0, Y = 0$
	$Y + M = W - B - G$
	$Y + C = W - B - R$
	$M + C = W - G - R$
	$W - R = G + B$
	$W - G = R + B$
	$W - B = R + G$

1.2. Растровая и векторная графика

1.2.1. Растровая графика

Растровые изображения. Растровые изображения формируются в процессе сканирования многоцветных иллюстраций и фотографий, а также при использовании цифровых фото- и видеокамер. Можно создать растровое изображение непосредственно на компьютере с использованием растрового графического редактора.

Растровое изображение создается с использованием точек различного цвета (пикселей), которые образуют строки и столбцы. Каждый пиксель может принимать любой цвет из палитры, содержащей десятки тысяч или даже десятки миллионов цветов, поэтому растровые изображения обеспе-

чивают высокую точность передачи цветов и полутонов. Качество растрового изображения возрастает с увеличением пространственного разрешения (количества пикселей в изображении по горизонтали и вертикали) и количества цветов в палитре.

Недостатком растровых изображений является их большой информационный объем, так как необходимо хранить код цвета каждого пикселя.

Растровые изображения формируются из точек различного цвета (пикселей), которые образуют строки и столбцы.

Растровые изображения очень чувствительны к уменьшению и увеличению. При уменьшении растрового изображения несколько соседних точек преобразуются в одну, и поэтому теряется четкость мелких деталей изображения. При увеличении растрового изображения точки добавляются, в результате несколько соседних точек принимают одинаковый цвет и появляется ступенчатый эффект (рис. 1.6).

Рис. 1.6. Растровое изображение российского герба, его уменьшенная копия и увеличенный фрагмент

Растровые графические редакторы. Растровые графические редакторы являются наилучшим средством обработки цифровых фотографий и отсканированных изображений, поскольку позволяют повышать их качество путем изменения цветовой палитры изображения и даже цвета каждого отдельного пикселя. Можно повысить яркость и контрастность старых или некачественных фотографий, удалить мелкие дефекты изображения (например, царапины), преобразовать черно-белое изображение в цветное и т. д.

Кроме того, растровые графические редакторы можно использовать для художественного творчества путем применения различных эффектов преобразования изображения. Обычную фотографию можно превратить в мозаичное панно, рисунок карандашом или рельефное изображение (рис. 1.7).

Рис. 1.7. Эффекты преобразования изображения в растровом графическом редакторе

Форматы растровых графических файлов. Графические редакторы позволяют открывать, обрабатывать и сохранять изображения и рисунки в различных графических форматах. Форматы графических файлов определяют способ хранения информации в файле (растровый или векторный), а также форму хранения информации (используемый метод сжатия).

Универсальным форматом растровых графических файлов, т. е. форматом, который «понимают» все растровые графические редакторы, является формат BMP. Растровые графические файлы в этом формате имеют большой информационный объем, так как в них хранятся коды цветов всех точек изображения.

Для размещения изображений на Web-страницах в Интернете используются форматы растровых графических файлов, в которых используется сжатие. В растровом графическом формате GIF используется метод сжатия, который позволяет неплохо сжимать файлы, в которых много одноцветных областей изображения (логотипы, надписи, схемы). Файлы в формате GIF могут содержать не одну, а несколько растровых картинок, которые показываются одна за другой с указанной в файле частотой, чем достигается иллюзия движения (**GIF-анимация**). Недостатком формата GIF является ограниченная палитра, в которой не может быть больше 256 цветов.

Растровый графический формат PNG использует метод сжатия без потери данных и является усовершенствован-

ным вариантом формата GIF, так как позволяет использовать в палитре до 16 миллионов цветов. При сохранении файлов в этом формате можно указать требуемую степень сжатия на шкале «высокая степень сжатия и плохое качество изображения — низкая степень сжатия и высокое качество изображения».

Для сжатия цифровых и отсканированных фотографий используется формат JPEG. Компьютер обеспечивает воспроизведение более 16 миллионов различных цветов, тогда как человек вряд ли способен различить более сотни цветов и оттенков. В формате JPEG отбрасывается «избыточное» для человеческого восприятия разнообразие цветов соседних пикселей. Применение этого формата позволяет сжимать файлы в десятки раз, однако приводит к необратимой потере информации (файлы не могут быть восстановлены в первоначальном виде).

Контрольные вопросы

1. Почему при уменьшении и увеличении растрового изображения ухудшается его качество?
2. В чем состоят основные различия между форматами растровых графических файлов?

Задания для самостоятельного выполнения

1.8. *Задание с выборочным ответом.* Растровые изображения формируются из:
1) линий; 2) окружностей; 3) прямоугольников; 4) пикселей.

1.2.2. Векторная графика

Векторные рисунки используются для хранения высокоточных графических объектов (рисунков, чертежей и схем), для которых имеет значение сохранение четких и ясных контуров.

Векторные рисунки формируются из графических объектов (линия, прямоугольник, окружность и др.), для каждого из которых задаются координаты опорных точек (например, для рисования окружности достаточно знать координаты ее центра и радиус) и формулы рисования объекта. Для каждого объекта можно также указать цвет, толщину и стиль линии (сплошная, пунктирная и т. д.) его контура.

Векторные рисунки формируются из базовых графических объектов, для каждого из которых задаются координаты опорных точек, формулы рисования объекта, а также цвет, толщина и стиль линии его контура.

Достоинством векторной графики является то, что векторные рисунки могут быть увеличены или уменьшены без потери качества (рис. 1.8). Это возможно, так как изменение размера рисунка производится с помощью простого умножения координат точек графических объектов на коэффициент масштабирования.

Рис. 1.8. Векторный рисунок российского герба, его уменьшенная копия и увеличенный фрагмент

Другое достоинство векторной графики — небольшой информационный объем файлов по сравнению с объемом файлов, содержащих растровые изображения.

Векторная графика лежит в основе **flash-анимации**, популярной в настоящее время технологии создания анимации. Эта технология позволяет реализовать движение, плавно изменяя расположение, размер и цвет объектов на рисунке, а также показать плавное *превращение одного объекта в другой*.

Векторные графические редакторы. Векторные графические редакторы используются для создания и редактирования рисунков, в которых существуют четкие контуры (*эмблемы, иллюстрации к книге, визитки* и *плакаты*, этикетки, схемы, графики и чертежи). Так как векторные рисунки состоят из отдельных графических объектов, то они легко редактируются (каждый из объектов может быть перемещен, удален, увеличен или уменьшен и т. д.).

Векторные графические редакторы позволяют рисовать не только плоские, но и объемные объекты: куб, шар, цилиндр и другие. При рисовании трехмерных тел можно устанавливать различные режимы освещенности объекта, материал, из которого он изготовлен, качество поверхности и другие параметры.

При классическом черчении с помощью карандаша, линейки и циркуля производится построение элементов чертежа (отрезков, окружностей и прямоугольников) с точностью, которую предоставляют чертежные инструменты. **Системы компьютерного черчения** (рис. 1.9), которые являются векторными графическими редакторами, позволяют создавать чертежи с гораздо большей точностью. Такие системы дают возможность измерять расстояния, углы, периметры и площади начерченных объектов.

Системы автоматизированного проектирования *используются на производстве, так как обеспечивают возможность* реализации сквозной технологии проектироания и изготовления деталей. На основе компьютерных чертежей создаются управляющие программы для станков с числовым программным управлением. Затем по компьютерным чертежам изготавливаются высокоточные детали из металла, пластмассы, дерева и других материалов.

Форматы векторных графических файлов. Широко распространенным форматом векторных графических файлов является формат WMF. Этот формат используется для хранения коллекции графических изображений Microsoft Clip Gallery. Некоторые программы обработки изображений используют оригинальные форматы, которые распознаются только самой создающей программой. Например, векторный редактор OpenOffice.org Draw сохраняет файлы в собственном формате ODG.

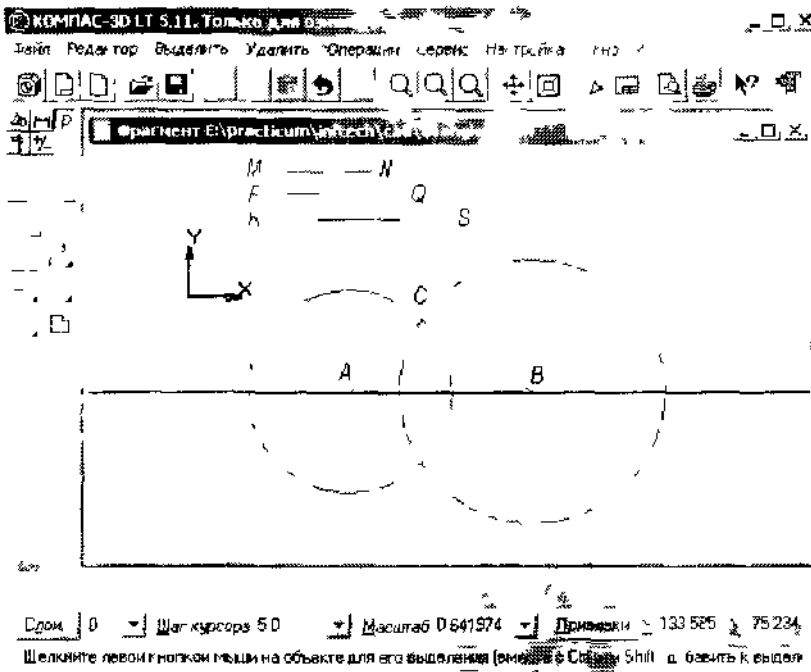


Рис. 1.9. Система компьютерного черчения Компас

Контрольные вопросы

1. В чем состоит различие между растровыми изображениями и векторными рисунками?
2. Какой графический редактор (растровый или векторный) вы будете использовать:
   - для разработки эмблемы организации, учитывая, что она должна будет печататься на маленьких визитных карточках и на больших плакатах;
   - для редактирования цифровой фотографии?

1.3. Интерфейс и основные возможности графических редакторов

1.3.1. Рисование графических примитивов в растровых и векторных графических редакторах

Область рисования. Для создания рисунка традиционными методами необходимо выбрать полотно (лист бумаги или холст) определенного размера и ориентации. В графических редакторах можно выбрать параметры области рисования (размер, поля и ориентацию — рис. 1.10), которая называется страницей, листом или слайдом.

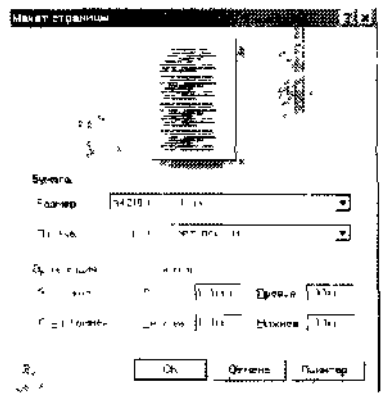


Рис. 1.10. Параметры области рисования

Область рисования может иметь различные размеры. Наиболее распространенным является формат А4, который соответствует размеру стандартного листа писчей бумаги (шириной 21 см и высотой 29,7 см), часто используются в половину меньший формат А5 (шириной 14,8 см и высотой 21 см) или в два раза больший формат А3 (шириной 29,7 см и высотой 42 см).

Рисунком можно занять всю площадь области рисования или оставить по краям поля. Поля оставлять рекомендуется, так как не все принтеры могут распечатывать листы без полей.

Область рисования можно расположить вертикально (ширина листа меньше высоты) — такая ориентация называется книжной. Область рисования можно также расположить горизонтально (ширина листа больше высоты) — такая ориентация называется альбомной.

Технология рисования графических примитивов. Растровые и векторные графические редакторы позволяют рисовать в поле рисования графические примитивы (прямая линия, кривая линия, прямоугольник, многоугольник и окружность).

Кнопки для рисования графических примитивов находятся на *Панели инструментов*, которая обычно размеща-

ется вертикально, вдоль левого края окна графического редактора (рис. 1.11). Для рисования выбранного объекта необходимо щелкнуть по кнопке с его изображением на *Панели инструментов* и переместить указатель мыши в поле рисования, где указатель примет форму крестика. Затем щелчками в поле рисования требуется зафиксировать положения опорных точек рисуемого объекта.

Процедуры рисования графических примитивов в растровом и векторном редакторах практически не различаются, однако существенно различаются результаты рисования. В растровом графическом редакторе нарисованный объект перестает существовать как самостоятельный элемент после окончания рисования и становится лишь группой пикселей на рисунке. В векторном редакторе этот объект продолжает сохранять свою индивидуальность и его можно копировать, перемещать, изменять размеры, цвет и прозрачность.

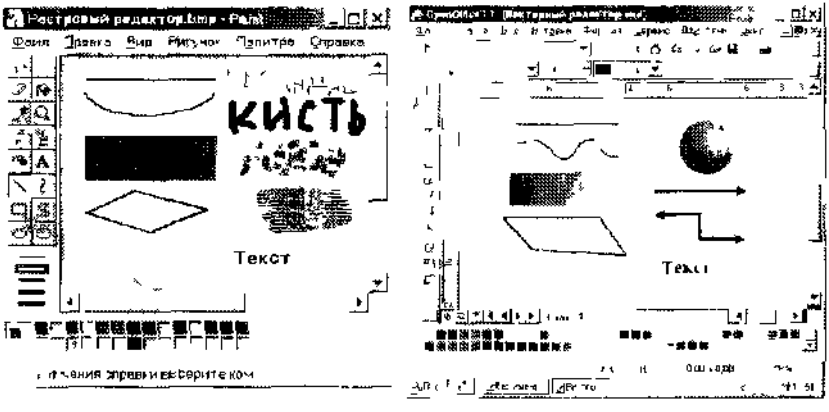


Рис. 1.11. Графические редакторы растровый Microsoft Paint и векторный OpenOffice Draw

Линия. Для рисования линии необходимо выбрать на *Панели инструментов* графический примитив *Линия*, переместить указатель мыши в определенное место окна редактора и щелчком мышью зафиксировать точку, из которой должна начинаться линия. Затем перетащить линию в нужном направлении и, осуществив повторный щелчок, зафиксировать второй конец линии.

Существует возможность перед рисованием задать тип линии (сплошная, пунктирная и т. д.), ее толщину и цвет с помощью дополнительных меню.

Кривая. Для рисования кривой необходимо выбрать графический примитив *Кривая*, нарисовать произвольную линию и перетаскиванием мышью придать ей требуемую форму.

Прямоугольник. Для рисования прямоугольника необходимо выбрать графический примитив *Прямоугольник*, щелчком зафиксировать положение первой вершины, перетащить указатель по диагонали прямоугольника и зафиксировать положение противоположной вершины.

Многоугольник. Для рисования многоугольника необходимо выбрать графический примитив *Многоугольник*, последовательно щелчками мышью зафиксировать положение вершин и двойным щелчком зафиксировать положение последней вершины.

Овал и окружность. Для рисования овала необходимо выбрать графический примитив *Овал* (*Эллипс*), щелчком мышью зафиксировать положение точки овала, перетащить указатель и зафиксировать положение точки, противоположной данной относительно центра овала. Если в процессе рисования держать нажатой клавишу {*Shift*}, то будет нарисована окружность.

Палитра цветов. Различают *цвет линии*, которым рисуется контур геометрического примитива, и *цвет заливки*, которым он закрашивается. Операцию выбора цвета можно осуществлять с помощью меню *Палитра*, содержащего набор цветов, используемых при создании объектов.

Расширенная палитра. Выбор цвета с использованием меню *Палитра* ограничен, так как оно содержит только несколько десятков цветов. Однако графические редакторы позволяют использовать расширенную палитру цветов, в которой можно осуществлять выбор среди набора из десятков миллионов цветов.

1.1.3. Палитры цветов в системах цветопередачи RGB, CMYK и HSB

Пипетка. В растровых графических редакторах для копирования цветов можно использовать инструмент *Пипетка*. Щелчок мышью на области с требуемым цветом задает его в качестве цвета линии или цвета заливки.

Если навести указатель мыши на кнопку на *Панели инструментов* графического редактора, то появится всплывающая подсказка с названием инструмента.

Контрольные вопросы

1. Перечислите основные параметры области рисования.
2. Перечислите графические примитивы и опишите процедуры их рисования.
3. Как можно задать цвет линии и цвет заливки?

1.3.2. Инструменты рисования растровых графических редакторов

Для создания изображения традиционными методами необходимо выбрать инструмент рисования (это могут быть фломастеры, кисть с красками, карандаши и многое другое). В растровых графических редакторах существуют аналогичные инструменты, позволяющие изменять цвет определенных групп пикселей. Кнопки инструментов рисования обычно располагаются на *Панели инструментов*.

Карандаш. Инструмент *Карандаш* позволяет рисовать произвольные тонкие линии.

Кисть. Инструмент *Кисть* позволяет рисовать произвольные линии различной толщины с использованием «кисти» выбранной формы.

Ластик. Инструмент *Ластик* (*Кисть*, рисующая цветом фона) позволяет стирать произвольные пиксели изображения, при этом размер *Ластика* можно менять.

Распылитель. Инструмент *Распылитель* позволяет разбрызгивать «краску» (закрашивать пиксели случайным образом) и таким образом закрашивать произвольные области.

Заливка. Инструмент *Заливка* позволяет закрашивать замкнутые области целиком.

Лупа. Инструмент *Лупа* позволяет увеличивать или уменьшать масштаб представления изображения на экране, но не влияет при этом на его реальные размеры.

Надпись. Инструмент *Надпись* (кнопка с буквой **А** на панели инструментов) позволяет создавать текстовые области на пиксельных изображениях. Установив курсор внутри текстовой области, можно произвести ввод текста, который становится частью пиксельного изображения.

Форматирование текста производится с помощью диалогового окна *Шрифты* (рис. 1.12).

Рис. 1.12. Диалоговое окно *Шрифты* растрового редактора

Контрольные вопросы

1. Перечислите основные инструменты рисования растровых графических редакторов и опишите их возможности.

1.3.3. Работа с объектами в векторных графических редакторах

Векторный графический редактор можно рассматривать как графический конструктор, который позволяет создавать рисунки из отдельных объектов (графических примитивов).

Слои объектов. Каждый графический примитив рисуется в своем слое, поэтому рисунки состоят из множества слоев. Графические примитивы можно накладывать друг на друга, *при этом одни объекты могут заслонять другие.* Например, если сначала был нарисован прямоугольник, а затем поверх него окружность, то слой окружности будет располагаться поверх слоя прямоугольника и окружность заслонит прямоугольник.

Существует возможность изменения видимости объектов путем изменения порядка размещения их слоев на рисунке. Для этого используются операции изменения порядка, которые позволяют перемещать выделенный объект на передний план (в самый верхний слой рисунка) или на задний план (самый нижний слой рисунка), а также на один слой вперед или назад.

Градиентная заливка объектов. В векторных редакторах существует возможность осуществлять градиентную заливку объектов. При градиентной заливке интенсивность *закраски может изменяться по длине, ширине или от цен-*

тра объекта. Кроме того, объекты могут быть заштрихованы различным способом (линиями, квадратами и т. д.).

Прозрачность объектов. Для каждого объекта (слоя рисунка) можно задать степень прозрачности (в процентах от 0 до 100). При нулевой прозрачности объект, нарисованный на нижерасположенном слое, виден не будет. Наоборот, при стопроцентной прозрачности он будет виден полностью.

Например, можно сначала нарисовать светлый прямоугольник и поверх него темный круг. Затем переместить светлый прямоугольник на передний план. Наконец, сделать прямоугольник частично прозрачным (рис. 1.13).

Рис. 1.13. Изменение видимости объектов в векторном графическом редакторе

Группировка объектов. В векторном редакторе отдельные графические примитивы можно преобразовать в единый объект (сгруппировать). С этим новым объектом можно производить те же действия, что и с графическими примитивами, т. е. перемещать, изменять его размеры, цвет и другие параметры. Можно и, наоборот, разбить объект, состоящий из нескольких, на самостоятельные объекты (разгруппировать).

Например, олимпийскую эмблему можно нарисовать с помощью пяти окружностей разного цвета. Затем, сгруппировав их в один объект, можно произвольно изменять размер, пропорции и цвет олимпийской эмблемы (рис. 1.14).

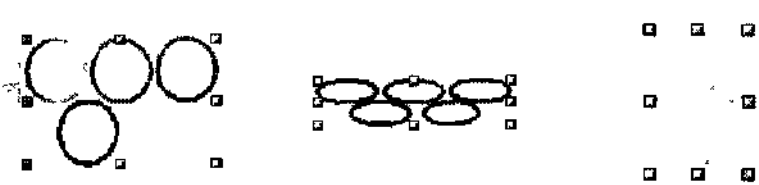


Рис. 1.14. Преобразования объекта, полученного с помощью операции группировки

Выравнивание объектов. Для большей точности рисования объектов в окне редактора по горизонтали и по вертикали размещаются *линейки с делениями* (рис. 1.15).

Для выравнивания нарисованных объектов по горизонтали и вертикали используется *сетка*, к которой привязываются объекты. Точность привязки объектов можно менять, изменяя размер ячеек сетки.

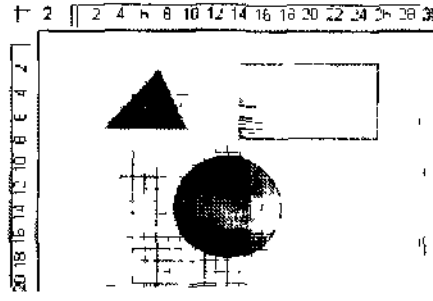


Рис. 1.15. Линейки и сетка

Выноски в векторных редакторах. В векторных редакторах можно создавать текстовые области, в которых можно вводить и форматировать текст. Кроме этого, для ввода надписей к рисункам можно использовать так называемые **выноски** различных форм (рис. 1.16). Текстовые области и выноски существуют в рисунке как самостоятельные объекты и поэтому могут легко масштабироваться и перемещаться.

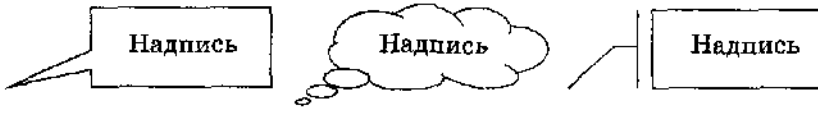


Рис. 1.16. Выноски в векторном редакторе

Контрольные вопросы

1. Что в векторных графических редакторах позволяет изменять видимость объектов, образующих рисунок?
2. В каких случаях полезно воспользоваться операцией группировки объектов?

1.3.4. Редактирование изображений и рисунков в растровых и векторных графических редакторах

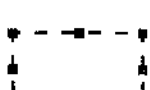
Редактирование изображения (рисунка) может производиться с использованием трех основных операций: копирования, перемещения и удаления. Перед выполнением каждой операции редактирования необходимо выделить область изображения в растровом редакторе или объект в векторном редакторе.

Выделение областей изображения и объектов в рисунках. В растровом графическом редакторе для выделения областей изображения (групп пикселей) используются два инструмента: *Выделение прямоугольной области* и *Выделение произвольной области.* Процедура выделения производится аналогично процедуре рисования. После окончания процедуры выделения остается пунктирный контур выделенной области с восьмью метками по периметру (рис. 1.17).

В векторном редакторе выделение объектов осуществляется с помощью инструмента *Выделение объекта* (на *Панели инструментов* изображается стрелкой). Для выделения объекта достаточно выбрать инструмент *Выделение объекта* и щелкнуть по любому объекту на рисунке. Вокруг выделенного объекта появятся восемь меток в виде маленьких квадратиков по его периметру (см. рис. 1.17).

Если поместить указатель мыши на такую метку, то он примет вид стрелки, направленной в две противоположные стороны (см. рис. 1.17). Перетаскивая метку, можно изменять размер объекта.

Для перемещения выделенной области (объекта) необходимо установить курсор внутри выделенной области (указатель мыши примет вид стрелки, указывающей «на все четыре стороны» (см. рис. 1.17)) и перетащить ее.



Выделение прямоугольной области в растровом редакторе

Выделение прямоугольной области в векторном редакторе

Рис. 1.17. Операция выделения

Копирование, перемещение и удаление областей растровых изображений и объектов в векторных рисунках. При копировании сначала выделяется область растрового

изображения или объект в векторном рисунке. Затем копия выделенной растровой области или векторного объекта помещается в специальную область памяти операционной системы, которая называется буфером обмена. Наконец копия из буфера обмена помещается в область рисования и перетаскивается мышью в нужное место растрового изображения или векторного рисунка.

При перемещении выделенная область растрового изображения или объект в векторном рисунке удаляется, а его копия помещается в буфер обмена. Затем копия из буфера обмена помещается в область рисования и перетаскивается мышью в нужное место растрового изображения или векторного рисунка.

При удалении выделенная область растрового изображения или объект в векторном рисунке просто удаляется.

Для иллюстрации вышесказанного нарисуем в растровом и векторном редакторах темный круг и светлый прямоугольник поверх него и выполним копирование, перемещение и удаление светлого прямоугольника (рис. 1.18).

В растровом редакторе исходное изображение представляет собой группу пикселей (пиксели светлого прямоугольника встроены в пиксели темного круга), и при выполнении перемещения и удаления пиксели, входящие в выделенный прямоугольник, удаляются.

В векторном редакторе исходное изображение представляет собой совокупность двух объектов (светлый прямоугольник наложен на темный круг) и при выполнении перемещения и удаления удаляется только выделенный объект (светлый прямоугольник).

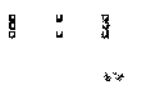
Вид графического редактора	Операция редактирования			
	Выделение	Копирование	Перемещение	Удаление
Растровый редактор				
Векторный редактор				

Рис. 1.18. Операции редактирования в растровом и векторном графических редакторах

Как видно из рис. 1.18, результаты выполнения операций перемещения и удаления, произведенных в растровом и векторном редакторах, различаются.

Геометрические преобразования областей растровых изображений и объектов в векторных рисунках. Изображения и рисунки могут быть подвергнуты геометрическим преобразованиям:

- изменению размера по горизонтали и вертикали;
- поворотам по часовой стрелке или против часовой стрелки;
- наклонам на различные углы;
- отражениям в различных плоскостях.

На рис. 1.19 показан результат последовательного применения к надписи операций растяжения по горизонтали, наклона на 45° и поворота на 180°.

Рис. 1.19. Геометрические преобразования надписи

Контрольные вопросы

1. Чем различаются результаты операции выделения в растровом и векторном графических редакторах?
2. Чем различаются результаты операций копирования, перемещения и удаления в растровом и векторном графических редакторах?

1.4. Растровая и векторная анимация

Анимация. При работе с растровыми изображениями и векторными рисунками широко используется **анимация**, т. е. создание иллюзии движения объектов на экране монитора. Компьютерная анимация использует быструю смену кадров (как это делается в кино), которую глаз человека воспринимает как непрерывное движение. Чем большее количество кадров меняется за одну секунду (в кино в секунду сменяется 24 кадра), тем более полная иллюзия движения возникает у человека.

Например, для создания компьютерной анимации, показывающей движение Земли вокруг Солнца, необходимо создать последовательность кадров, на которых нарисованы положения Земли на орбите (рис. 1.20). Для создания иллюзии движения требуется осуществить их быстрый последовательный вывод на экран монитора.

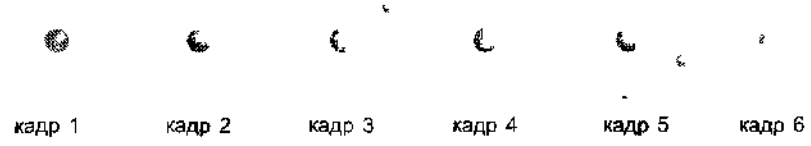


Рис. 1.20. *Последовательность кадров для создания анимации*

Анимация в презентациях. Программы разработки презентаций позволяют выбрать один из типов анимационных эффектов, который будет реализовываться в процессе смены слайдов. Например, при использовании эффекта *Наплыв влево* следующий слайд будет появляться постепенно, наезжая на предыдущий слайд справа налево.

Анимационные эффекты можно использовать и при размещении объектов на слайдах. Для каждого объекта можно выбрать наиболее подходящий эффект: постепенно проявиться, вылететь сбоку, развернуться до заданного размера, уменьшиться, двигаться по выбранной траектории, вспыхнуть, вращаться и т. д.

Анимационные эффекты позволяют привлечь внимание при размещении на слайде длинного текста: текст может появиться целиком, по словам или даже по отдельным буквам.

Gif-анимация. GIF-анимация является последовательностью растровых графических изображении (кадров), которые хранятся в одном растровом графическом файле в формате GIF. В процессе просмотра такого GIF-файла растровые графические изображения последовательно появляются на экране монитора, что и создает иллюзию движения.

Для создания последовательности растровых изображений и для их преобразования в GIF-анимацию можно использовать многофункциональные растровые редакторы или специальные редакторы GIF-анимаций.

Например, для получения анимации, демонстрирующей вращение Земли (рис. 1.21), необходимо:

- в растровом графическом редакторе создать последовательность растровых изображений фаз ее вращения;
- в GIF-аниматоре собрать из этих изображений анимацию.

Рис. 1.21. Последовательность кадров в GIF-анимации, демонстрирующей вращение Земли

При создании GIF-анимации можно задать величину задержки появления каждого кадра, чем она меньше, тем лучше качество анимации. Кроме того, можно установить количество повторений (от одного до бесконечности) последовательности кадров, хранящихся в GIF-файле.

Большое количество кадров ведет к лучшему качеству анимации, но при этом увеличивает размер GIF-файла. Для уменьшения его информационного объема можно анимировать только некоторые части изображения.

Flash-анимация. Flash-анимация базируется на использовании векторной графики и представляет собой последовательность векторных рисунков (кадров). Кадр строится с использованием набора векторных графических объектов (прямых и произвольных линий, окружностей и прямоугольников), для каждого из которых можно задать размер, цвет линии и заливки и другие параметры.

Достоинством flash-анимации является то, что нет необходимости прорисовывать каждый кадр. Достаточно нарисовать **ключевые кадры** и задать тип перехода между ними (свободная трансформация, трансформация с вращением, трансформация с отражением и т. д.). Редактор flash-анимации автоматически построит промежуточные кадры. Если промежуточных кадров много, то анимация получается плавной, а если мало, то быстрой.

Например, для создании анимации, демонстрирующей преобразование синего квадрата сначала в зеленый треугольник, а затем в красный круг*, достаточно (рис. 1.22):

- в ключевых кадрах (первом, четвертом и седьмом) нарисовать вышеперечисленные объекты;
- задать тип анимационного перехода между ними.

* На рис. 1.22. синий, зеленый и красный цвета переданы оттенками серого.

Рис. 1.22. Последовательность кадров flash-анимации преобразования синего квадрата в зеленый треугольник и красный круг

В процессе просмотра flash-анимации векторные кадры последовательно появляются на экране монитора, что и создает иллюзию движения. При создании flash-анимации можно задать количество кадров в секунду, чем оно больше, тем лучше качество анимации.

Достоинством flash-анимации является небольшой информационный объем файлов, и поэтому flash-анимация широко используется на Web-сайтах в Интернете. Для разработки flash-анимации используются специализированные flash-редакторы (например, Macromedia Flash), которые сохраняют анимационные файлы в специализированном формате **FLA**.

Контрольные вопросы

1. Объясните технологию создания компьютерной анимации.
2. Какие типы анимации могут быть использованы в презентациях?
3. Как можно ускорить или замедлить GIF-анимацию?
4. В чем состоит различие между ключевыми и обычными кадрами flash-анимации?

1.5. Кодирование и обработка звуковой информации

Звуковая информация. Звук представляет собой распространяющуюся в воздухе, воде или другой среде волну (колебания воздуха или другой среды) с непрерывно меняющейся **амплитудой** и **частотой**. Человек воспринимает звуковые волны с помощью слуха в форме звука различной

громкости и **тона**. Чем больше амплитуда звуковой волны, тем громче звук, чем больше частота колебаний, тем выше тон звука (рис. 1.23).

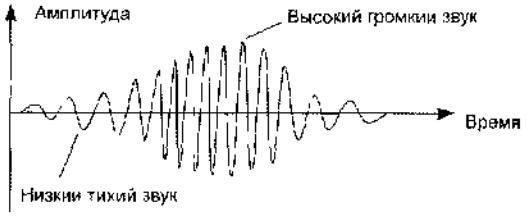


Рис. 1.23. Звуковая волна

Человеческое ухо воспринимает звук с частотой от 20 колебаний в секунду (низкий звук) до 20 000 колебаний в секунду (высокий звук).
Человек может воспринимать звук в огромном диапазоне амплитуд, в котором максимальная амплитуда больше минимальной в 10^{14} раз (в сто тысяч миллиардов раз). Для измерения громкости звука применяется специальная единица **децибел** (дБ). Уменьшение или увеличение громкости звука на 10 дБ соответствует уменьшению или увеличению амплитуды звука в 10 раз.

Таблица 1.5. **Громкость звука**

Звук	Громкость, дБ
Нижний предел чувствительности человеческого уха	0
Шорох листьев	10
Разговор	60
Гудок автомобиля	90
Реактивный двигатель	120
Болевой порог	140

Временна́я дискретизация звука. Для того чтобы компьютер мог обрабатывать звук, непрерывный звуковой сигнал должен быть преобразован в цифровую дискретную форму с помощью **временно́й дискретизации.** Непрерывная звуковая волна разбивается на отдельные маленькие временны́е участки, причем для каждого такого участка устанавливается определенный уровень громкости.

Таким образом, непрерывная зависимость громкости звука от времени $A(t)$ заменяется на дискретную последовательность уровней громкости. На графике это выглядит как замена гладкой кривой на последовательность «ступенек» (рис. 1.24).

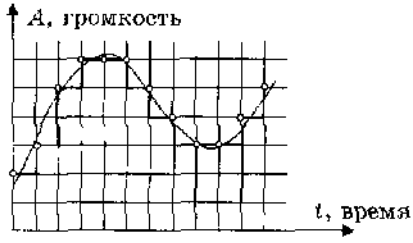


Рис. 1.24. *Временна́я дискретизация звука*

Частота дискретизации. Для записи аналогового звука и его преобразования в цифровую форму используется микрофон, подключенный к звуковой плате. Качество полученного цифрового звука зависит от количества измерений громкости звука в единицу времени, т. е. **частоты дискретизации**. Чем большее количество измерений производится за одну 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.

Глубина кодирования. Каждой «ступеньке» присваивается определенный уровень громкости звука. Уровни громкости звука можно рассматривать как набор N возможных состояний, 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 (1.1). Пусть глубина кодирования звука составляет 16 битов, тогда количество уровней громкости звука равно:

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

В процессе кодирования каждому уровню громкости звука присваивается свой 16-битовый двоичный код, наименьшему уровню громкости будет соответствовать код 0000000000000000, а наибольшему — 1111111111111111.

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

Необходимо помнить, что чем выше качество цифрового звука, тем больше информационный объем звукового файла. Можно оценить информационный объем цифрового стереозвукового файла длительностью звучания одна секунда при среднем качестве звука (16 битов, 24 000 измерений в секунду). Для этого глубину кодирования необходимо умножить на количество измерений в одну секунду и умножить на 2 (стереозвук):

$$16 \text{ битов} \cdot 24\,000 \cdot 2 = 768\,000 \text{ битов} = 96\,000 \text{ байтов} = 93{,}75 \text{ Кбайт}.$$

Звуковые редакторы. Звуковые редакторы позволяют не только записывать и воспроизводить звук, но и редактировать его. Оцифрованный звук представляется в звуковых редакторах в наглядной форме, поэтому операции копирования, перемещения и удаления частей звуковой дорожки можно легко осуществлять с помощью мыши. Кроме того, можно накладывать звуковые дорожки друг на друга (микшировать звуки) и применять различные акустические эффекты (эхо, воспроизведение в обратном направлении и др.).

Звуковые редакторы позволяют изменять качество цифрового звука и объем звукового файла путем изменения частоты дискретизации и глубины кодирования. Оцифрованный звук можно сохранять без сжатия в звуковых файлах

в универсальном формате WAV, а также в формате со сжатием MP3.

При сохранении звука в форматах со сжатием отбрасываются «избыточные» для человеческого восприятия звуковые частоты с малой амплитудой, совпадающие по времени со звуковыми частотами с большой амплитудой. Применение такого формата позволяет сжимать звуковые файлы в десятки раз, однако приводит к необратимой потере информации (файлы не могут быть восстановлены в первоначальном виде).

Контрольные вопросы

1. Объясните, как частота дискретизации и глубина кодирования влияют на качество цифрового звука.

Задания для самостоятельного выполнения

1.9. *Задание с выборочным ответом.* Звуковая плата производит двоичное кодирование аналогового звукового сигнала. Какое количество информации необходимо для кодирования каждого из 65 536 возможных уровней громкости сигнала?
1) 65 536 битов; 2) 256 битов; 3) 16 битов; 4) 8 битов.

1.10. *Задание с развернутым ответом.* Оценить информационный объем цифровых звуковых файлов длительностью 10 секунд при глубине кодирования и частоте дискретизации звукового сигнала, обеспечивающих минимальное и максимальное качество звука:
а) моно, 8 битов, 8000 измерений в секунду;
б) стерео, 16 битов, 48 000 измерений в секунду.

1.11. **Задание с развернутым ответом.* Определить длительность звукового файла, который уместится на дискете 3,5". Учесть, что для хранения данных на такой дискете выделяется 2847 секторов объемом 512 байтов каждый:
а) при низком качестве звука: моно, 8 битов, 8000 измерений в секунду;
б) при высоком качестве звука: стерео, 16 битов, 48 000 измерений в секунду.

1.6. Цифровое фото и видео

Цифровая фотография. Цифровые фотокамеры позволяют получить изображение высокого качества непосредственно в цифровом формате. Полученное цифровое изображение сохраняется в цифровой камере на сменной карте flash-памяти. После подключения цифровой камеры к USB-порту компьютера производится копирование изображений на жесткий диск компьютера (рис. 1.25). При необходимости можно провести редактирование фотографии с помощью растрового графического редактора. Высококачественная цветная печать цифровых фотографий производится на струйном принтере.

Рис. 1.25. Цифровая фотография

Размер растровых цифровых фотографий может достигать 3000 × 2000 точек при глубине цвета 24 бита на точку. Если сохранить фотографию на карте flash-памяти в формате BMP, информационный объем такого изображения получается достаточно большой:

I = 24 бита · 3000 · 2000 = 144 000 000 бита =
= 18 000 000 байтов ≈ 17578 Кбайта ≈ 17 Мбайт.

Возможность хранения на карте flash-памяти десятков цифровых фотографий обеспечивается использованием графического формата со сжатием по методу JPEG.

Цифровое видео. Цифровые видеокамеры позволяют снимать видеофильмы непосредственно в цифровом формате. Цифровое видео, представляющее собой последовательность кадров с определенным разрешением, сохраняется в видеокамере на магнитной кассете. После подключения цифровой видеокамеры к DV-порту компьютера и запуска программы цифрового видеомонтажа производится захват и копирование видео на жесткий диск компьютера (рис. 1.26).

В процессе захвата программа цифрового видеомонтажа автоматически обнаруживает изменения изображения в потоке видео и разбивает видео на фрагменты, называемые сценами. Пользователь в процессе монтажа может разбивать видео

Рис. 1.26. Цифровое видео

на сцены по времени или произвольно. Монтаж цифрового видеофильма производится путем выбора лучших сцен и размещения их в определенной временной последовательности. При переходе между сценами можно использовать различные анимационные эффекты: наплыв, растворение и др.

Просмотр цифрового видео можно осуществлять непосредственно на экране монитора компьютера или на подключенном телевизоре.

Видеофильм состоит из потока сменяющих друг друга кадров и звука. Показ полноцветных кадров и воспроизведение высококачественного звука требуют передачи очень больших объемов информации в единицу времени. Поэтому в процессе захвата и сохранения видеофайла на диске производится его сжатие. Во-первых, используются методы сжатия неподвижных растровых графических изображений и звука, описанные выше.

1.2.1. Растровая графика
1.5. Кодирование и обработка звуковой информации

Во-вторых, используется потоковое сжатие. В последовательности кадров выделяются сцены, в которых изображение меняется незначительно. Затем в сцене выделяется **ключевой кадр**, на основании которого строятся следующие, **зависимые кадры**. В зависимых кадрах вместо передачи кодов цвета всех пикселей передаются коды цвета только небольшого количества пикселей — те, которые были изменены.

Телевизионный стандарт воспроизведения видео использует разрешение кадра 720 × 576 пикселей с 24-битовой глубиной цвета. Скорость воспроизведения составляет 25 кадров в секунду. Следовательно, в одну секунду необходимо передать огромный объем видеоданных:

I = 24 бита · 720 · 576 · 25 = 248 832 000 битов ≈
≈ 31 104 000 байтов ≈ 30375 Кбайт ≈ 30 Мбайт.

При захвате и сохранении цифрового видео может использоваться один из двух способов сжатия данных.

При сохранении видеофайлов в формате AVI могут использоваться различные методы, использующие «фирменные» алгоритмы сжатия данных. При сохранении видеофайлов в формате MPEG используется стандартизированный метод сжатия данных.

Потоковое видео. Для передачи видео в Интернет к USB-порту компьютера подключается Web-камера (рис. 1.27). Так как скорость передачи данных в Интернете ограничена, используются потоковые методы сжатия с использованием одного из двух стандартов: **RealVideo** или **Windows Media**.

Рис. 1.27. Потоковое видео

Потоковое сжатие применяется как для видео, так и для звука. Сжатие видео обеспечивается за счет уменьшения размера кадра, уменьшения частоты кадров, а также уменьшения количества цветов. Для сжатия звука можно уменьшить частоту дискретизации и глубину кодирования, а также вместо стерео выбрать монофонический звук (один канал).

Однако в связи с широким распространением широкополосного высокоскоростного подключения к Интернету качество потокового видео и звука существенно улучшилось.

1.5. Кодирование и обработка звуковой информации

Контрольные вопросы

1. Опишите процесс получения цифровых фотографий.
2. Опишите основные этапы создания цифрового видеофильма.
3. Как можно уменьшить информационный объем потокового видео, передающегося в единицу времени по компьютерным сетям?

Практические работы компьютерного практикума, рекомендуемые для выполнения в процессе изучения главы 1

Компьютерный практикум

1.1 Кодирование графической информации.

1.2 Редактирование изображений в растровом графическом редакторе.

1.3. Создание рисунков в векторном графическом редакторе.

1.4 Создание GIF- и flash-анимации.

1.5. Кодирование и обработка звуковой информации.

1.6. Захват и редактирование цифрового фото и создание слайд-шоу

1.7 Захват и редактирование цифрового видео с использованием системы нелинейного видеомонтажа

Глава 2

Кодирование и обработка текстовой информации

2.1. Кодирование текстовой информации

Двоичное кодирование текстовой информации в компьютере. Информация, выраженная с помощью естественных и формальных языков в письменной форме, обычно называется текстовой информацией.

Для представления текстовой информации (прописные и строчные буквы русского и латинского алфавитов, цифры, знаки и математические символы) достаточно 256 различных знаков. По формуле (1.1) можно вычислить, какое количество информации необходимо, чтобы закодировать каждый знак:

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

1.3.2. Определение количества информации **Информатика и ИКТ-8**

Для обработки текстовой информации в компьютере необходимо представить ее в двоичной знаковой системе. Для кодирования каждого знака требуется количество информации, равное 8 битам, т. е. длина двоичного кода знака составляет восемь двоичных знаков. Каждому знаку необходимо поставить в соответствие уникальный двоичный код в интервале от 00000000 до 11111111 (в десятичном коде от 0 до 255) (табл. 2.1).

Человек различает знаки по их начертанию, а компьютер — по их двоичным кодам. При вводе в компьютер текстовой информации происходит ее двоичное кодирование, изображение знака преобразуется в его двоичный код. Пользователь нажимает на клавиатуре клавишу со знаком, и в компьютер поступает определенная последовательность из восьми электрических импульсов (двоичный код знака). Код знака хранится в оперативной памяти компьютера.

Таблица 2.1. **Кодировки знаков**

Двоичный код	Десятичный код	КОИ-8	Windows	MS-DOS	Mac	ISO
00000000	0					
	..		.			
00001000	8	удаление последнего символа (клавиша {Backspace})				
	..		..			
00001101	13	перевод строки (клавиша {Enter})				
..			...			
00100000	32	клавиша {Пробел}				
00100001	33	!				
..	...					
01011010	90	Z				
..	..		.			
10000000	128	—	Ъ	А	А	к
...	.		.	.	..	...
11000010	194	б	В	—	—	Т
..	.	..	.	.	.	..
11001100	204	л	М	|	|	Ь
	.	.	.		..	
11011101	221	щ	Э	_	Е	н
..	...	..	..		.	
11111111	255	ь	я	неразде-ляемый пробел	неразде-ляемый пробел	п

В процессе вывода знака на экран компьютера производится обратное кодирование, т. е. преобразование двоичного кода знака в его изображение.

Различные кодировки знаков. Присвоение знаку конкретного двоичного кода — это вопрос соглашения, которое фиксируется в кодовой таблице. Первые 33 кода в кодовой таблице (десятичные коды с 0 по 32) соответствуют не знакам, а операциям (перевод строки, ввод пробела и т. д.).

Десятичные коды с 33 по 127 являются интернациональными и соответствуют знакам латинского алфавита, цифрам, знакам арифметических операций и знакам препинания.

Десятичные коды с 128 по 255 являются национальными, т. е. в различных национальных кодировках одному и тому же коду соответствуют разные знаки. 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.

Кодовые таблицы Windows-CD

В последние годы широкое распространение получил новый международный стандарт кодирования текстовых символов *Unicode*, который отводит на каждый символ 2 байта (16 битов). По формуле (1.1) определим 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.

Итак, в настоящее время имеется шесть различных кодировок для букв русского алфавита, в которых один и тот же знак имеет различные коды (табл. 2.2). К счастью, в большинстве случаев пользователь не должен заботиться о перекодировках текстовых документов, так как это делают специальные программы-конверторы, встроенные в операционную систему и приложения.

Таблица 2.2. **Десятичные коды некоторых знаков в различных кодировках**

Символ	Windows	MS-DOS	КОИ-8	Mac	ISO	Unicode
А	192	128	225	128	176	1040
В	194	130	247	130	178	1042
М	204	140	237	140	188	1052
Э	221	157	252	157	205	1069
я	255	239	241	223	239	1103

Например, в кодировке *Windows* последовательность числовых кодов 221 194 204 образует слово «ЭВМ» (см. табл. 2.2), тогда как в других кодировках это будет бессмысленный набор символов.

Контрольные вопросы

1. Почему при кодировании текстовой информации в компьютере в большинстве кодировок используется 256 различных символов, хотя русский алфавит включает только 33 буквы?
2. С какой целью ввели кодировку *Unicode*, которая позволяет закодировать 65 536 различных символов?

Задания для самостоятельного выполнения

2.1. *Задание с кратким ответом.* В текстовом режиме экран монитора компьютера обычно разбивается на 25 строк по 80 символов в строке. Определить объем текстовой информации, занимающей весь экран монитора, в кодировке *Unicode*.

2.2. *Задание с развернутым ответом.* Пользователь компьютера, хорошо владеющий навыками ввода информации с клавиатуры, может вводить в минуту 100 знаков. Какое количество информации может ввести пользователь в компьютер за одну минуту в кодировке *Windows*? Кодировке *Unicode*?

2.2. Создание документов в текстовых редакторах

Текстовые редакторы. Для обработки текстовой информации на компьютере используются приложения общего назначения — текстовые редакторы. Текстовые редакторы позволяют создавать, редактировать, форматировать, сохранять и распечатывать документы.

Простые текстовые редакторы позволяют редактировать текст, а также осуществлять простейшее форматирование шрифта.

Более совершенные текстовые редакторы (их называют также текстовыми процессорами) имеют широкий спектр возможностей по созданию документов (вставка списков и таблиц, средства проверки орфографии, сохранение исправлений и др.).

Для подготовки к изданию книг, журналов и газет в процессе макетирования издания используются мощные программы обработки текста — настольные издательские системы.

Для подготовки к публикации в Интернете Web-страниц и Web-сайтов используются Web-редакторы.

Способы создания документов. В текстовых редакторах для создания многих типов документов со сложной структурой (письма, резюме, факсы и т. д.) используются **Мастера**. Разработка документа с помощью Мастера производится путем внесения необходимых данных в последовательно появляющиеся диалоговые окна. Например, можно использовать Мастер для создания календаря, который должен разместить на странице в определенном порядке обязательный набор надписей (год, месяц, дату и др.)

Создание документов можно производить с помощью **шаблонов**, т. е. пустых заготовок документов определенного назначения. Шаблон задает структуру документов, которую пользователь заполняет определенным содержанием. Текстовые процессоры имеют обширные библиотеки шаблонов для создания документов различного назначения (визитная карточка, реферат и др.).

Однако в большинстве случаев для создания документов используется пустой шаблон *Новый документ*, который пользователь заполняет содержанием по своему усмотрению.

Выбор параметров страницы (рис. 2.1). Любой документ состоит из страниц, поэтому в начале работы над документом необходимо задать параметры страницы: **формат**, **ориентацию** и **размер полей**.

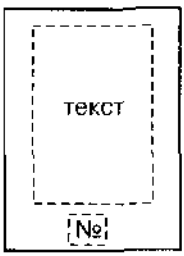
Формат страниц документа определяет их размер. При создании реферата или заявления целесообразно выбрать формат страницы А4 (21 × 29,7 см), который соответствует размеру стандартного листа бумаги для принтера. Для объявлений и плакатов подходит формат А3, размер листа этого формата в два раза больше размера стандартного листа. Наоборот, для писем можно выбрать формат А5, который в два раза меньше стандартного.

Ориентация позволяет выбрать расположение страницы на экране монитора. Существуют две возможные ориентации страницы — *книжная* и *альбомная*. Для обычных текстов чаще всего используется книжная ориентация, а для таблиц с большим количеством столбцов — альбомная.

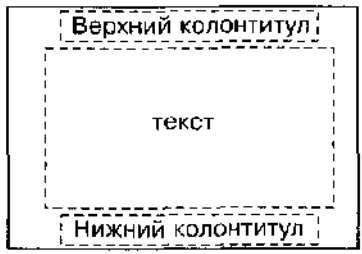
На странице можно установить требуемые **размеры полей** (*верхнего* и *нижнего*, *правого* и *левого*), которые определяют расстояния от краев страницы до границ текста.

Колонтитулы и номера страниц. Для вывода на каждой странице документа одинакового текста (например, имени автора, названия документа и др.) удобно использовать *верхний* или *нижний колонтитулы*. Расстояния от краев страницы до колонтитула можно изменять.

Страницы документа рекомендуется нумеровать, причем номера можно размещать вверху или внизу страницы по центру, справа или слева.



Формат А4,
книжная ориентация,
номер страницы



Формат А3,
альбомная ориентация,
верхний и нижний колонтитулы

Рис. 2.1. Параметры страницы: формат, ориентация, поля, колонтитулы, номер страницы

Контрольные вопросы

1. Какие существуют способы создания новых документов?
2. Какие параметры страниц необходимо задать перед началом создания документа?

2.3. Ввод и редактирование документа

Ввод текста. Основой большинства документов является текст, т. е. последовательность различных символов: прописных и строчных букв русского и латинского алфавитов, цифр, знаков препинания, математических символов и др. Для быстрого ввода текстов целесообразно научиться (например, с использованием клавиатурного тренажера) десятипальцевому «слепому» методу ввода символов.

Для представления текстов могут использоваться 256 символов или 65 536 символов, однако невозможно ввести все эти символы с клавиатуры компьютера. Для ввода некоторых знаков математических операций, букв греческого алфавита, денежных знаков и многих других символов используются таблицы символов.

Вставка изображений, формул и других объектов в документ. Большинство современных документов содержат не только текст, но и другие объекты (изображения, формулы, таблицы, диаграммы и т. д.). Текстовые редакторы позволяют вставлять в документ изображения, созданные в графических редакторах, таблицы и диаграммы, созданные в электронных таблицах, и даже звуковые и видеофайлы, созданные в соответствующих приложениях.

При решении задач по физике или математике часто необходимо вставлять формулы, которые требуют двухстрочного представления и использования специальных математических знаков. Для ввода формул в текстовые редакторы встроены специальные редакторы формул (рис. 2.2).

$$I = \frac{U}{R}$$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Рис. 2.2. Формулы закона Ома и корней квадратного уравнения, введенные с помощью редактора формул

Копирование, перемещение и удаление фрагментов документа. Редактирование документа производится путем копирования, перемещения или удаления выделенных символов или фрагментов документа. Выделение производится с помощью мыши или клавиш управления курсором на клавиатуре при нажатой клавише ⟨*Shift*⟩.

Копирование позволяет размножить выделенный фрагмент документа, т. е. вставить его копии в указанные места документа:

- *после выделения фрагмента документа и ввода команды Копировать* выделенная часть документа помещается в *буфер обмена* (специальную область памяти);
- с помощью мыши или клавиш управления курсором на клавиатуре курсор устанавливается в определенное место документа, затем вводится команда *Вставить*. Копируемый фрагмент документа, хранящийся в *буфере обмена*, помещается в указанное место;

- для многократного копирования фрагмента достаточно несколько раз повторить команду *Вставить*.

Перемещение позволяет вставить копии выделенного фрагмента документа в указанные места документа, но удаляет сам выделенный фрагмент.

Удаление позволяет удалить выделенный фрагмент.

Например, если исходный документ содержит слово «информатика», то после операций копирования, перемещения и удаления фрагмента «форма» документ примет вид, отображенный в табл. 2.3.

Таблица 2.3. **Операции редактирования текстового документа**

Состояние документа	Операция редактирования		
	Копирование	Перемещение	Удаление
до редактирования	информатика	информатика	информатика
после редактирования	информатика форма	интика форма	интика

Поиск и замена. В процессе работы над документом иногда бывает необходимо заменить одно многократно использованное слово на другое. Если делать это вручную, то процесс замены отнимет много времени и сил.

К счастью, в большинстве текстовых редакторов существует операция *Найти и заменить*, которая обеспечивает автоматический поиск и замену слов во всем документе (например, замену слова «ЭВМ» на слово «компьютер» — рис. 2.3).

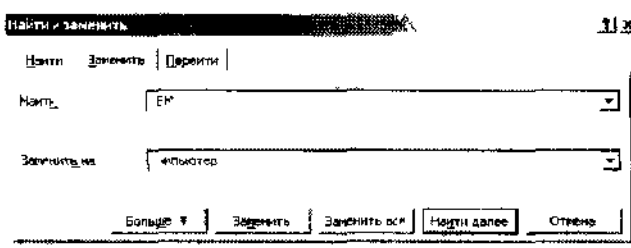


Рис. 2.3. Поиск и замена слов в документе

Проверка правописания. В процессе создания документа могут быть допущены орфографические ошибки в написании слов и синтаксические ошибки в построении предложений.

Ошибки можно исправить, если запустить встроенную во многие текстовые редакторы систему проверки правописания, которая содержит орфографические словари и грамматические правила для нескольких языков (это позволяет исправлять ошибки в многоязычных документах). Система проверки правописания не только выделяет орфографические ошибки (красной волнистой линией) и синтаксические ошибки (зеленой волнистой линией), но и предлагает варианты их исправления (рис. 2.4).

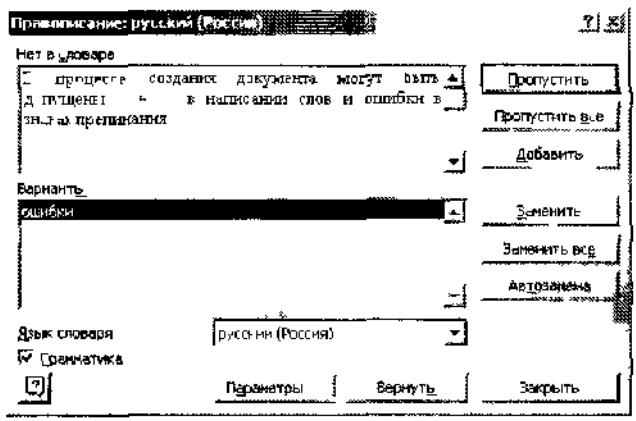


Рис. 2.4. Проверка правописания в документе

Проверку правописания текстовые редакторы могут проводить как непосредственно в процессе ввода текста, так и в готовом документе по команде пользователя.

Автозамена частых опечаток. В процессе ввода текста иногда допускаются опечатки (например, в начале слова случайно вводятся ДВе прописные буквы). В этом случае срабатывает функция *Автозамена*, которая автоматически исправляет такие опечатки.

Кроме того, каждый пользователь может добавить в *словарь автозамены* те слова, в которых он часто делает ошибки (например, неправильное «програма» должно заменяться на правильное «программа») (рис. 2.5).

Сохранение исправлений. В работе над документом могут участвовать несколько пользователей. Исправления, вносимые каждым из них, запоминаются и могут быть просмотрены и распечатаны (вставленные фрагменты обычно

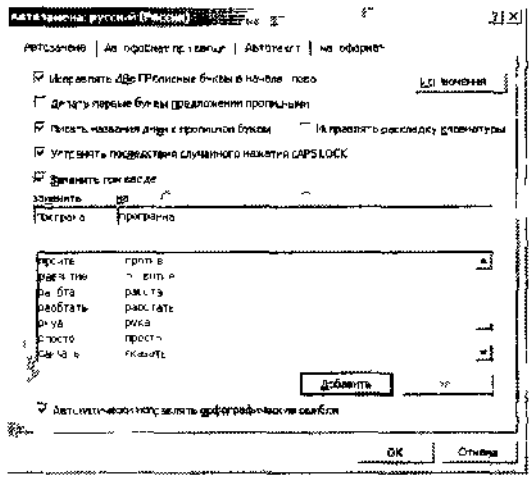


Рис. 2.5. Настройка параметров автозамены

отображаются подчеркнутым шрифтом синего цвета, а удаленные фрагменты текста — зачеркнутым шрифтом красного цвета — рис. 2.6).

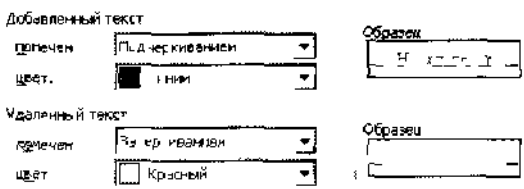


Рис. 2.6. Настройка выделения исправлений

В процессе работы над окончательной редакцией документа может быть проведено сравнение исправлений, сделанных различными авторами, и принят лучший вариант.

Контрольные вопросы

1. Какие существуют способы ввода содержания документов?
2. Какие существуют способы редактирования документов?

Задания для самостоятельного выполнения

2.3. *Практическое задание.* В текстовом редакторе создать документ и ввести последовательность символов: ≥, ≤, ±, ≈, ∞, α, π, ©.

2.4. *Практическое задание.* В текстовом редакторе создать документ, содержащий слово «килобайт», и провести копирование, перемещение и удаление приставки «кило».

2.5. *Практическое задание.* В текстовом редакторе осуществить проверку правописания в сочинении по литературе и в упражнении по иностранному языку.

2.6. *Практическое задание.* В текстовом редакторе установить автозамену тех слов, в которых вы чаще всего допускаете опечатки.

2.4. Сохранение и печать документов

Сохранение документов. В процессе сохранения документа необходимо прежде всего в иерархической файловой системе компьютера выбрать диск и папку, в которой файл документа необходимо сохранить.

Кроме того, необходимо выбрать формат файла, который определяет способ хранения текста в файле. Существуют универсальные форматы текстовых файлов, которые могут быть прочитаны большинством текстовых редакторов, и оригинальные форматы, которые используются только определенными текстовыми редакторами.

Формат *Только текст* (расширение в имени файла txt) является наиболее универсальным текстовым форматом. Файлы, сохраненные в этом формате, могут быть прочитаны приложениями, работающими в различных операционных системах. Достоинством этого формата является небольшой информационный объем файлов, а недостатком то, что не сохраняются результаты форматирования текста.

Расширенный текстовый формат (расширение в имени файла rtf) является также универсальным форматом текстовых файлов, который сохраняет результаты форматирования. Недостатком этого формата является большой информационный объем файлов.

Формат *Документ Word* (расширение в имени файла doc) является оригинальным форматом текстового редактора Microsoft Word, полностью сохраняющим форматирование. Этот формат фактически является универсальным, так как понимается практически всеми текстовыми редакторами. В последней версии Microsoft Office используется формат DOCX, для перевода в формат DOC существуют конверторы.

В интегрированном офисном приложении OpenOffice используется открытый формат документов для офисных приложений *OpenDocument Format* (ODF), в том числе текстовых документов (ODT), электронных таблиц (ODS), рисунков (ODG) и презентаций (ODP).

Формат *Web-страница* (расширение в имени файла htm или html) используется для хранения Web-страниц в компьютерных сетях, так как файлы имеют небольшой информационный объем, но сохраняется форматирование. Документы в формате *Web-страница* создаются в Web-редакторах, а также могут сохраняться в этом формате многими текстовыми редакторами. Достоинством формата является его универсальность, так как Web-страницы могут просматриваться с использованием специализированных программ (браузеров) в любых операционных системах.

Современные текстовые редакторы обеспечивают автоматическое, указанное пользователем, преобразование текстового файла из одного формата в другой при его открытии и сохранении.

Печать документа. Перед выводом документа на печать полезно выполнить предварительный просмотр документа, это позволяет увидеть, как будет выглядеть документ, напечатанный на бумаге с использованием подключенного к компьютеру принтера.

Вид напечатанного документа (например, распределение текста по страницам) может зависеть от используемого принтера, так как могут несколько отличаться шрифты, используемые в разных принтерах.

При выводе документа на печать необходимо установить параметры печати: задать номера выводимых на печать страниц, количество копий документа и др.

Кроме того, целесообразно проверить установки самого принтера: ориентацию бумаги, качество бумаги, качество печати и др.

Контрольные вопросы

1. Какие существуют форматы текстовых файлов и чем они отличаются друг от друга?
2. В каком формате нужно сохранить файл, чтобы он мог быть прочитан в других приложениях с сохранением форматирования? Без сохранения форматирования?
3. Какие параметры необходимо установить перед началом печати документа?
4. Влияет ли на вид напечатанного документа выбор принтера? Почему?

Задания для самостоятельного выполнения

2.7. *Практическое задание.* В текстовом редакторе открыть текстовый файл, содержащий форматирование, и сохранить файл в различных текстовых форматах. Сравнить вид и информационные объемы документов, сохраненных в различных форматах.

2.5. Форматирование документа

2.5.1. Форматирование символов

Для представления содержания документа в понятной и выразительной форме применяется форматирование. Символы являются основными объектами, из которых состоит текстовый документ, и поэтому прежде всего необходимо правильно установить основные параметры, определяющие их внешний вид: **шрифт**, **размер**, **начертание** и **цвет**.

Шрифт. Шрифт — это полный набор символов (букв, цифр, знаков пунктуации, математических знаков, а также специальных символов) определенного рисунка. Для каждого исторического периода и каждой страны характерны свои шрифты. Шрифты имеют названия, например: Times New Roman, Arial, Courier New и др. (табл. 2.4).

По способу представления в компьютере различаются шрифты **растровые и векторные.** Для представления растровых шрифтов используются методы растровой графики, когда символы шрифта представляют собой группы пикселей. Растровые шрифты допускают масштабирование только с определенными коэффициентами (например, MS Sans Serif 8, 10, 12 и т. д.). В векторных шрифтах символы описываются математическими формулами и допускают произвольное масштабирование.

Обычно различные символы шрифта имеют и различную ширину, например, буква Ш шире, чем буква А. Однако имеются и **моноширинные шрифты**, в которых ширина всех символов одинакова. Примером такого шрифта является шрифт Courier New.

Шрифты также разделяют на **шрифты с засечками** (например, Times New Roman) и **рубленые** (например, Arial). Считается, что шрифты с засечками легче воспринимаются глазом, и поэтому в большинстве печатных текстов используются именно они. Рубленые шрифты используют обычно для заголовков, выделений в тексте и подписей к рисункам.

Таблица 2.4. **Примеры шрифтов различного типа**

Шрифт	Вид шрифта
Times New Roman	информатика
Arial	информатика
Courier New	информатика

Размер шрифта. Единицей измерения размера шрифта является *пункт* (1 пункт (пт) = 0,376 мм). Размеры шрифтов можно изменять в больших пределах (обычно от 1 до 1638 пунктов), причем в большинстве редакторов по умолчанию используется шрифт размером 10 пт. Ниже приведены примеры представления текста с помощью шрифта различного размера:

Шрифт размером 16 пт.
Шрифт размером 12 пт.
Шрифт размером 8 пт.

Начертание и вид символов (табл. 2.5). Кроме обычного начертания символов может применяться **полужирное**, *курсивное* и ***полужирное курсивное***.

Можно установить дополнительные параметры форматирования символов: *подчеркивание* символов различными типами линий, изменение вида символов (*верхний индекс*, *нижний индекс*, *зачеркнутый*), изменение расстояния между символами (*разреженный*, *уплотненный*) и др.

Таблица. 2.5. **Дополнительные параметры форматирования символов**

Параметр форматирования	Внешний вид символов
Тип линии подчеркивания	сплошная, пунктирная, волнистая
Вид символов	верхний индекс, нижний индекс, ~~зачеркнутый~~
Расстояние между символами	разреженный, уплотненный

Цвет символов. Если планируется многоцветная печать документа, то для различных групп символов можно задать различные цвета, выбранные из предлагаемой текстовым редактором палитры.

Контрольные вопросы

1. Какие параметры определяют внешний вид символов?
2. Какие существуют типы шрифтов?

Задания для самостоятельного выполнения

2.8. *Задание с кратким ответом.* Какое начертание имеют символы текста: ***текст***?

2.5.2. Форматирование абзацев

Абзац выделяет в текстовом документе часть текста, представляющую законченный по смыслу фрагмент документа, окончание которого служит естественной паузой для перехода к новой мысли. В компьютерных документах абзац

заканчивается управляющим знаком конца абзаца. Ввод конца абзаца обеспечивается нажатием клавиши {*Enter*} и отображается символом ¶, если включен режим отображения непечатаемых символов.

Абзац может состоять из любого набора символов, рисунков и объектов других приложений. Форматирование абзацев позволяет подготовить правильно и красиво оформленный документ.

Выравнивание абзацев. Выравнивание отражает расположение текста относительно границ полей страницы. Чаще всего используют четыре способа выравнивания абзацев (рис. 2.7).

По левому краю — *левый* край *абзаца ровный*, а правый край — рваный.

По центру — оба края имеют неровные очертания, каждая строка абзаца симметрична по горизонтали относительно середины страницы.

По правому краю — правый край ровный, а левый — рваный.

По ширине — оба края ровные, т. е. располагаются точно по границам полей страницы. В этом случае последняя строка абзаца ведет себя, как при выравнивании по левому краю.

Рис. 2.7. Выравнивание абзацев

Отступ первой строки (красная строка). Чаще всего абзац начинается отступом первой строки. Существуют отступы различных типов (рис. 2.8).

Положительный (*отступ*) — первая строка абзаца начинается правее всех остальных строк абзаца, применяется в обычном тексте.

Отрицательный (*выступ*) — первая строка выходит влево относительно остальных строк, применяется в словарях и определениях.

Нулевой — применяется для абзацев, выровненных по центру, и для обычного текста.

Рис. 2.8. Отступы первой строки

Отступы и интервалы. Весь абзац целиком может иметь отступы слева и справа, которые отмеряются от границ полей страницы (рис. 2.9). Так, эпиграф к художественному произведению или реквизиты адресата в заявлении имеют отступ слева, а при изготовлении углового штампа можно использовать отступ справа.

Отступ абзаца слева — начало всех строк абзаца смещено на заданное расстояние вправо.

Отступ абзаца справа — конец всех строк абзаца смещен на заданное расстояние влево.

Рис. 2.9. Отступы абзаца

Расстояние между строками документа можно изменять, задавая различные значения междустрочных интервалов (*одинарный*, *двойной* и т. д.). Для визуального отделения абзацев друг от друга можно устанавливать увеличенные интервалы *до* и *после* абзацев.

Контрольные вопросы

1. Каковы основные параметры форматирования абзацев?
2. В чем состоит различие между отступом первой строки абзаца и отступом абзаца?
3. В чем состоит различие между междустрочными интервалами и интервалами между абзацами?

Задания для самостоятельного выполнения

2.9. *Задание с выборочным ответом.* Абзацем в текстовом редакторе является:

1) фрагмент документа между двумя маркерами абзаца;
2) выделенный фрагмент документа;
3) строка символов;
4) фрагмент документа, начинающийся с отступа (красной строки).

2.5.3. Нумерованные и маркированные списки

Списки являются удобным вариантом форматирования абзацев по единому образцу и применяются для размещения в документе различных перечней.

Нумерованные списки. В нумерованных списках элементы списка последовательно обозначаются с помощью чисел (арабских или римских) и букв (русского или латинского алфавитов) (рис. 2.10). При создании, удалении или перемещении элементов нумерованного списка автоматически меняется вся нумерация. Пользователь может установить свою систему нумерации, например начать список с любого номера, пропустить номер и т. д.

Пользователь может установить удобный формат номеров (*размер и начертание шрифта, отступ номера от поля страницы, расстояние от номера до текста* и т. д.).

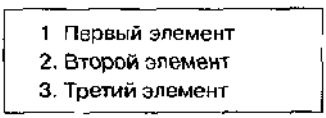


Рис. 2.10. Нумерованный список

Маркированные списки. В маркированных списках *элементы списка обозначаются с помощью маркеров (специальных значков)*: •, ▪, ⇨ и т. д. (рис. 2.11). Пользователь может выбрать тип маркера, изменить его размер и цвет, а также выбрать в качестве маркера любой символ из таблицы символов.

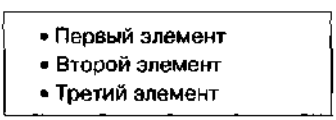


Рис. 2.11. Маркированный список

Многоуровневые списки. Многоуровневые списки удобно использовать для отображения иерархических перечней. В многоуровневых списках в пункты списка более высокого уровня вставляются списки более низкого уровня (вложенные списки). Вложенные списки могут совпадать по типу с основным списком, но могут и отличаться от него.

В качестве примера рассмотрим следующий многоуровневый список (рис. 2.12). На первом уровне находится нумерованный список из двух элементов, в каждый из которых вложен нумерованный список из двух элементов второго уровня, во второй элемент которого вложен маркированный список из одного элемента третьего уровня.

1. Первый уровень, первый элемент
   a. Второй уровень, первый элемент
   a. Второй уровень, второй элемент
      - третий уровень
2. Первый уровень, второй элемент
   b. Второй уровень, первый элемент
   b. Второй уровень, второй элемент
      - третий уровень

Рис. 2.12. Многоуровневый список

Контрольные вопросы

1. В чем состоит различие между нумерованными и маркированными списками?
2. Может ли многоуровневый список включать как нумерованные, так и маркированные списки?

Задания для самостоятельного выполнения

2.10. *Задание с выборочным ответом.* В маркированном списке для обозначения элемента списка используются:
1) латинские буквы; 3) римские цифры;
2) русские буквы; 4) графические значки.

2.6. Таблицы

Таблицы используются при создании текстовых документов, содержащих большое количество однотипных названий (например, расписание уроков), числовых данных (например, таблица Менделеева), или изображений с текстовой подписью (например, алфавит в букваре).

Строки, столбцы, ячейки. Таблицы состоят из **строк** и **столбцов**, на пересечении которых образуются **ячейки**. В ячейках таблиц могут быть размещены различные типы данных (текст, числа, изображения и пр.) (табл. 2.6).

Таблица 2.6 **Таблица, содержащая текст, изображения, числа и формулу**

Наименование устройства	Изображение	Цена, руб.
Системный блок		15000
Монитор		12000
Клавиатура		1000
Мышь		200
Итого:		**28200**

Создание и изменение таблицы. В документ можно вставить пустую таблицу, указав необходимое количество строк и столбцов, а также их, соответственно, высоту и ширину. В таблицу можно преобразовать уже имеющийся текст, при этом требуется указать разделитель текста (например, знак окончания абзаца), который позволит текстовому редактору автоматически распределить выделенный текст по ячейкам создаваемой таблицы.

В дальнейшем параметры таблицы можно модернизировать:

- вставлять или удалять строки, столбцы и ячейки;
- изменять ширину столбцов и высоту строк с помощью мыши (перетаскиванием границ) или заданием их точного значения в сантиметрах или процентах;

- изменять размеры отдельных ячеек, разделять их на несколько ячеек или объединять с соседними ячейками.

Границы и заливка. Можно подобрать подходящий внешний вид таблицы, изменив тип, ширину и цвет границ ячеек, а также цвет фона ячеек. Изменение внешнего вида таблицы можно провести автоматически, используя готовые форматы, или настроить вручную. Так в таблице 2.6 используется двойная линия в качестве внешней границы таблицы и заливка серого цвета в первой строке.

Вычисления в таблице. При размещении в таблице чисел можно производить над ними вычисления по формулам. Так, в таблице 2.6 содержится информация обо всех устройствах, из которых состоит компьютер, и поэтому можно вычислить его стоимость. Для этого в последнюю ячейку третьего столбца необходимо ввести формулу =SUM(ABOVE), которая обеспечивает суммирование чисел во всех вышерасположенных ячейках данного столбца таблицы.

Таблицы широко используются в Интернет-магазинах для представления информации о товарах. Действительно, в этом случае документ должен содержать большое количество строк, включающих текст (наименование товара), изображение (фотография товара) и число (цена товара).

Контрольные вопросы

1. Можно ли из таблицы удалить столбец? Строку? Ячейку?
2. Данные каких типов могут храниться в ячейках таблицы?

Задания для самостоятельного выполнения

2.11. *Практическое задание.* В текстовом редакторе создать документ, содержащий расписание уроков. Применить различные варианты форматирования таблицы (шрифт, выравнивание, границы и фон ячеек).

2.7. Компьютерные словари и системы машинного перевода текстов

Компьютерные словари. Словари необходимы для перевода текстов с одного языка на другой. Существуют тысячи словарей для перевода между сотнями языков (англо-русский, немецко-французский и т. д.), причем каждый из них может содержать десятки тысяч слов. В бумажном варианте словарь представляет собой толстую книгу объемом в сотни страниц, в которой поиск нужного слова является достаточно долгим и трудоемким процессом.

Компьютерные словари содержат переводы на разные языки сотен тысяч слов и словосочетаний, а также предоставляют пользователю дополнительные возможности.

Во-первых, компьютерные словари могут быть многоязычными, позволяющими пользователю выбрать языки и направление перевода (например, англо-русский, испанско-русский и т. д.).

Во-вторых, компьютерные словари могут кроме основного словаря общеупотребительных слов содержать десятки специализированных словарей по областям знаний (техника, медицина, информатика и др.).

В-третьих, компьютерные словари обеспечивают быстрый поиск словарных статей: «быстрый набор», когда в процессе набора слова возникает список похожих слов; доступ к часто используемым словам по закладкам; возможность ввода словосочетаний и т. д.

В-четвертых, компьютерные словари могут быть мультимедийными, т. е. предоставлять пользователю возможность прослушивания слов в исполнении дикторов, носителей языка.

Системы компьютерного перевода. Процесс глобализации мира приводит к необходимости частого обмена документами между людьми и организациями, находящимися в разных странах мира и говорящими на различных языках.

В этих условиях использование традиционной технологии перевода вручную тормозит развитие межнациональных контактов. Перевод многостраничной документации вручную требует длительного времени и высокой оплаты труда переводчиков. Перевод полученного по электронной почте письма или просматриваемой в браузере Web-страницы необходимо осуществить «здесь и сейчас», когда нет возможности и времени пригласить переводчика.

Системы компьютерного перевода позволяют решить эти проблемы. Они способны переводить многостраничные документы с высокой скоростью (страница в секунду), а также переводить Web-страницы «на лету», в режиме реального времени.

Системы компьютерного перевода осуществляют перевод текстов, основываясь на формальном «знании» языка: синтаксиса языка (правил построения предложений), правил словообразования и использовании словарей. Программа-переводчик сначала анализирует текст на одном языке, а затем конструирует этот текст на другом языке.

Современные системы компьютерного перевода позволяют достаточно качественно переводить техническую документацию, деловую переписку и другие специализированные тексты. Однако они неприменимы для перевода художественных произведений, так как не способны адекватно переводить метафоры, аллегории и другие элементы художественного творчества человека.

Контрольные вопросы

1. Какими преимуществами обладают компьютерные словари перед традиционными бумажными словарями?
2. В каких случаях целесообразно использовать системы компьютерного перевода?

Задания для самостоятельного выполнения

2.12. *Практическое задание.* С помощью компьютерного словаря перевести пять англоязычных терминов, использующихся в учебнике.

2.8. Системы оптического распознавания документов

Системы оптического распознавания символов. Системы оптического распознавания символов используются при создании электронных библиотек и архивов путем перевода книг и документов в цифровой компьютерный формат.

Сначала с помощью сканера необходимо получить изображение страницы текста в графическом формате. Далее для получения документа в текстовом формате необходимо провести распознавание текста, т. е. преобразовать элементы графического изображения в последовательность текстовых символов.

Системы оптического распознавания символов сначала определяют структуру размещения текста на странице и разбивают его на отдельные области: колонки, таблицы, изображения и т. д. Далее выделенные текстовые фрагменты графического изображения страницы разделяются на изображения отдельных символов.

Для отсканированных документов типографского качества (достаточно крупный шрифт, отсутствие плохо напечатанных символов или исправлений) распознавание символов проводится путем их сравнения с растровыми шаблонами.

Растровое изображение каждого символа последовательно накладывается на растровые шаблоны символов, хранящиеся в памяти системы оптического распознавания. Результатом распознавания является символ, шаблон которого в наибольшей степени совпадает с изображением (рис. 2.13).

Рис. 2.13. Распознаваемый символ «Б» накладывается на растровые шаблоны символов (А, Б, В и т. д.)

При распознавании документов с низким качеством печати (машинописный текст, факс и т. д.) используется векторный метод распознавания символов. В распознаваемом изображении символа выделяются геометрические примитивы (отрезки, окружности и др.) и сравниваются с векторными шаблонами символов. В результате выбирается тот символ, для которого совокупность всех геометрических примитивов и их расположение больше всего соответствуют распознаваемому символу (рис. 2.14).

Системы оптического распознавания символов используют как растровый, так и векторный метод распознавания.

Рис. 2.14. Распознаваемый символ «Б» накладывается на векторные шаблоны символов (А, Б, В и т. д.)

Кроме того, эти системы являются «самообучающимися» (для каждого конкретного документа они создают соответствующий набор шаблонов символов), и поэтому скорость и качество распознавания многостраничного документа постепенно возрастают.

С появлением первого карманного компьютера Newton фирмы Apple в 1990 году начали создаваться системы распознавания рукописного текста. Такие системы преобразуют текст, написанный на экране карманного компьютера специальной ручкой, в текстовый компьютерный документ.

Системы оптического распознавания форм. При заполнении документов большим количеством людей (например, при сдаче ЕГЭ выпускниками школы) используются бланки с пустыми полями. Данные вводятся в поля печатными буквами от руки, а затем распознаются с помощью систем оптического распознавания форм и вносятся в компьютерные базы данных.

Сложность состоит в том, что необходимо распознавать символы, написанные от руки, которые довольно сильно различаются у разных людей. Кроме того, такие системы должны уметь определять, к какому полю относится распознаваемый текст.

Контрольные вопросы

1. В чем состоят различия в технологии распознавания текста при использовании растрового и векторного методов?

Практические работы компьютерного практикума, рекомендуемые для выполнения в процессе изучения главы 2

Компьютерный практикум

2.1. Кодирование текстовой информации.

2.2. Вставка в документ формул.

2.3. Форматирование символов и абзацев

2.4. Создание и форматирование списков.

2.5. Вставка в документ таблицы, ее форматирование и заполнение данными.

2.6. Перевод текста с помощью компьютерного словаря.

2.7. Сканирование и распознавание бумажного текстового документа

Глава 3

Кодирование и обработка числовой информации

3.1. Кодирование числовой информации

3.1.1. Представление числовой информации с помощью систем счисления

Для записи информации о количестве объектов используются числа. Числа записываются с использованием особых знаковых систем, которые называются **системами счисления**. Алфавит системы счисления состоит из знаков, которые называются цифрами.

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

Все системы счисления делятся на две большие группы: **позиционные** и **непозиционные** системы. В позиционных системах счисления количественное значение цифры зависит от ее положения в числе, а в непозиционных — не зависит.

Непозиционные системы 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: зарубок, черточек, точек.

Такая система записи чисел называется **единичной**, так как любое число в ней образуется путем повторения одного знака, символизирующего единицу (рис. 3.1). Единичной системой счисления пользуются малыши, показывая на пальцах свой возраст или используя для этого счетные палочки.

Рис. 3.1. Единичная система счисления

Примером непозиционной системы, которая сохранилась до наших дней, может служить **римская система счисления**, которая начала применяться более двух с половиной тысяч лет назад в Древнем Риме.

В основе римской системы счисления лежат 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 используются латинские буквы C, D и M (рис. 3.2).

1	I	11	XI	30	XXX	400	CD
2	II	12	XII	40	XL	500	D
3	III	13	XIII	50	L	600	DC
4	IV	14	XIV	60	LX	700	DCC
5	V	15	XV	70	LXX	800	DCCC
6	VI	16	XVI	80	LXXX	900	CM
7	VII	17	XVII	90	XC	1000	M
8	VIII	18	XVIII	100	C	2000	MM
9	IX	19	XIX	200	CC	3000	MMM
10	X	20	XX	300	CCC	4000	MMMM

Рис. 3.2. Римская система счисления

В римской системе счисления значение цифры не зависит от ее положения в числе. Например, в римском числе XXX (30) цифра X встречается трижды и в каждом случае обозначает одну и ту же величину — число 10, три раза по 10 в сумме дают 30.

Чтобы записать число в римской системе счисления, необходимо разложить его на сумму тысяч, полутысяч, сотен, полусотен, десятков, пятков, единиц. Например, десятичное число 28 представляется следующим образом:

10 + 10 + 5 + 1 + 1 + 1 = XXVIII
(два десятка, пяток, три единицы).

При записи чисел в римской системе счисления применяется правило: каждый меньший знак, поставленный справа от большего, прибавляется к большему знаку, а каждый меньший знак, поставленный слева от большего, вычитается из большего знака.
Например, римское число IX обозначает 9 (–1 + 10), а XI обозначает 11 (10 + 1). Например, число 99 имеет следующее представление в римской системе счисления:

XCIX = –10 + 100 – 1 + 10

Позиционные системы счисления. Каждая позиционная система счисления имеет определенный алфавит цифр и основание. **Основание системы** равно количеству цифр (знаков) в ее алфавите.

В позиционных системах счисления количественное значение цифры зависит от ее позиции в числе. Позиция цифры в числе называется **разрядом**. Разряды числа возрастают справа налево, от младших разрядов к старшим, причем значения цифр в соседних разрядах числа различаются в количество раз, равное основанию системы.

В настоящее время наиболее распространенной позиционной системой счисления является десятичная система. В информатике широко используются двоичная, восьмеричная и шестнадцатеричная системы счисления.

Десятичная система счисления (табл. 3.1). В десятичной системе счисления крайняя справа позиция соответствует минимальному значению, в которой цифра обозначает единицы, цифра, смещенная на одну позицию влево, обозначает десятки, еще левее — сотни, затем тысячи и т. д. Рассмотрим в качестве примера десятичное число 555. Цифра 5 встречается в числе трижды, причем самая правая обозначает пять единиц, вторая справа — пять десятков и, наконец, третья — пять сотен.

Число в позиционной системе счисления записывается **в виде суммы числового ряда степеней основания**, в качестве коэффициентов которых выступают цифры данного числа.

Выше десятичное число 555 было записано в привычной для нас **свернутой форме**. Мы настолько привыкли к такой форме записи, что уже не замечаем, как в уме умножаем цифры числа на различные степени числа 10, которое является основанием десятичной системы счисления.

В **развернутой форме** записи числа умножение цифр числа на основание производится в явной форме. Так, в развернутой форме запись числа 555 в десятичной системе будет выглядеть следующим образом:

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

Для записи десятичных дробей используются разряды с отрицательными значениями степеней основания. Например, число 555,55 в развернутой форме будет записано 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:

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

Двоичная система счисления (см. табл. 3.1). Числа в двоичной системе в развернутой форме записываются в виде суммы ряда степеней основания 2 с коэффициентами, в качестве которых выступают цифры 0 или 1.

Например, развернутая запись двоичного числа выглядит следующим образом:

$$A_2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2},$$

а в свернутой форме:

$$A_2 = 101{,}01_2.$$

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:

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

Восьмеричная система счисления (см. табл. 3.1). В восьмеричной системе основание равно 8 и алфавит состоит из восьми цифр {0, 1, 2, 3, 4, 5, 6, 7}. Запишем восьмеричное число в свернутой и развернутой форме:

$$77_8 = 7 \cdot 8^1 + 7 \cdot 8^0.$$

Шестнадцатеричная система счисления (см. табл. 3.1). В шестнадцатеричной системе основание равно 16 и алфавит состоит из шестнадцати цифр {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}, причем первые десять цифр имеют об-

щепринятое обозначение, а для записи остальных цифр {10, 11, 12, 13, 14, 15} используются первые шесть букв латинского алфавита. Запишем шестнадцатеричное число в свернутой и развернутой формах:

$$ABCDEF_{16} = A\cdot16^5 + B\cdot16^4 + C\cdot16^3 + D\cdot16^2 + E\cdot16^1 + F\cdot16^0 = 10\cdot16^5 + 11\cdot16^4 + 12\cdot16^3 + 13\cdot16^2 + 14\cdot16^1 + 15\cdot16^0.$$

Таблица 3.1. **Позиционные системы счисления**

Система счисления	Основание	Алфавит цифр
Десятичная	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Двоичная	2	0, 1
Восьмеричная	8	0, 1, 2, 3, 4, 5, 6, 7
Шестнадцатеричная	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Первая позиционная система счисления была придумана еще в древнем Вавилоне, причем вавилонская нумерация была шестидесятеричной, т. е. в ней использовалось шестьдесят цифр! Интересно, что до сих пор при измерении времени мы используем основание, равное 60 (в 1 минуте содержится 60 секунд, а в 1 часе — 60 минут).

В XIX веке довольно широкое распространение получила двенадцатеричная система счисления. До сих пор мы часто употребляем дюжину (число 12): в сутках две дюжины часов, круг содержит тридцать дюжин градусов и т. д.

Контрольные вопросы

1. Чем отличаются позиционные системы счисления от непозиционных?
2. Каково основание десятичной системы счисления? Двоичной системы счисления? Восьмеричной системы счисления? Шестнадцатеричной системы счисления?
3. Какие цифры входят в алфавит десятичной системы счисления? Двоичной системы счисления? Восьмеричной системы счисления? Шестнадцатеричной системы счисления?

4. Во сколько раз в позиционных системах счисления различаются цифры соседних разрядов числа?

5. Может ли в качестве цифры в системе счисления использоваться символ буквы?

Задания для самостоятельного выполнения

3.1. *Задание с кратким ответом.* Записать числа $3,14_{10}$ и $10,1_2$ в развернутой форме.

3.2. *Задание с кратким ответом.* Во сколько раз увеличатся числа $10,1_{10}$ и $10,1_2$ при переносе запятой на один знак вправо?

3.3. *Задание с кратким ответом.* При переносе запятой на два знака вправо число $11,11_x$ увеличилось в 4 раза. Чему равно основание x системы счисления?

3.4. *Задание с кратким ответом.* Какое минимальное основание может иметь система счисления, если в ней записано число 11? Число 99?

3.5. *Задание с кратким ответом.* Записать год, месяц и число своего рождения с помощью римских цифр.

3.1.2. Арифметические операции в позиционных системах счисления

Арифметические операции во всех позиционных системах счисления выполняются по одним и тем же хорошо известным вам правилам.

Сложение. Рассмотрим сложение чисел в двоичной системе счисления. В его основе лежит таблица сложения одноразрядных двоичных чисел:

$$0 + 0 = 0,$$
$$0 + 1 = 1,$$
$$1 + 0 = 1,$$
$$1 + 1 = 10.$$

Важно обратить внимание на то, что при сложении двух единиц происходит переполнение разряда и производится перенос в старший разряд. Переполнение разряда наступает тогда, когда число в нем становится равным или большим основания системы счисления. Для двоичной системы счисления это число равно двум.

Сложение многоразрядных двоичных чисел производится 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:

$$\begin{array}{r} 110_2 \\ + \quad 11_2 \\ \hline 1001_2 \end{array}$$

Проверим правильность вычислений сложением в десятичной системе счисления. Переведем двоичные числа в десятичную систему счисления и затем их сложим:

$$110_2 = 1\cdot2^2 + 1\cdot2^1 + 0\cdot2^0 = 6_{10},$$
$$11_2 = 1\cdot2^1 + 1\cdot2^0 = 3_{10},$$
$$6_{10} + 3_{10} = 9_{10}.$$

Теперь переведем результат двоичного сложения в десятичное число:

$$1001_2 = 1\cdot2^3 + 0\cdot2^2 + 0\cdot2^1 + 1\cdot q2^0 = 9_{10}.$$

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:

$$0 - 0 = \quad 0,$$
$$0 - 1 = \bar{1}1,$$
$$1 - 0 = \quad 1,$$
$$1 - 1 = \quad 0.$$

Вычитание многоразрядных двоичных чисел производится в соответствии с вышеприведенной таблицей вычитания с учетом возможных заемов из старших разрядов. В качестве примера произведем вычитание двоичных чисел 110_2 и 11_2:

$$\begin{array}{r} 110_2 \\ - \quad 11_2 \\ \hline 11_2 \end{array}$$

Умножение. В основе умножения лежит таблица умножения одноразрядных двоичных чисел:

$$0 \times 0 = 0,$$
$$0 \times 1 = 0,$$
$$1 \times 0 = 0,$$
$$1 \times 1 = 1.$$

Умножение многоразрядных двоичных чисел производится 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:

$$\begin{array}{r|l} 110_2 & 11_2 \\ \hline -11 & 10_2 \\ \hline 0 & \end{array}$$

Для проведения арифметических операций над числами, выраженными в различных системах счисления, необходимо предварительно перевести их в одну и ту же систему.

Задания для самостоятельного выполнения

3.6. *Задание с развернутым ответом.* Провести сложение, вычитание, умножение и деление двоичных чисел 1010_2 и 10_2.

3.1.3. *Двоичное кодирование чисел в компьютере

Числа в компьютере хранятся и обрабатываются в двоичной системе счисления. Оперативная память компьютера состоит из ячеек, в каждой из которых может храниться

8 битов информации, т. е. в каждой ячейке может храниться 8 разрядов двоичного числа.

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.

Для хранения **целых неотрицательных чисел** отводится одна ячейка памяти (8 битов). Например, число $A_2 = 11110000_2$ будет храниться в ячейке памяти следующим образом:

1	1	1	1	0	0	0	0

Определим диапазон чисел, которые могут храниться в оперативной памяти в формате целого неотрицательного числа. Минимальное число соответствует восьми нулям, хранящимся в восьми ячейках памяти, и равно 0. Максимальное число соответствует восьми единицам, хранящимся в ячейках памяти, и равно:

$$A = 1\cdot2^7 + 1\cdot2^6 + 1\cdot2^5 + 1\cdot2^4 + 1\cdot2^3 + 1\cdot2^2 + 1\cdot2^1 + 1\cdot2^0 =$$
$$= 1\cdot2^8 - 1 = 255_{10}.$$

Таким образом, диапазон изменения целых неотрицательных чисел от 0 до 255.

Для хранения **целых чисел со знаком** отводится две ячейки памяти (16 битов), причем старший (левый) разряд отводится под знак числа (если число положительное, то в знаковый разряд записывается 0, если число отрицательное, записывается 1).

Например, отрицательное число $-2002_{10} = -11111010010_2$ будет записано в 16-разрядном представлении следующим образом:

Знак	Число													
1	0	0	0	0	1	1	1	1	1	0	1	0	0	1	0

Максимальное положительное число (с учетом выделения одного разряда на знак) для данного формата представления равно:

$$A = 2^{15} - 1 = 32\,767_{10}.$$

Достоинствами представления чисел в формате с фиксированной запятой являются простота и наглядность представления чисел, а также простота алгоритмов реализации арифметических операций. Недостатком является небольшой диапазон представления величин, недостаточный для решения математических, физических, экономических и других задач, в которых используются как очень малые дробные, так и очень большие числа.

Для представления чисел в диапазоне от очень маленьких дробей до очень больших чисел с высокой точностью используется формат **с плавающей запятой**. В этом случае положение запятой в записи числа может изменяться. Число в форме с плавающей запятой занимает в памяти компьютера четыре (**число обычной точности**) или восемь (**число двойной точности**) байтов.

Задания для самостоятельного выполнения

3.7. **Задание с развернутым ответом.* Как будет храниться в компьютере десятичное число 10_{10} в формате целого неотрицательного числа и целого числа со знаком?

3.2. Электронные таблицы

3.2.1. Основные параметры электронных таблиц

Электронные таблицы позволяют обрабатывать большие массивы числовых данных. В отличие от таблиц на бумаге, электронные таблицы обеспечивают проведение динамических вычислений, т. е. пересчет по формулам при введении новых чисел. В математике с помощью электронных таблиц можно представить функцию в числовой форме и построить ее график, в физике — обработать результаты лабораторной работы, в географии или истории — представить статистические данные в форме диаграммы.

Электронные таблицы — это работающее в диалоговом режиме приложение, хранящее и обрабатывающее данные в прямоугольных таблицах.

Столбцы, строки, ячейки. Электронная таблица состоит из **столбцов и строк**. Заголовки столбцов обозначаются буквами или сочетаниями букв (A, C, AB и т. п.), заголовки строк — числами (1, 2, 3 и далее) (табл. 3.2).

На пересечении столбца и строки находится **ячейка**, которая имеет индивидуальный адрес. Адрес ячейки электронной таблицы составляется из заголовка столбца и заголовка строки, например *A1*, *B5*, *E3*. *Ячейка, с которой* производятся какие-то действия, выделяется рамкой и называется **активной**. Так, в приведенной ниже таблице 3.2 активной является ячейка B2.

Таблица 3.2. **Электронные таблицы (столбцы, строки, ячейки)**

	A	B	C	D	E
1					
2					
3					
4					
5					

Рабочие листы и книги. При работе на компьютере электронная таблица существует в форме **рабочего листа**, который имеет имя (например, *Лист 1*). Рабочие листы объединяются в **книги**, *причем пользователь может рабочие листы вставлять, копировать, удалять и переименовывать. При* создании, открытии или сохранении документа в электронных таблицах речь идет фактически о создании, открытии или сохранении книги.

При работе с электронными таблицами можно вводить и изменять данные одновременно на нескольких рабочих листах, а также выполнять вычисления на основе данных из нескольких листов.

Диапазон ячеек. В процессе работы с электронными таблицами достаточно часто требуется работать с несколькими ячейками. Эти ячейки образуют диапазон, который задается адресами ячеек верхней и нижней границ диапазона, разделенными *двоеточием. Можно выделить несколько ячеек* в столбце (диапазон B1:B4), несколько ячеек в строке (диапазон C1:E1) или прямоугольный диапазон (диапазон D3:E4) (табл. 3.3).

Таблица 3.3 **Диапазоны ячеек в столбце, строке и прямоугольный диапазон**

	A	B	C	D	E
1					
2					
3					
4					
5					

Внешний вид таблицы. Внешний вид таблицы, выделенных диапазонов ячеек или отдельных ячеек можно изменять. Для границ ячеек можно установить различные типы линий (одинарная, пунктирная, двойная и др.), их толщину и цвет. Сами ячейки можно закрасить в любой цвет путем выбора цвета из палитры цветов.

Редактирование листов. Из таблицы можно удалять столбцы, строки, диапазоны ячеек и отдельные ячейки. В процессе удаления диапазонов ячеек и отдельных ячеек требуется указать, в какую сторону (влево или вверх) будет производиться сдвиг ячеек.

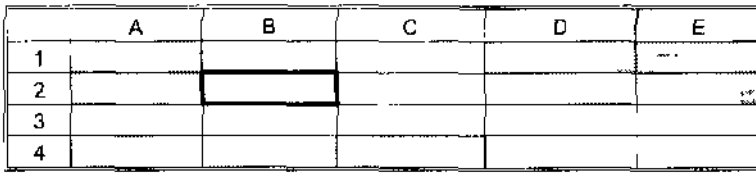
В таблицу можно вставлять столбцы, строки и ячейки. В процессе вставки диапазонов ячеек и отдельных ячеек требуется указать, в какую сторону (вправо или вниз) будет производиться сдвиг ячеек.

Контрольные вопросы

1. Как обозначаются столбцы и строки электронной таблицы? Как задается имя ячейки?
2. Какие операции можно производить над основными объектами электронных таблиц (ячейками, диапазонами ячеек, столбцами, строками, листами, книгами)?

Задания для самостоятельного выполнения

3.8. *Задание с кратким ответом* Записать имя активной ячейки и имена выделенных диапазонов ячеек.



	A	B	C	D	E
1					
2					
3					
4					

3.2.2. Основные типы и форматы данных

В работе с электронными таблицами можно выделить три основных типа данных: *числа*, *текст* и *формулы*.

Числа. Для представления чисел могут использоваться форматы нескольких различных типов: *числовой*, *экспоненциальный*, *дробный* и *процентный*. Существуют специальные форматы для хранения *дат* (например, 25.09.2003) и *времени* (например, 13:30:55), а также *финансовый* и *денежный* форматы (например, 1500,00 р.), которые используются при проведении бухгалтерских расчетов.

По умолчанию для представления чисел электронные таблицы *используют числовой* формат, который отображает два десятичных знака числа после запятой (например, 195,20).

Экспоненциальный формат применяется, если число, содержащее большое количество разрядов, не умещается в ячейке. В этом случае разряды числа представляются с помощью положительных или отрицательных степеней числа 10. Например, числа 2000000 и 0,000002, представленные в экспоненциальном формате как $2 \cdot 10^6$ и $2 \cdot 10^{-6}$, будут записаны в ячейке электронных таблиц в виде 2,00E+06 и 2,00E–06.

По умолчанию числа выравниваются в ячейке по правому краю. *Это объясняется тем, что при размещении чисел* друг под другом (в столбце таблицы) удобно иметь выравнивание по разрядам (единицы под единицами, десятки под десятками и т.д.).

Текст. Текстом в электронных таблицах является последовательность символов, состоящая из букв, цифр и пробелов, *например текстом может быть последовательность* цифр 2008. По умолчанию текст выравнивается в ячейке по левому краю. Это объясняется традиционным способом *письма (слева направо)*.

Формулы. Формула должна начинаться со знака равенства и может включать в себя числа, имена ячеек (ссылки на адреса ячеек), функции и знаки математических операций. Однако в формулу не может входить текст.

Например, формула =A1+B1 обеспечивает сложение чисел, хранящихся в ячейках A1 и B1, а формула =A1*5 — умножение числа, хранящегося в ячейке A1, на 5. При изменении исходных значений, входящих в формулу, результат пересчитывается немедленно.

В процессе ввода формулы она отображается как в самой ячейке, так и в строке формул (рис. 3.3). Если задан режим отображения значений, то после окончания ввода, которое обеспечивается нажатием клавиши {*Enter*}, в ячейке отображается не сама формула, а результат вычислений по этой формуле.

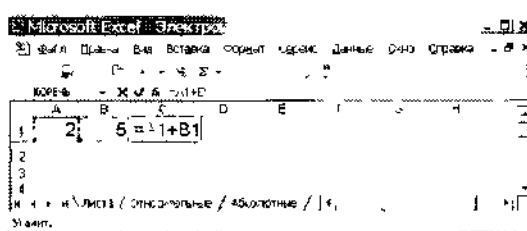


Рис 3.3. Формула в электронных таблицах

Для просмотра формулы необходимо выделить ячейку с формулой, в строке формул появится введенная ранее формула. Для редактирования формулы необходимо щелкнуть по ячейке или строке формул и провести редактирование. Для одновременного просмотра всех введенных формул можно задать специальный режим отображения формул, при котором в ячейках отображаются не результаты вычислений, а сами формулы.

Ввод и копирование данных. Ввод в ячейки чисел, текстов и формул производится с помощью клавиатуры.

Ввод в формулы имен ячеек можно осуществлять выделением нужной ячейки с помощью мыши.

Данные можно копировать или перемещать из одних ячеек или диапазонов ячеек в другие ячейки или диапазоны ячеек. В процессе копирования можно вставлять в ячейки не только сами данные, но и формат данных и параметры оформления ячеек (*тип границы и цвет заливки*).

Для быстрого копирования данных из одной ячейки сразу во все ячейки определенного диапазона используется специальный метод: сначала выделяется ячейка и требуемый диапазон, а затем вводится команда *Заполнить* ⇒ *вниз* [*вправо, вверх, влево*].

Контрольные вопросы

1. Данные каких типов могут обрабатываться в электронных таблицах?
2. В каких форматах данные могут быть представлены в электронных таблицах?

Задания для самостоятельного выполнения

3.9. Задание с кратким ответом. Записать формулы:
а) сложения чисел, хранящихся в ячейках A1 и B1;
б) вычитания чисел, хранящихся в ячейках A3 и B5;
в) умножения чисел, хранящихся в ячейках C1 и C2;
г) деления чисел, хранящихся в ячейках A10 и B10.

3.2.3. Относительные, абсолютные и смешанные ссылки

Как мы говорили, в формулах могут использоваться ссылки на адреса ячеек. Существуют два основных типа ссылок: относительные и абсолютные. Различия между относительными и абсолютными ссылками проявляются при копировании формулы из активной ячейки в другие ячейки.

Относительные ссылки. При перемещении или копировании формулы из активной ячейки относительные ссылки автоматически изменяются в зависимости от положения ячейки, в которую скопирована формула. При смещении положения ячейки на одну строку в формуле изменяются на единицу номера строк, а при смещении на один столбец на одну букву смещаются имена столбцов.

Так, при копировании формулы из активной ячейки C1, содержащей относительные ссылки на ячейки A1 и B1, в ячейку D2 значения столбцов и строк в формуле изменятся на один шаг вправо и вниз. При копировании формулы из ячейки C1 в ячейку E3 значения столбцов и строк в формуле изменятся на два шага вправо и вниз и т. д. (табл. 3.4).

Таблица 3.4. **Относительные ссылки**

	A	B	C	D	E
1			=A1*B1		
2				=B2*C2	
3					=C3*D3

Абсолютные ссылки. Абсолютные ссылки в формулах используются для указания фиксированных адресов ячеек. При перемещении или копировании формулы абсолютные ссылки не изменяются. В абсолютных ссылках перед неизменяемыми обозначениями столбца и строки, составляющими адрес ячейки, ставится знак доллара (например, $A$1).

Так, при копировании формулы из активной ячейки C1, содержащей абсолютные ссылки на ячейки $A$1 и $B$1, значения столбцов и строк в формуле не изменятся (табл. 3.5).

Таблица 3.5. **Абсолютные ссылки**

	A	B	C	D	E
1			=$A$1*$B$1		
2				=$A$1*$B$1	
3					=$A$1*$B$1

Смешанные ссылки. В формуле можно использовать смешанные ссылки, в которых координата столбца относительная, а строки — абсолютная (например, A$1), или, наоборот, координата столбца абсолютная, а строки — относительная (например, $B1) (табл. 3.6).

Таблица 3.6. **Смешанные ссылки**

	A	B	C	D	E
1			=A$1*$B1		
2				=B$1*$B2	
3					=C$1*$B3

Контрольные вопросы

1. Как изменяется при копировании в ячейку, расположенную в соседнем столбце и строке, формула, содержащая относительные ссылки? Абсолютные ссылки? Смешанные ссылки?

3.2.4. Встроенные функции

Формулы могут включать в себя не только адреса ячеек и знаки арифметических операций, но и функции. Электронные таблицы имеют несколько сотен встроенных функций, которые подразделяются на категории: *Математические*, *Статистические*, *Финансовые*, *Дата и время* и т. д.

Суммирование. Одной из наиболее часто используемых операций является суммирование значений диапазона ячеек. Для суммирования значений диапазона необходимо его выделить, причем для ячеек, расположенных в одном столбце или строке, достаточно для вызова функции суммирования чисел СУММ() щелкнуть по кнопке Σ *Автосумма* на панели инструментов *Стандартная*.

Результат суммирования будет записан в ячейку, следующую за последней ячейкой диапазона в столбце (например, =СУММ(A2:A4)), строке (например, =СУММ(C1:E1)) или прямоугольном диапазоне ячеек (например, =СУММ(C3:E4)) (рис. 3.4).

	A	B	C	D	E	F
1			1	2	3	=СУММ(C1:E1)
2	1					
3	2		1	2	3	
4	3		4	5	6	=СУММ(C3:E4)
5	=СУММ(A2:A4)					

Рис. 3.4. Суммирование значений диапазонов ячеек

При суммировании значений ячеек выделенный диапазон можно откорректировать путем перемещения границы диапазона с помощью мыши или введением в формулу адресов ячеек с клавиатуры.

Степенная функция. В математике широко используется степенная функция $y = x^n$, где x — аргумент, а n — показатель степени (например, $y = x^2$, $y = x^3$ и т. д.). Ввод функций в формулы можно осуществлять с помощью клавиатуры или с помощью *Мастера функций*, который предоставляет пользователю возможность вводить функции с использованием последовательностей диалоговых окон.

Например, если в ячейке B1 хранится значение аргумента функции x, то вид функции x^2, введенной с клавиатуры (ячейка B2), будет =B1^2, а введенной с помощью *Мастера функций* (ячейка B3) — СТЕПЕНЬ(B1;2) (рис. 3.5).

	A	B
1	x	-4
2	y = x^2	=B1^2
3	y=СТЕПЕНЬ(x;2)	=СТЕПЕНЬ(B1,2)

Рис. 3.5. Степенная функция $y = x^2$

Квадратный корень. Квадратный корень является степенной функцией с дробным показателем, где $n = 1/2$. Записывается эта функция обычно с использованием знака квадратного корня $y = \sqrt{x}$.

Например, если в ячейке B1 хранится значение аргумента функции x, то вид функции $\sqrt{x}$, введенной с клавиатуры (ячейка B2), будет =B1^(1/2), а введенной с помощью *Мастера функций* (ячейка B3) — КОРЕНЬ(B1) (рис. 3.6).

	A	B
1	x	4
2	y = x^(1/2)	=B1^(1/2)
3	y =КОРЕНЬ(x)	=КОРЕНЬ(B1)

Рис. 3.6. Квадратный корень $y = \sqrt{x}$

Таблица значений функции. В электронных таблицах можно не только вычислить значение функции для любого заданного значения аргумента, но и представить ее в форме таблицы числовых значений аргумента и вычисленных значений функции.

Заполнение таблицы можно существенно ускорить, если использовать операцию *Заполнить*. Сначала в первую ячейку строки аргументов вводится наименьшее значение аргумента (например, в ячейку B1 вводится число –4), а во

вторую ячейку вводится формула, вычисляющая следующее значение аргумента с учетом величины шага аргумента (например, =B1+1). Далее эта формула вводится во все остальные ячейки таблицы с использованием операции *Заполнить вправо*.

Аналогично, в первую ячейку строки значений функции вводится формула вычисления функции (например, в ячейку B2 вводится формула =B1^2), далее эта формула вводится во все остальные ячейки таблицы с использованием операции *Заполнить вправо* (табл. 3.7).

Таблица 3.7. **Числовое представление квадратичной функции $y = x^2$**

	A	B	C	D	E	F	G	H	I	J
1	*x*	−4	−3	−2	−1	0	1	2	3	4
2	*y* =*x*^2	16	9	4	1	0	1	4	9	16

Задания для самостоятельного выполнения

3.10. *Задание с кратким ответом.* Какие значения будут получены в ячейках A5, F1 и F4 после суммирования различных диапазонов ячеек (см. рис. 3.4)? Проверить в электронных таблицах.

3.11. *Задание с кратким ответом.* Какие значения будут получены в ячейках B2 и B3 после вычисления значений степенной функции (см. рис. 3.5)? Проверить в электронных таблицах.

3.12. *Задание с кратким ответом.* Какие значения будут получены в ячейках B2 и B3 после вычисления значений квадратного корня (см. рис. 3.6)? Проверить в электронных таблицах.

3.3. Построение диаграмм и графиков в электронных таблицах

Электронные таблицы позволяют визуализировать данные, размещенные на рабочем листе, в виде **диаграммы**. Диаграммы наглядно отображают зависимости между данными, что облегчает восприятие и помогает при анализе и сравнении данных.

Типы диаграмм (рис. 3.7). Различные типы диаграмм позволяют представлять данные в различных формах. Для каждого набора данных важно правильно подобрать тип создаваемой диаграммы.

Для наглядного сравнения различных величин используются **линейчатые диаграммы**, в которых высота столбца пропорциональна значению величины. Линейчатые диаграммы могут быть плоскими или объемными, причем столбцы могут быть расположены как вертикально (гистограмма), так и горизонтально. Например, с помощью линейчатой диаграммы можно наглядно представить данные о численности населения различных стран мира.

Для отображения величин частей некоторого целого применяется **круговая диаграмма**, в которой площадь кругового сектора пропорциональна величине части. Круговые диаграммы могут быть плоскими или объемными, причем секторы могут быть раздвинуты (разрезанная круговая диаграмма). Например, круговая диаграмма позволяет наглядно показать долю стоимости отдельных устройств компьютера в его общей стоимости.

Для построения графиков функций и отображения изменения величин в зависимости от времени используются диаграммы типа **график**. На плоских графиках маркерами отображаются значения числовой величины, которые соединяются между собой плавными линиями. Объемные графики представляют изменение величины с помощью цветной трехмерной фигуры (см. рис. 3.7).

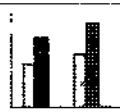
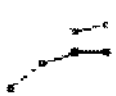
Тип диаграммы	Внешний вид диаграммы	
Линейчатая	 Вертикальная (гистограмма)	Горизонтальная объемная
Круговая	Плоская	Объемная разрезанная
График	 С маркерами	Объемный

Рис. 3.7. Основные типы диаграмм: линейчатая, круговая, график

Диапазон исходных данных: ряды данных и категории. При создании диаграммы в электронных таблицах прежде всего необходимо выделить диапазон ячеек, содержащий исходные данные для ее построения. Диаграммы связаны с исходными данными на рабочем листе и обновляются при обновлении данных на рабочем листе.

Выделенный диапазон исходных данных включает в себя ряды данных и категории.

Ряд данных — это множество значений, которые необходимо отобразить на диаграмме. На линейчатой диаграмме значения ряда данных отображаются с помощью столбцов, на круговой — с помощью секторов, на графике — точками, имеющими заданные координаты *Y*.

Категории задают положение значений ряда данных на диаграмме. На линейчатой диаграмме категории являются подписями под столбцами, на круговой диаграмме — названиями секторов, а на графике категории используются для обозначения делений на оси *X*. Если диаграмма отображает изменение величины во времени, то категории всегда являются интервалами времени (дни, месяцы, годы и т. д.).

Ряды данных и категории могут размещаться как в столбцах, так и в строках электронной таблицы.

Оформление диаграммы. Диаграммы могут располагаться как на отдельных листах, так и на листе с данными (внедренные диаграммы). *Область диаграммы* кроме обязательной *области построения диаграммы* может содержать *названия оси категорий* и *оси значений*, *заголовок диаграммы* и *легенду* (рис. 3.8).

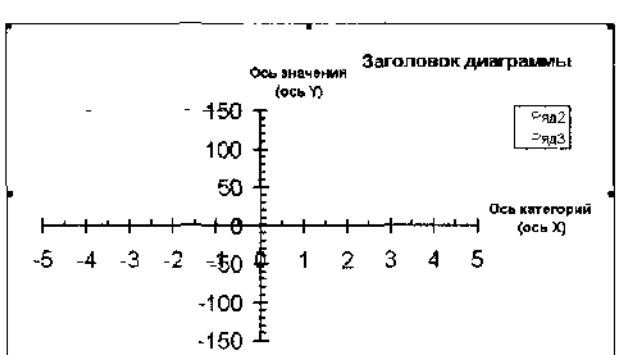


Рис. 3.8. Область диаграммы

Внешний вид диаграммы можно настраивать. С помощью мыши можно изменять размеры области внедренной диаграммы, а также перемещать ее по листу.

Область построения диаграммы является основным объектом в области диаграммы, так как именно в ней производится графическое отображение данных. В линейных диаграммах можно изменять цвет столбцов, в круговых — цвет секторов, в графиках форму, размер и цвет маркеров и соединяющих их линий.

Линейчатые диаграммы и графики содержат *ось категорий* (ось *X*) и *ось значений* (ось *Y*), формат которых можно изменять (толщину, вид и цвет линий).

Важнейшим параметром осей является *шкала*, которая определяет минимальное и максимальное значения шкалы, а также цену основных и промежуточных делений. Рядом с делениями шкалы по оси категорий размещаются названия категорий, а рядом с делениями шкалы по оси значений — значения ряда данных. В круговых диаграммах названия категорий и значения ряда данных отображаются рядом с секторами диаграммы.

Для более точного определения величины столбцов линейчатой диаграммы и положений маркеров графика можно использовать горизонтальные и вертикальные линии *сетки*. Основные линии сетки продолжают основные деления шкалы, а промежуточные линии — промежуточные деления шкалы.

Название диаграммы и *названия осей* можно перемещать и изменять их размеры, а также можно изменять тип шрифта, его размер и цвет.

Легенда содержит названия категорий и показывает используемый для их отображения цвет столбцов в линейчатых диаграммах, цвет секторов в круговых диаграммах, форму и цвет маркеров и линий на графиках. Легенду можно перемещать и изменять ее размеры, а также можно изменять тип используемого шрифта, его размер и цвет.

Контрольные вопросы

1. Какой тип диаграммы целесообразно использовать и почему:
   • для построения графика функции;
   • для сравнительного анализа площадей территорий некоторых стран;
   • для анализа распределения вами времени суток на различные виды деятельности (сон, учеба, выполнение домашних заданий, развлечения и др.)?
2. Как отображаются на диаграммах ряды данных и категории?
3. Каковы основные элементы области диаграммы и их назначение?

3.4. Базы данных в электронных таблицах

3.4.1. Представление базы данных в виде таблицы и формы

Базы данных. Для упорядоченного хранения и обработки связанных между собой данных используются **базы данных.**

База данных представляет собой определенным образом организованную совокупность данных некоторой предметной области, хранящуюся в компьютере.

Табличная форма представления баз данных. Базы данных удобно представлять в виде **таблицы.** В строке таблицы размещаются значения свойств одного объекта, а столбец таблицы хранит значения определенного свойства всех объектов. Например, в базе данных «Записная книжка» в каждой строке таблицы содержится информация об определенном человеке, а значения его «свойств» *Фамилия*, *Телефон*, *E-mail* хранятся в различных столбцах (табл. 3.8).

Таблица 3.8. **База данных «Записная книжка» в табличной форме**

№	Фамилия	Телефон	E-mail
1	Сидоров	111-11-11	sidorov@server.ru
2	Иванов	222-22-22	ivanov@server.ru
3	Петров	333-33-33	petrov@server.ru

Столбцы табличной базы данных называют **полями**. Каждое поле имеет **имя** и может хранить данные определенного **типа** (текст, число, дата/время и т. д.). В базе данных «Записная книжка» полями являются *№* (число), *Фамилия*, *Телефон* и *E-mail* (текст).

Строки таблицы являются **записями** об объектах. Строка хранит набор значений, содержащихся в полях базы данных. Записи можно нумеровать с использованием **счетчика** (поле *№*) — это позволяет однозначно идентифицировать каждую запись в таблице.

Так, в базе данных «Записная книжка» содержится три записи, в каждой из которых хранятся значения четырех свойств.

Достоинством табличного представления базы данных является возможность видеть одновременно несколько записей. Однако если база данных содержит много полей, а значения полей содержат много символов, то не очень удобно осуществлять ввод, просмотр и редактирование записей.

Представление записей базы данных с помощью формы. Для поочередного ввода, просмотра и редактирования записей базы данных часто используется **форма**. Форма позволяет последовательно отображать записи в удобном для пользователя виде.

Обычно на форме размещаются **надписи**, являющиеся именами полей базы данных, и **поля**, в которых отображаются данные выбранной записи базы данных (рис. 3.9).

В процессе создания формы можно указать, какие поля базы данных включить в форму и как расположить поля в окне формы. Пользователь может подобрать подходящий **дизайн** формы (размер и цвет) надписей, текстовых полей и самой формы.

Системы управления базами данных (СУБД). Создание баз данных, а также операции поиска и сортировки данных выполняются специальными программами — системами

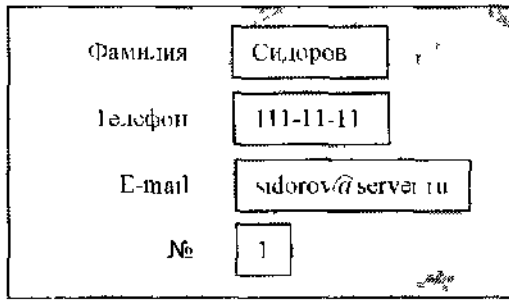


Рис. 3.9. Первая запись базы данных «Записная книжка», отображенная на форме

управления базами данных (СУБД). Таким образом, необходимо различать собственно **базы данных**, которые являются упорядоченными наборами данных, и **системы управления базами данных** — приложения, управляющие хранением и обработкой данных.

Система управления базами данных — это приложение, позволяющее создавать базы данных и осуществлять в них сортировку и поиск данных.

Функцию простои СУБД могут выполнять электронные таблицы. Столбцы таблицы являются полями базы данных, а в строках таблицы размещаются записи базы данных. Первая строка таблицы должна содержать имена полей базы данных.

Создание базы данных с использованием СУБД начинается с создания полей базы данных, установки их типов и ввода имен. Затем в режиме *таблица* или *форма* производится ввод, просмотр и редактирование записей базы данных. После этого в созданной базе данных можно осуществлять сортировку и поиск данных.

В электронных таблицах ввод, просмотр и редактирование записей можно осуществлять как в режиме *таблица*, так и в режиме *форма*. В электронных таблицах Microsoft Excel для вызова формы необходимо выделить ячейки с данными и ввести команду [*Данные Форма...*]. Появится форма, содержащая запись базы данных (рис. 3.10).

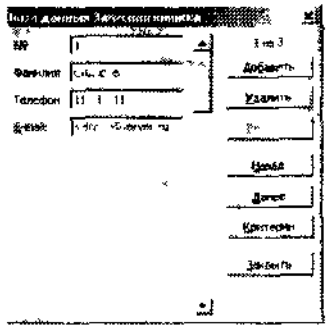


Рис. 3.10. Форма, содержащая первую запись базы данных «Записная книжка»

Контрольные вопросы

1. В чем состоят преимущества и недостатки табличного представления баз данных?
2. В чем состоят преимущества и недостатки представления баз данных с использованием формы?
3. В чем заключается разница между записью и полем в базе данных?
4. Поля каких типов могут присутствовать в базе данных?
5. Существует ли разница между базой данных и СУБД?

3.4.2. Сортировка и поиск данных в электронных таблицах

Сортировка данных в столбцах электронной таблицы. Электронные таблицы позволяют сортировать данные в отдельных столбцах. Если в столбец электронной таблицы ввести данные одного типа (числа, текст, даты или время), можно произвести их сортировку по возрастанию или убыванию. Ниже приведена таблица 3.9, в которой сортировка данных в столбцах проведена следующим образом:

- в столбце A — сортировка чисел по возрастанию;
- в столбце B — сортировка текста по убыванию;
- в столбце C — сортировка дат по возрастанию;
- в столбце D — сортировка времени по убыванию.

Таблица 3.9. **Сортировка чисел, текста, дат и времени в столбцах**

	A	B	C	D
1	-10	бит	суббота, Январь 01, 2000	20:30
2	-5	bit	понедельник, Март 03, 2003	16:30
3	0	$	понедельник, Январь 12, 2004	12:30
4	1	5	среда, Март 03, 2004	8:30
5	5	1	среда, Январь 12, 2005	4:30

При сортировке по возрастанию данные различных типов выстраиваются в следующем порядке:

- числа — от наименьшего отрицательного до наибольшего положительного числа;
- текст — в алфавитном порядке (числа, знаки, латинский алфавит, русский алфавит);
- дата и время — в хронологическом порядке.

При сортировке по убыванию данные выстраиваются в порядке, обратном указанному выше.

Сортировка записей в электронных таблицах. Электронные таблицы могут содержать сотни и тысячи записей (строк). Часто бывает необходимо их упорядочить, т. е. расположить в определенной последовательности. Упорядочение записей называется **сортировкой.**

В электронных таблицах существует режим сортировки, который позволяет после выбора любого столбца расширить диапазон сортируемых данных. В этом случае по данным выделенного столбца будут сортироваться строки (записи базы данных) целиком.

Значения, содержащиеся в выбранном поле, располагаются в порядке **возрастания** или **убывания** их значений, который определяется типом поля. В процессе сортировки целостность записей сохраняется, т. е. строки таблицы перемещаются целиком.

Сортировка данных в электронных таблицах — это упорядочение **записей** (строк) по значениям одного из полей.

Например, после сортировки по возрастанию по текстовому полю *Фамилия* база данных «Записная книжка» примет следующий вид (табл. 3.10).

Таблица 3.10. **Результат сортировки базы данных «Записная книжка»**

№	Фамилия	Телефон	E-mail
2	Иванов	222-22-22	ivanov@server.ru
3	Петров	333-33-33	petrov@server.ru
1	Сидоров	111-11-11	sidorov@server.ru

В электронных таблицах можно проводить **вложенную сортировку**, т. е. сортировать данные последовательно по нескольким полям. При вложенной сортировке строки, имеющие одинаковые значения в ячейках первого поля, будут упорядочены по значениям в ячейках второго поля, а строки, имеющие одинаковые значения во втором поле, будут упорядочены по значениям третьего поля.

Поиск данных в электронных таблицах. Поиск данных в электронных таблицах осуществляется с помощью **фильтров**. Фильтр просто скрывает в исходной таблице записи, не удовлетворяющие условиям поиска.

Поиск данных в электронной таблице — это отбор записей (строк), удовлетворяющих **условиям** поиска, заданным в форме **фильтра**.

Фильтры позволяют отбирать записи, которые удовлетворяют условиям поиска. Условия поиска записей создаются с использованием **операторов сравнения** (=, >, < и т. д.).

Для числовых данных существуют следующие операции сравнения:

- = (равно);
- > (больше);
- < (меньше);
- >= (больше или равно);
- <= (меньше или равно);
- <> (не равно).

Для текстовых данных возможны следующие операции сравнения:

- *равно* (сравниваются все символы);
- *начинается с* и *не начинается с* (сравниваются первые символы);

- *заканчивается на* и *не заканчивается на* (сравниваются последние символы);
- *содержит* и *не содержит* (сравниваются последовательности символов в различных частях текста).

В электронной таблице для задания условия поиска необходимо в базе данных выделить поле, выбрать операцию сравнения и ввести число или последовательность символов. В процессе поиска данные, хранящиеся в ячейках таблицы, будут сравниваться с введенными данными. В результате будут отобраны только те записи базы данных, которые содержат данные, удовлетворяющие условию поиска.

Простые фильтры содержат условие поиска записей только для одного поля. Составные фильтры содержат несколько условий поиска для различных полей. В результате применения составного фильтра будут отобраны только те записи, которые удовлетворяют всем условиям одновременно.

Например, если в базе данных «Записная книжка» ввести простой фильтр для поля *Фамилия*, состоящий из условия *равно Иванов*, то будет найдена и оставлена на экране одна запись базы данных (табл. 3.11).

Таблица 3.11. **Результат сортировки базы данных «Записная книжка»**

№	Фамилия	Телефон	E-mail
2	Иванов	222-22-22	ivanov@server.ru

Контрольные вопросы

1. В чем состоит различие между сортировкой записей базы данных и сортировкой данных в столбцах электронной таблицы?
2. Какие операции сравнения могут использоваться для числовых данных? Для текстовых данных?
3. В чем состоит различие между простыми и составными фильтрами?

Практические работы компьютерного практикума, рекомендуемые для выполнения в процессе изучения главы 3

Компьютерный практикум

3.1 Перевод чисел из одной системы счисления в другую с помощью калькулятора

3.2 *Относительные, абсолютные и смешанные ссылки* в электронных таблицах

3.3 Создание таблиц значений функций в электронных таблицах

3.4 Построение диаграмм различных типов

3.5 Сортировка и поиск данных в электронных таблицах

Глава 4

Основы алгоритмизации и объектно-ориентированного программирования

4.1. Алгоритм и его формальное исполнение

4.1.1. Свойства алгоритма и его исполнители

Во многих отраслях человеческой деятельности для достижения требуемого результата используются **алгоритмы**, содержащие четкие описания последовательности действий. Примерами алгоритмов являются кулинарные рецепты, в которых подробно описана последовательность действий по приготовлению пищи.

Алгоритм приготовления блюда быстрого питания:

1. Высыпать в емкость содержимое пакетика.
2. Налить в емкость 200 мл горячей воды.
3. Тщательно перемешать.

Дискретность. Алгоритмы кулинарных рецептов состоят из отдельных действий, которые обычно нумеруются. Разделение алгоритма на последовательность шагов является важным свойством алгоритма и называется **дискретностью**.

Результативность. Алгоритмами являются известные из начальной школы правила сложения, вычитания, умножения и деления столбиком. Применение этих алгоритмов независимо от количества разрядов в числах и, соответственно, количества вычислительных шагов алгоритма всегда приводит к результату. Получение из исходных данных результата за конечное число шагов называется **результативностью** алгоритма.

Алгоритм сложения целых чисел в десятичной системе счисления:

1. Записать числа в столбик, так чтобы цифры самого младшего разряда чисел (единицы) расположились одна под другой (на одной вертикали).

2. Сложить цифры младшего разряда.
3. Записать результат под горизонтальной чертой на вертикали единиц, если при этом полученная сумма больше или равна величине основания системы счисления (в данном случае 10), перенести десятки в старший разряд десятков.

	1	
2	5	6
1	2	8

4. Повторить пункты 2 и 3 для всех разрядов с учетом переносов из младших разрядов.

Массовость. Алгоритмы сложения, вычитания, умножения и деления могут быть применены для любых чисел, причем не только в десятичной, но и в других позиционных системах счисления (двоичной, восьмеричной, шестнадцатеричной и др.). Возможность применения алгоритма к большому количеству различных исходных данных называется **массовостью**.

Само слово «алгоритм» происходит от «algorithmi» — латинской формы написания имени выдающегося математика IX века аль-Хорезми, который сформулировал правила выполнения арифметических операций.

Исполнители алгоритмов. Алгоритмы широко используются в технике в системах управления объектами. В любой системе управления существует управляющий объект, *который является* **исполнителем** *алгоритма управления*. Так, в системах терморегуляции для поддержания определенной температуры в помещении исполнителем алгоритма может являться как человек, так и микропроцессор.

Алгоритм терморегуляции:

1. Измерить температуру в помещении.
2. Если измеренная температура ниже заданной, включить обогреватель.

Детерминированность. При управлении самолетом используются сложные алгоритмы, исполнителями которых являются пилот или бортовой компьютер. Последовательность выполнения действий, например, при взлете должна быть строго определенной (например, нельзя отрываться от взлетной полосы, пока самолет не набрал необходимую взлетную скорость). Исполнитель алгоритма, выполнив очередную команду, должен точно знать, какую команду необходимо исполнять следующей. Это свойство алгоритма называется детерминированностью.

Выполнимость и понятность. После включения компьютера начинают выполняться алгоритмы тестирования компьютера и загрузки операционной системы. Исполнителем этих алгоритмов является компьютер, поэтому они должны быть записаны на понятном компьютеру машинном языке.

Каждый исполнитель обладает определенным набором, **системой команд**, которые он может выполнить. Алгоритм должен быть понятен исполнителю, т. е. должен содержать только те команды, которые входят в систему его команд.

Выше были приведены примеры алгоритмов из различных областей человеческой деятельности и знаний. В этих алгоритмах различные исполнители выполняли операции над объектами различной природы (материальными объектами и числами). При этом во всех примерах можно выделить следующие основные **свойства алгоритма**:

Результативность и дискретность. Алгоритм должен обеспечивать получение из исходных данных результата за конечное 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. Алгоритм должен содержать команды, входящие в систему команд исполнителя и записанные на понятном исполнителю языке.

Алгоритм — это описание детерминированной последовательности действий, направленных на получение из исходных данных результата за конечное число дискретных шагов с помощью понятных исполнителю команд

Формальное исполнение алгоритма. Из приведенных выше свойств алгоритма вытекает возможность его формального выполнения. Это означает, что алгоритм можно выполнять, не вникая в содержание поставленной задачи, а только строго выполняя последовательность действий, описанных в алгоритме.

Контрольные вопросы

1. Приведите примеры известных вам алгоритмов.
2. Перечислите основные свойства алгоритмов и проиллюстрируйте их примерами.
3. Как вы понимаете формальное исполнение алгоритма?

Задания для самостоятельного выполнения

4.1. *Задание с развернутым ответом.* Записать алгоритм вычитания столбиком целых чисел в десятичной системе счисления.

4.1.2. 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, он может по блок-схеме легко проследить выполнение алгоритма, так как элементы блок-схемы 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
Начало	Прямоугольник с закругленными углами, применяется для обозначения начала или конца алгоритма
Данные	Параллелограмм, предназначен для описания ввода или вывода данных, имеет один вход вверху и один выход внизу
Последовательность команд	Прямоугольник, применяется для описания линейной последовательности команд, имеет один вход вверху и один выход внизу
Да Условие Нет	Ромб, служит для обозначения условий в алгоритмических структурах «ветвление» и «выбор», имеет один вход верху и два выхода (налево, если условие выполняется, и направо если условие не выполняется)
Объявление переменных	Прямоугольник со срезанным углом, применяется для объявления переменных или ввода комментариев

Контрольные вопросы

1. Перечислите основные элементы блок-схем и их назначение.

4.1.3. Выполнение алгоритмов компьютером

Выше мы говорили о возможности формального исполнения алгоритма. Это означает, что 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 и представляли собой очень длинные последовательности нулей и единиц. Составление и отладка таких программ являлись 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, которые называются ассемблерами. 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.

Языки программирования высокого уровня. С середины 50-х годов XX века начали создаваться первые языки программирования высокого уровня. Эти языки были ма-

шинно-независимыми, 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. Одним из первых языков программирования высокого уровня был созданный в 1964 году известный всем Бейсик (Basic).

С конца 50-х годов XX века начали создаваться 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`. Такие языки позволяют легко кодировать основные алгоритмические структуры. Наибольшее влияние на переход к структурному программированию оказал языкALGOL (АЛГОЛ), а затем Pascal (назван его создателем Виртом в честь великого физика Блеза Паскаля. Компания Microsoft создала язык QBasic, а в настоящее время *язык Basic встроен в интегрированную офисную систему* OpenOffice.org.

Существуют различные стили программирования. Перечисленные выше языки поддерживают *процедурный стиль*. Программа, составленная в соответствии с этим стилем, представляет собой последовательность операторов (инструкций), задающих те или иные действия.

Объектно-ориентированные языки. С 70-х годов XX века *начали создаваться объектно-ориентированные языки* программирования, на которых было удобно программировать в *объектно-ориентированном стиле*. В основу этих языков были положены программные объекты, которые объединяли данные и методы их обработки. С течением времени для этих языков были созданы интегрированные среды разработки, позволяющие визуально конструировать графический интерфейс приложений:

- язык Object Pascal был разработан компанией Borland на основе 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;

- язык Gambas был создан по аналогии с языком Visual Basic для разработки приложений с графическим интерфейсом в среде операционной системы Linux.

Java. В 90-е годы XX века в связи с бурным развитием Интернета был создан язык Java, обеспечивающий 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 языке Java компилируется в промежуточный код, который исполняется на компьютере встроенной в браузер виртуальной машиной.

Платформа .NET. В настоящее время многие программисты выбирают интегрированную систему программирования Visual Studio .NET, разработанную корпорацией Microsoft. Эта система предоставляет возможность создавать приложения в различных системах объектно-ориентированного программирования, в которых для создания программного кода используются объектно-ориентированные языки программирования (Visual Basic .NET, Visual C#, Visual J#, Turbo Delphi и др.).

История развития языков программирования

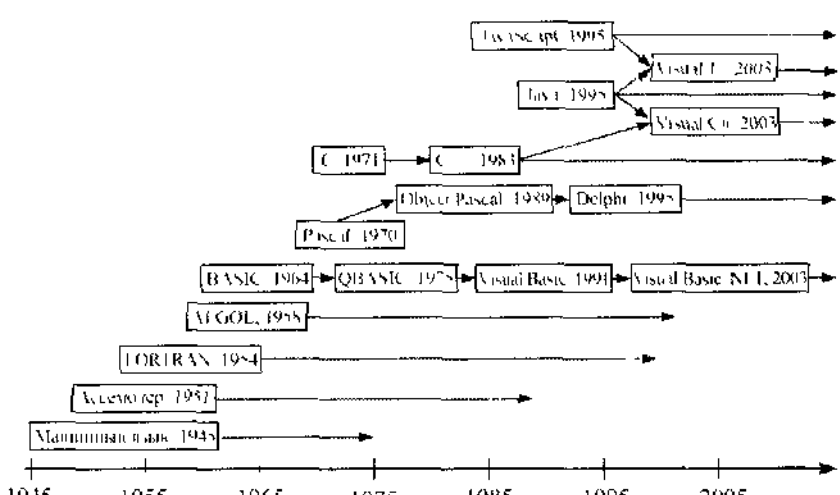


Рис. 4.1. История развития языков программирования

Программы-трансляторы. Для того чтобы программа, записанная на языке программирования, могла быть выполнена компьютером, она должна быть переведена на машинный язык. Эту функцию выполняют программы-трансляторы, загруженные в оперативную память компьютера.

Программы-трансляторы с языков программирования бывают двух типов: **интерпретаторы и компиляторы.** Интерпретатор — это программа, которая обеспечивает последовательный «перевод» команд 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 исполняемом файле (обычно с расширением exe). Затем этот уже готовый к исполнению файл, записанный на машинном языке, можно запускать на выполнение. Достоинством компиляторов является большая скорость выполнения программы, а недостатком большинства из них — трудоемкость отладки, так как невозможно пошаговое выполнение программы.

Системы объектно-ориентированного программирования Visual Basic и Gambas позволяют работать как в режиме интерпретатора, так и в режиме компилятора. На этапе разработки и отладки программы используется режим интерпретатора, а для получения готовой исполняемой программы — режим компилятора.

Система алгоритмического программирования OpenOffice.org Basic позволяет работать только в режиме интерпретатора.

Контрольные вопросы

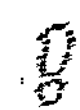
1. Какие преимущества имеют машинно-независимые языки программирования перед машинно-зависимыми языками?
2. В чем состоят достоинства и недостатки интерпретаторов и компиляторов?

4.2. Кодирование основных типов алгоритмических структур на языках объектно-ориентированного и процедурного программирования

Основные алгоритмические структуры кодируются одинаково на объектно-ориентированных языках программирования Visual Basic и Gambas и языке процедурного программирования OpenOffice.org Basic. В этом можно легко убедиться, последовательно рассмотрев основные алгоритмические структуры и их изображение в виде блок-схем и соответствующих операторов на языках, поддерживающих объектно-ориентированный и процедурный стили программирования.

4.2.1. Линейный алгоритм

Существует большое количество алгоритмов, в которых команды должны быть выполнены последовательно одна за другой. Такие последовательности команд будем называть **сериями**, а алгоритмы, состоящие из таких серий, **линейными**.



Алгоритм, в котором команды выполняются последовательно одна за другой, называется **линейным алгоритмом**.

Для того чтобы сделать алгоритм более наглядным, часто используют **блок-схемы**.

На блок-схеме (рис. 4.2) хорошо видна структура линейного алгоритма, по которой исполнителю (человеку) удобно отслеживать процесс его выполнения.

Рис. 4.2. Линейный алгоритм

Контрольные вопросы

1. Как выполняются команды в линейном алгоритме?

4.2.2. Алгоритмическая структура «ветвление»

В отличие от линейных алгоритмов, в которых команды выполняются последовательно одна за другой, в алгоритмическую структуру «ветвление» входит **условие**. В зависимости от выполнения (истинности) или невыполнения (ложности) условия реализуется одна или другая последовательность команд (серий).

В алгоритмической структуре «**ветвление**» в зависимости от истинности или ложности **условия** выполняется одна или другая серия команд.

В условии два числа, две строки, две переменных, два арифметических или строковых выражения сравниваются между собой с использованием операций сравнения (`>`, `<`, `=`, `>=`, `<=`). Например: `5 > 3`, `"A" = "B"` и т. д.

Алгоритмическая структура «ветвление» может быть наглядно представлена с помощью блок-схемы. На языках Visual Basic и Gambas, а также на языке OpenOffice.org Basic ветвление кодируется с использованием **оператора условного перехода** **`If`** ... **`Then`** ... **`Else`** ... **`End If`** (Если ... То ... Иначе ... Конец Если) (рис. 4.3).

В операторе условного перехода после первого ключевого слова **`If`** должно быть размещено условие. Второе ключевое слово **`Then`** размещается на той же строке. Во второй строке размещается последовательность команд (*Серия 1*), которая должна выполняться, если условие истинно. На третьей строке размещается ключевое слово **`Else`**. На четвертой строке размещается последовательность команд (*Серия 2*), которая должна выполняться, если условие ложно. На пятой строке размещается конец инструкции ветвления **`End If`**.

В случае отсутствия серии команд, которую необходимо выполнить при ложности условия, используется сокращенная форма алгоритмической структуры «ветвление». В этом случае в операторе условного перехода отсутствует ключевое слово **`Else`** и, соответственно, последовательность команд *Серия 2* (на рис 4.3 и далее необязательные части оператора заключены в квадратные скобки). Тогда, если условие ложно, выполнение оператора условного перехода заканчивается и происходит переход на следующую строку программы.

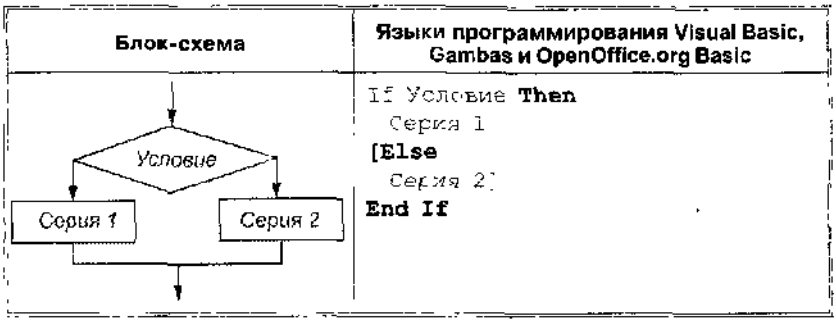


Рис. 4.3. Алгоритмическая структура «ветвление»

Контрольные вопросы

1. В каком случае в алгоритмической структуре «ветвление» выполняется последовательность команд *Серия 1*? *Серия 2*?
2. В каком случае можно использовать сокращенную форму алгоритмической структуры «ветвление»?

Задания для самостоятельного выполнения

4.2. *Задание с развернутым ответом.* Начертить блок-схему алгоритмической структуры «ветвление».

4.2.3. Алгоритмическая структура «выбор»

Алгоритмическая структура «выбор» применяется для реализации ветвлений со многими вариантами серий команд. В структуру выбора входят несколько условий, которые последовательно проверяются. При истинности одного из условий *Условие 1*, *Условие 2* и т. д. выполняется соответствующая последовательность команд *Серия 1*, *Серия 2* и т. д. Если ни одно из условий не истинно, то выполняется последовательность команд *Серия*.

В алгоритмической структуре **«выбор»** выполняется одна из нескольких последовательностей команд при истинности соответствующего **условия**.

Алгоритмическая структура «выбор» может быть наглядно представлена с помощью блок-схемы.

На языках Visual Basic и Gambas, а также на языке OpenOffice.org Basic оператор выбора начинается с ключевых слов **Select Case**, после которых записывается переменная или выражение. После ключевых слов **Case** записываются условия, в которых заданная переменная или выражение сравнивается с определенными значениями. При истинности одного из условий выполняется соответствующая серия команд. Если ни одно из условий не истинно, то выполняется серия команд после ключевого слова **Else**. Заканчивается оператор ключевыми словами **End Select** (рис. 4.4).

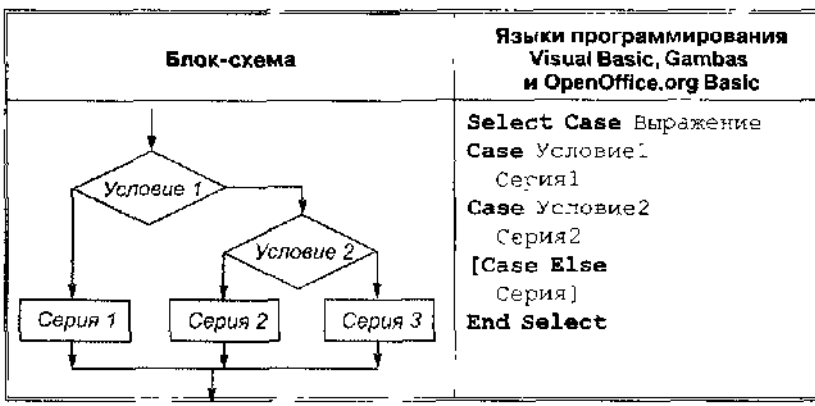


Рис. 4.4. Алгоритмическая структура «выбор»

В случае отсутствия серии команд, которую необходимо выполнить при ложности всех условий, используется сокращенная форма алгоритмической структуры «выбор». В этом случае в операторе выбора отсутствуют ключевые слова **Case Else** и, соответственно, последовательность команд *Серия*. Тогда, если все условия ложны, выполнение оператора выбора заканчивается и происходит переход на следующую строку программы.

Контрольные вопросы

1. В каком случае в алгоритмической структуре «выбор» выполняется последовательность команд *Серия 1*? *Серия 2*?
2. В каком случае можно использовать сокращенную форму алгоритмической структуры «выбор»?

Задания для самостоятельного выполнения

4.3. *Задание с развернутым ответом.* Начертить блок-схему алгоритмической структуры «выбор».

4.2.4. Алгоритмическая структура «цикл»

В алгоритмическую структуру «цикл» входит серия команд, выполняемая многократно. Такая последовательность команд называется **телом цикла**.

Циклические алгоритмические структуры бывают двух типов:

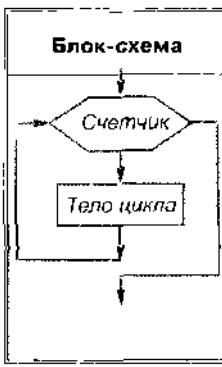
- **цикл со счетчиком**, в котором тело цикла выполняется определенное количество раз;
- **цикл по условию**, в котором тело цикла выполняется, пока истинно условие.

В алгоритмической структуре **«цикл»** серия команд (тело цикла) выполняется многократно.

Цикл со счетчиком. Алгоритмическая структура «цикл со счетчиком» используется, если заранее известно, какое число повторений тела цикла необходимо выполнить. Цикл *со счетчиком может быть зафиксирован графически*, с помощью блок-схемы, а также записан на языках Visual Basic и Gambas и на языке OpenOffice.org Basic с использованием оператора цикла **For ... Next** (рис. 4.5).

Синтаксис оператора **For...Next** следующий: строка, начинающаяся с ключевого слова **For**, является заголовком цикла, а строка с ключевым словом **Next** — концом цикла, между ними располагаются операторы, являющиеся телом цикла.

В начале выполнения цикла значение переменной Счетчик устанавливается равным НачЗнач. При каждом «проходе» цикла значение переменной Счетчик увеличивается на величину шага. Если оно достигает величины КонЗнач, то цикл завершается, и происходит переход на следующую строку программы.



Языки программирования Visual Basic, Gambas и OpenOffice.org Basic

```
For Счетчик=НачЗнач To КонЗнач [Step шаг
   Тело цикла
Next [Счетчик]
```

Рис. 4.5. Алгоритмическая структура «цикл со счетчиком»

Цикл с условием. Алгоритмическая структура «цикл с условием» используется, если заранее неизвестно, какое количество раз необходимо повторить тело цикла. В этом случае количество повторений тела цикла зависит от истинности условия. Цикл с условием можно отобразить с помощью блок-схемы и записать на языках Visual Basic и Gambas и на языке OpenOffice.org Basic с помощью оператора цикла **Do While...Loop** (рис. 4.6).

После ключевого слова **While** записывается условие продолжения цикла. Цикл выполняется, пока истинно условие. Как только условие примет значение «ложь», выполнение цикла закончится. Если условие продолжения цикла стоит перед телом цикла, то такой цикл называется **циклом с предусловием.**

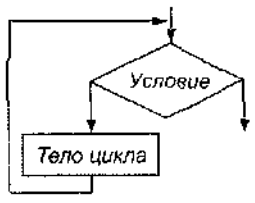
Блок-схема	Языки программирования Visual Basic, Gambas и OpenOffice.org Basic
	**Do While** Условие Тело цикла **Loop**

Рис. 4.6. Алгоритмическая структура «цикл с предусловием»

Контрольные вопросы

1. В каких случаях используется алгоритмическая структура «цикл со счетчиком», а в каких — алгоритмическая структура «цикл с условием»?

Задания для самостоятельного выполнения

4.4. *Задание с развернутым ответом.* Начертить блок-схемы алгоритмических структур «цикл со счетчиком» и «цикл с условием».

4.3. Переменные: тип, имя, значение

В языках Visual Basic и Gambas и в языке OpenOffice.org Basic **переменные** используются для хранения и обработки данных в программах.

Переменные задаются **именами**, определяющими области оперативной памяти компьютера, в которых хранятся **значения** переменных. Значениями переменных могут быть **данные различных типов** (целые или вещественные числа, последовательности символов, логические значения и т. д.).

Переменная в программе представлена **именем** и служит для обращения к **данным** определенного **типа**, конкретное **значение** которых хранится в ячейке оперативной памяти.

Тип переменной. Тип переменной определяется типом данных, которые могут быть значениями переменной. Значениями переменных числовых типов **Byte**, **Short**, **Integer**, **Long**, **Single**, **Double** являются числа, логического типа **Boolean** — значения «истина» (**True**) или «ложь» (**False**), строкового типа **String** — последовательности символов. Обозначения типов переменных являются ключевыми словами языка и поэтому выделяются.

Данные различных типов требуют для своего хранения в оперативной памяти компьютера различное количество ячеек (байтов) (табл. 4.2).

Таблица 4.2. **Типы переменных**

Visual Basic 2005	Gambas	OpenOffice.org Basic	Занимаемая память	Диапазон значений
Целочисленные переменные				
`Byte`	`Byte`		1 байт	от 0 до 255
`Short`	`Short`	`Integer`	2 байта	от –32 768 до 32 767
`Integer`	`Integer`	`Long`	4 байта	от –2 147 483 648 до 2 147 483 647
`Long`	`Long`		8 байтов	от –9 223 372 036 854 775 808 до 9 223 372 036 854 775 807
Переменные с плавающей запятой				
`Single`	`Single`	`Single`	4 байта	от $-1{,}5 \cdot 10^{-45}$ до $3.4 \cdot 10^{38}$, 7–8 значащих цифр
`Double`	`Float`	`Double`	8 байтов	от $-5{,}0 \cdot 10^{-324}$ до $1{,}7 \cdot 10^{308}$, 15–16 значащих цифр
`Decimal`		`Float`	16 байтов	от $\pm 1{,}0 \cdot 10^{-28}$ до $\pm 7{,}9 \cdot 10^{28}$, 28–29 значащих цифр
Строковые переменные				
`String`	`String`	`String`	2 байта × количество символов	от 0 до 65 535 знаков в кодировке *Unicode*. (В языке Gambas 1 байт на символ в кодировке *ASCII*.)
Логические переменные				
`Boolean`	`Boolean`	`Boolean`	2 байта	`True` или `False`

Имя переменной. Имена переменных определяют области оперативной памяти компьютера, в которых хранятся **значения** переменных. Имя каждой переменной (идентификатор) уникально и не может меняться в процессе выполнения программы. В рассматриваемых языках имя переменной может состоять из различных символов (латинские и русские буквы, цифры и т. д.), но должно обязательно начинаться с буквы и не должно включать знак точка «.». Количество символов в имени не может быть более 1023, однако для удобства обычно ограничиваются несколькими символами.

Объявление переменных. Необходимо объявлять переменные, для того чтобы исполнитель программы (компьютер) «понимал», переменные какого типа используются в программе.

Для объявления переменной используется **оператор объявления переменных** `Dim`. С помощью одного оператора можно объявить сразу несколько переменных, например:

```
Dim A As Byte, B As Short, C As Single,
D As String, G As Boolean
```

Присваивание переменным значений. Задать или изменить значение переменной можно с помощью **оператора присваивания**. При выполнении оператора присваивания переменная, имя которой указано слева от знака равенства, получает значение, которое находится справа от знака равенства. Например:

```
A = 255
B = -32768
C = 3.14
D = "информатика"
G = True
```

Значение переменной может быть задано числом, строкой или логическим значением, а также может быть представлено с помощью арифметического, строкового или логического выражения.

Проанализируем процесс выполнения программы компьютером (для определенности записанной на языке Visual Basic). После запуска проекта оператор объявления переменных `Dim` отведет в оперативной памяти для их хранения необходимое количество ячеек (табл. 4.3):

- для целой неотрицательной переменной `A` одну ячейку;
- для целочисленной переменной `B` две ячейки;

- для переменной одинарной точности C четыре ячейки;
- для строковой переменной D по две ячейки на символ;
- для логической переменной G две ячейки.

Таблица 4.3. **Значения переменных в оперативной памяти**

Имя переменной	Оперативная память	
	Номера ячеек	Значение переменной
A	1	255
B	2–3	-32768
C	4–7	3,14
D	8–29	информатика
E	30–31	**True**

Таким образом, в памяти для хранения значений переменных будет отведена 31 ячейка, например, ячейки с 1-й по 31-ю.

Контрольные вопросы

1. В чем состоит разница между типом, именем и значением переменной?
2. Какие основные типы переменных используются в языке программирования Visual Basic 2005? Gambas? OpenOffice.org Basic?
3. Почему рекомендуется объявлять переменные перед их использованием в программе?

Задания для самостоятельного выполнения

4.5. *Задание с кратким ответом.* Определить количество ячеек оперативной памяти, необходимое для хранения значений переменных первых семи типов языка Visual Basic, перечисленных в табл. 4.2.

4.4. Арифметические, строковые и логические выражения

Арифметические выражения. В состав арифметических выражений могут входить переменные числового типа, числа, знаки арифметических операций, а также математические функции.

Порядок вычисления арифметических выражений производится в соответствии с общеизвестным порядком выполнения арифметических операций (возведение в степень, умножение или деление, сложение или вычитание), который может изменяться с помощью скобок.

Строковые выражения. В состав строковых выражений могут входить переменные строкового типа, строки (последовательности символов) и строковые функции.

Над переменными строкового типа и строками может производиться операция **конкатенации.** Она объединяет строки или значения строковых переменных в единую строку. Операция конкатенации обозначается знаком «+», который не следует путать со знаком сложения чисел в арифметических выражениях, или знаком «&».

***Логические выражения.** В состав логических выражений могут входить логические переменные, логические значения, операторы сравнения чисел и строк, а также логические операции. Логические выражения могут принимать лишь два значения: **True** (истина) и **False** (ложь).

Операторы сравнения =, <, >, , <= и >= сравнивают выражение в левой части оператора с выражением в правой части оператора и представляют результат в виде логического значения **True** или **False**. Например:

```
5 > 3 = True;     "A" = "B" = False
```

Над элементами логических выражений могут производиться логические операции, которые на языках программирования обозначаются следующим образом: логическое умножение — **And**, логическое сложение — **Or** и логическое отрицание — **Not**. При записи сложных логических выражений используются скобки. Например:

```
(5 > 3) And ("A" = "B") = False
(5 > 3) Or  ("A" = "B") = True
Not (5 > 3) = False
```

Контрольные вопросы

1. Какие элементы могут входить в состав арифметических, строковых и логических выражений?

4.5. Функции в языках объектно-ориентированного и процедурного программирования

Понятие функции в языках программирования близко к понятию функции в математике. Функция может иметь один или более аргументов. Для каждого набора значений аргументов можно определить значение функции. В программировании говорят, что функция возвращает свое значение, если заданы значения ее аргументов. Функции обычно входят в состав выражений, значения которых присваиваются переменным.

Функции могут быть различных типов: математические, строковые, ввода и вывода, даты и времени и др. Тип функции определяется возможными значениями аргументов и значением функции. В математических функциях значениями как аргументов, так и функций являются числа.

Математические функции. В языке Visual Basic 2005 математические функции реализуются с помощью методов: синус `Math.Sin()`, косинус `Math.Cos()`, квадратный корень `Math.Sqrt()` и др.

В языке Gambas и языке OpenOfficeюцкп Basic математические функции реализуются с помощью числовых функций: синус `Sin()`, косинус `Cos()`, квадратный корень `Sqr()` и др.

Строковые функции (табл. 4.4). В строковых функциях строками являются либо аргументы, либо возвращаемые функциями значения. В языке Visual Basic 2005 и языке OpenOffice.org Basic строковые функции оперируют данными в кодировке *Unicode*, а в языке Gambas — в кодировке *ASCII*.

Функция вырезания левой подстроки `Left()`. В функции вырезания подстроки (части строки) `Left(Строка, Длина)` значением функции является левая подстрока. Подстрока начинается от крайнего левого символа аргумента `Строка` и имеет количество символов, равное значению числового аргумента `Длина`.

Функция вырезания правой подстроки `Right()`. В функции вырезания подстроки `Right(Строка, Длина)` значением функции является правая подстрока. Подстрока начинается от крайнего правого символа аргумента `Строка` и имеет количество символов, равное значению числового аргумента `Длина`.

Функция вырезания произвольной подстроки `Mid()`. В функции вырезания подстроки `Mid(Строка, Позиция, Длина)` значением функции является подстрока. Подстрока начинается с символа аргумента `Строка`, позиция которого задана числовым аргументом `Позиция`, и имеет количество символов, равное значению числового аргумента `Длина`.

Функция определения длины строки `Len()`. В функции определения длины строки `Len(Строка)` аргументом является строка `Строка`, а возвращает функция числовое *значение длины строки (количество символов в строке)*.

Функция `Asc()`. Функция `Asc(Строка)` осуществляет преобразование строки в числовой код ее первого символа. Аргументом функции является строка, а значением — число.

Функция `Chr()`. Функция `Chr(Число)` осуществляет преобразование числового кода в символ. Аргументом функции является число, а значением — символ.

Таблица 4.4. **Строковые функции и их значения**

Язык Visual Basic 2005	Язык OpenOffice.org Basic	Значение функции
`Microsoft.VisualBasic.Len("бит")`	`Len("бит")`	3
`Microsoft.VisualBasic.Left("Килобайт",4)`	`Left("Килобайт",4)`	"Кило"
`Microsoft.VisualBasic.Right("Килобайт",4)`	`Right("Килобайт",4)`	"байт"
`Microsoft.VisualBasic.Mid("информатика",3,5)`	`Mid("информатика",3,5)`	"форма"
`Microsoft.VisualBasic.Asc("и")`	`Asc("и")`	252
`Microsoft.VisualBasic.Chr(255)`	`Chr(255)`	"я"

Функции ввода/вывода данных. В Visual Basic 2005 и в языке OpenOffice.org Basic для ввода данных может использоваться функция `InputBox()`, которая позволяет вводить данные с помощью диалогового окна ввода (рис. 4.7).

Аргументами этой функции являются две строки: `"Сообщение"` и `"Заголовок"`, а значением функции является строка, введенная пользователем в текстовое поле:

```
A = InputBox ("Сообщение",
"Заголовок")
```

Рис. 4.7. Диалоговое окно ввода функции `InputBox()`

Если пользователь введет строку в текстовое поле и щелкнет по кнопке *OK*, то значением функции станет строка, введенная пользователем в текстовое поле. Если пользователь щелкнет по кнопке *Отмена*, то значением функции станет пустая строка `""`.

В языке Visual Basic 2005 *и в языке* OpenOffice.org Basic для вывода данных может использоваться функция `MsgBox()`. Эта функция позволяет выводить сообщения с помощью окна сообщений, на котором можно разместить определенный набор кнопок и информационный значок о типе сообщения:

```
MsgBox("Сообщение" [,ЧисКод1+ЧисКод2]
 ,"Заголовок"])
```

Аргумент `"Сообщение"` выводится в окне сообщений, аргумент `ЧисКод1+ЧисКод2` определяет внешний вид окна, а строка `"Заголовок"` выводится в строке заголовка окна. Последние два аргумента не являются обязательными.

Необязательные части программного кода заключаются в квадратные скобки.

Например, для функции `MsgBox("Сообщение", 48 + 3, "Заголовок")` будет выведено следующее окно сообщений (рис. 4.8).

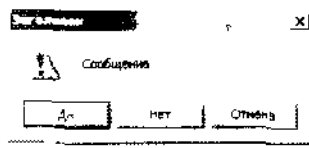


Рис. 4.8. Диалоговое окно сообщений функции `MsgBox()`

Значение, возвращаемое функцией MsgBox(), зависит от того, какая из кнопок на окне сообщений была нажата (табл. 4.5).

Таблица 4.5. **Значения функции MsgBox()**

Нажатая кнопка	Значения функции
ОК	1
Отмена	2
Стоп	3
Повторить	4
Пропустить	5
Да	6
Нет	7

Однако в языке OpenOffice.org Basic для вывода данных часто до сих пор используется оператор **Print**, который выводит строки или числовые выражения, разделенные запятой или точкой с запятой, в диалоговом окне.

В языках Visual Basic 2005 и Gambas для ввода и вывода данных чаще используются элементы управления графического интерфейса. Для ввода данных используется элемент управления *текстовое поле* TextBox, а для вывода данных — элемент управления *метка* Label.

Функции даты и времени. В языках Visual Basic 2005 и Gambas и в языке OpenOffice.org Basic существуют функции даты и времени. Для определенности рассмотрим функции даты и времени, принятые в языке Visual Basic 2005.

Функция Today возвращает значение текущей даты, которое можно присвоить переменным типа Date. Значение даты представляется в виде тройки чисел #Число/Месяц/Год#, разделенных знаком «/».

Функция TimeOfday возвращает значение текущего времени типа **String**, которое можно вывести на надпись. Значение времени представляется в виде тройки чисел #Часы:Минуты:Секунды#, разделениых знаком «:».

Функция Now одновременно возвращает значение текущей даты и текущего времени.

Функция DateDiff(DateInterval.Day, Dat1, Dat2) возвращает разность значений аргументов Dat1, Dat2, рав-

ную количеству дней между датами. Первый аргумент `DateInterval.Day` задает единицу измерения времени.

Контрольные вопросы

1. Какой тип данных могут иметь аргументы и возвращаемые значения математических функций?
2. Какой тип данных могут иметь аргументы и возвращаемые значения строковых функций?
3. Какой тип данных могут иметь аргументы и возвращаемые значения функций ввода и вывода?
4. Какой тип данных могут иметь аргументы и возвращаемые значения функций даты и времени?

4.6. Основы объектно-ориентированного визуального программирования

Проект (Project). С одной стороны, системы объектно-ориентированного визуального программирования являются **системами программирования**, так как позволяют кодировать алгоритмы на этом языке. С другой стороны, системы объектно-ориентированного визуального программирования являются **средами проектирования**, так как позволяют осуществлять визуальное конструирование графического интерфейса.

Результатом процессов программирования и конструирования является **проект**, который объединяет в себе программный код и графический интерфейс. Системы объектно-ориентированного программирования Visual Basic и Gambas содержат и интерпретатор, и компилятор, поэтому проекты могут выполняться в самой системе, а также могут быть преобразованы в **приложения**, которые выполняются непосредственно в операционной системе Windows.

Графический интерфейс проекта. Графический интерфейс необходим для реализации интерактивного диалога пользователя с запущенным на выполнение готовым проек-

том. Основой для создания графического интерфейса разрабатываемого проекта является объект **форма**, которая представляет собой окно, на котором размещаются другие объекты — **элементы управления**.

Элементы управления (рис. 4.9) имеют различное назначение в графическом интерфейсе проекта. *Текстовые поля* (TextBox) используются для ввода и вывода данных, *метки* (Label) — для вывода данных и пояснительных текстов, *графические окна* (PictureBox) — для вывода графики, *кнопки* (Button) — для запуска обработчиков событий.

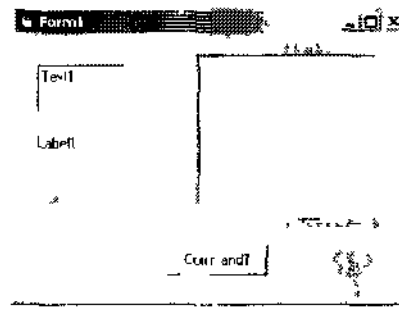


Рис. 4.9. *Элементы управления на форме*

Графический интерфейс проекта представляет собой **форму**, на которой размещены **элементы управления**.

Визуальное конструирование графического интерфейса проекта состоит в том, что на форме с помощью мыши «рисуются» те или иные элементы управления. Выбрав щелчком мышью на *Панели объектов* нужный элемент, мы можем поместить его на *форму разрабатываемого проекта*. Процесс размещения на форме элементов управления аналогичен рисованию графических примитивов с использованием графического редактора.

Объекты (Objects). Как конструирование графического интерфейса, так и разработка программного кода базируется на использовании программных **объектов**. Каждый объект обладает определенным набором **свойств** и может использовать определенные **методы** обработки данных. Если говорить *образно, то объекты — это существительные, свойства объекта* — это прилагательные, а методы объекта — это глаголы.

Программные **объекты** обладают **свойствами** и могут использовать **методы** обработки данных

Объекты могут реагировать на внешние **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 объектов имеет уникальное для данного класса имя.

Например, на основе класса объектов *форма* (`Form`), *который является основой для создания графического интерфейса проекта*, можно создавать экземпляры объектов *форма*, которые получают имена `Form1`, `Form2` и т. д.

Свойства объекта (Properties). Каждый класс объектов обладает определенным набором свойств. Так, например, класс объектов `Form` обладает несколькими десятками различных свойств, которые определяют размеры объекта *форма*, ее цвет, положение на экране монитора и т. д. (табл. 4.6).

Таблица 4.6. **Некоторые свойства объекта *форма***

Свойство	Значение по умолчанию	Комментарий
`Name`	`Form1`	Имя объекта, используется в программном коде для обращения к объекту
`Text`	`Form1`	Текст в левом верхнем углу формы
`BackColor`	`Control`	Серый цвет фона формы
`Font`	`MS Sans Serif`, обычный, 8	Шрифт, его начертание и размер

Различные экземпляры класса объектов обладают одинаковым набором свойств, однако значения свойств у них могут отличаться. Первоначальные значения свойств объектов можно установить с использованием диалогового окна *Свойства* (*Properties*) системы программирования (см. практическую работу 4.1).

Так, для объекта *форма* `Form1` можно установить требуемое значение любого свойства. Для этого необходимо выбрать свойство из списка и изменить его значение.

Значения свойств объектов можно изменять в программном коде. Для присваивания свойству объекта нового значения в левой части строки программного кода необходимо указать имя объекта и затем название свойства, которые в

соответствии с правилами точечной нотации разделяются между собой точкой. В правой части строки необходимо записать конкретное значение свойства:

```
Объект.Свойство = ЗначениеСвойства
```

Например, новая надпись «Первый проект» в левом верхнем углу объекта `Form1` (значение свойства `Text`) появится в результате выполнения программного кода:

```
Form1.Text = "Первый проект"
```

Методы объекта (Methods). Объекты могут использовать различные методы обработки данных. Методы имеют аргументы, которые позволяют задать параметры выполняемых действий.

Для использования метода в строке программного кода необходимо указать имя объекта и затем метод, которые в соответствии с правилами точечной нотации разделяются между собой точкой. В скобках при необходимости записываются аргументы метода, разделяемые запятыми:

```
Объект.Метод(арг1, арг2)
```

Например, с помощью метода `Scale(x,y)` можно изменить размеры формы или элемента управления. Аргументы метода `x` и `y` являются коэффициентами масштабирования по горизонтали и вертикали, т. е. позволяют увеличить или уменьшить ширину и высоту элемента управления. Например, можно в два раза увеличить размер объекта по оси *X* и в два раза его умешьшить по оси *Y*:

```
Me.Scale(2,0.5)
```

Если производятся операции над самой формой, то вместо ее имени (например, `Form1`) в программном коде используется имя **Me**.

Событие (Event). Событие (`Event`) представляет собой действие, распознаваемое элементом управления. Событие может создаваться пользователем (например, щелчок мышью или нажатие клавиши) или быть результатом *воздействия других программных объектов.*

Каждый объект реагирует на определенный набор событий. Например, кнопка реагирует на щелчок (`Click`), нажатие (`MouseDown`) и отпускание (`MouseUp`) кнопки мыши или нажатие определенной клавиши на клавиатуре (`KeyPress`).

Обработчик события. Для каждого события можно запрограммировать отклик, т. е. реакцию объекта на произошедшее событие. Если пользователь производит какое-либо воздействие на элемент графического интерфейса (например, щелчок), в качестве отклика выполняется обработчик события (событийная процедура), представляющий собой программу. Программа может включать основные алгоритмические конструкции (линейная, ветвление, выбор, цикл).

Для того чтобы создать заготовку обработчика события, необходимо в режиме разработки проекта осуществить двойной щелчок мышью по объекту. Например, после щелчка по кнопке `Button1` в окне *Программный код* будет создана заготовка обработчика события:

```
Private Sub Button1_Click(…)

End Sub
```

Служебные слова **Private Sub** и **End Sub** обозначают начало и конец обработчика события. Имя обработчика события `Button1_Click()` включает в себя имя объекта и имя события.

Далее необходимо ввести в обработчик события программный код, который реализует определенный алгоритм.

Обработчик события представляет собой программу, которая начинает выполняться после реализации определенного события.

Контрольные вопросы

1. Что можно изменить в выбранном объекте: набор свойств, набор методов, значения свойств?
2. Какие объекты могут быть использованы при конструировании графического интерфейса проекта?
3. На какие события реагирует кнопка?

4.7. *Графические возможности объектно-ориентированного языка программирования Visual Basic 2005

На форме и управляющих элементах можно рисовать линии, прямоугольники, окружности и другие графические фигуры. Для рисования необходимо определить объекты *Область рисования* Graphics, *Перо* Pen и *Кисть* Brush.

Область рисования. Объект *Область рисования* Graphics позволяет выбрать в качестве области рисования определенный элемент управления и обладает методами рисования графических фигур. Сначала необходимо в разделе объявления переменных определить имя объекта, например:

```
Dim Graph1 As Graphics
```

Затем в программном коде обработчика события необходимо указать определенный элемент управления в качестве области рисования. Обычно в качестве области рисования выбирается размещенное на форме графическое поле (например, PictureBox1):

```
Graph1 = Me.PictureBox1.CreateGraphics()
```

Перо. Объект *Перо* Pen определяет цвет и ширину линии рисования. Сначала необходимо в разделе объявления переменных определить имя объекта (например, Pen1), установить цвет (например, красный Color.Red) и ширину линии в пикселях (например, 3):

```
Dim Pen1 As New Pen(Color.Red, 3)
```

Затем в программном коде обработчика события можно установить новые значения цвета и ширины линии, например:

```
Pen1.Color = Color.Green
Pen1.Width = 15
```

Кисть. Объект *Кисть* Brush определяет цвет и стиль закрашивания прямоугольников, окружностей и других замкнутых фигур. Сначала необходимо в разделе объявления переменных определить имя объекта (например, Brush1) и установить тип закраски и цвет (например, сплошная закраска синего цвета SolidBrush(Color.Blue)):

```
Dim Brush1 As New SolidBrush(Color.Blue)
```

Затем в программном коде обработчика события можно установить новый цвет закраски (например, пурпурный):

```
Brush1.Color = Color.Magenta
```

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 (табл. 4.7).

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` и желтого цвета для объекта `Brush1`:

```
Pen1.Color = Color.Green
Brush1.Color = Color.Yellow
```

Для установки цвета в 24-битовой палитре цветов RGB используется метод `Color.FromArgb(Red, Green, Blue)`, аргументами которого являются три числа в диапазоне от 0 до 255 (интенсивности красного, зеленого и синего цветов). Например, так можно установить пурпурный цвет для объекта `Brush1`:

```
Brush1.Color = Color.FromArgb(255,0,255)
```

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:

```
Dim drawFont As New Font("Arial", 16)
Dim drawBrush As New SolidBrush(Color.Black)
```

Рисование текста в поле рисования можно осуществить так:

```
Graph1.DrawString("Текст", drawFont, drawBrush,
10, 10)
```

Системы координат. Рисование линий, прямоугольников и других фигур производится в компьютерной системе координат, начало которой расположено в верхнем левом углу формы или элемента управления. Ось X направлена вправо, а ось Y — вниз. Единицей измерения по умолчанию является точка (пиксель). Компьютерная система координат графического поля шириной 300 точек и высотой 200 точек приведена на рис. 4.10.

При геометрических построениях и построении графиков функций удобнее использовать математическую систе-

Таблица 4.7. **Графические методы**

Фигура	Графический метод	Аргументы
Линия	`DrawLine(Pen1, X1, Y1, X2, Y2)`	*Перо* (например, `Pen1`), а также координаты концов линии `X1, Y1` и `X2, Y2`
Прямоугольник	`DrawRectangle(Pen1, X1, Y1, Width, Height)`	*Перо* (например, `Pen1`), а также координаты левого верхнего угла `X1, Y1`, ширина `Width` и высота `Height`
Закрашенный прямоугольник	`FillRectangle(Brush1, X1, Y1, Width, Height)`	*Кисть* (например, `Brush1`), а также координаты левого верхнего угла `X1, Y1`, ширина `Width` и высота `Height`
Окружность	`DrawEllipse(Pen1, X1, Y1, Width, Height)`	*Перо* (например, `Pen1`) а также координаты левого верхнего угла описанного прямоугольника `X1, Y1`, его ширина `Width` и высота `Height`
Закрашенная окружность	`FillEllipse(Brush1, X1, Y1, Width, Height)`	*Кисть* (например, `Brush1`), а также координаты левого верхнего угла описанного прямоугольника `X1, Y1`, его ширина `Width` и высота `Height`
Точка	`DrawRectangle(Pen1, X1, Y1, 1, 1)` `DrawEllipse(Pen1, X1, Y1, 1, 1)`	Аргументы `Width` и `Height` равны 1
Стирание	`Clear(Color.White)`	Аргумент – цвет

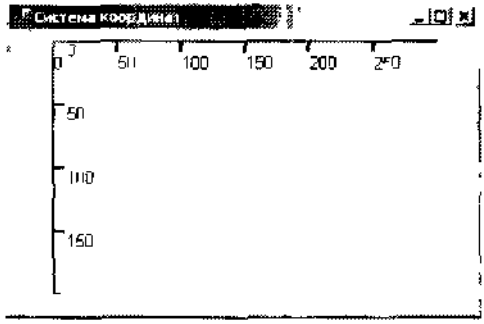


Рис. 4.10. Компьютерная система координат графического поля

му координат, начало которой обычно находится в центре области рисования. Ось *X* направлена вправо, а ось *Y* — вверх. Математическая система координат графического поля шириной 300 точек и высотой 200 точек приведена на рис. 4.11.

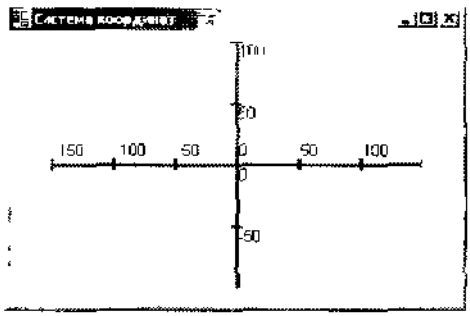


Рис. 4.11. Математическая система координат графического поля

Для преобразования компьютерной системы координат в математическую систему координат используется метод масштабирования и поворота осей `ScaleTransform()` и метод сдвига начала координат `TranslateTransform()`.

Метод

```
Graph1.ScaleTransform(1, -1)
```

обеспечивает поворот оси *Y*.

Метод

```
Graph1.TranslateTransform(150, -100)
```

обеспечивает сдвиг по оси *X* на 150 точек вправо и сдвиг по оси *Y* на 100 точек вниз.

Анимация. Для создания анимации (иллюзии движения на экране какого-либо объекта) применяется принцип смены кадров (изображений), как это делается в мультипликации. Для этого необходимо с определенной частотой рисовать объект в поле рисования, причем координаты объекта должны каждый раз изменяться на определенную величину.

Контрольные вопросы

1. Перечислите методы рисования графических фигур и их аргументы.
2. Каким образом можно изменить систему координат формы или графического поля?
3. Каковы основные этапы создания анимации движения объекта?

Практические работы компьютерного практикума, рекомендуемые для выполнения в процессе изучения главы 4

Компьютерный практикум

4.1 Знакомство с системами объектно-ориентированного и алгоритмического программирования.

4.2 Проект «Переменные»

4.3. Проект «Калькулятор».

4.4 Проект «Строковый калькулятор».

4.5 Проект «Даты и время»

4.6 Проект «Сравнение кодов символов».

4.7 Проект «Отметка».

4.8 Проект «Коды символов».

4.9. Проект «Слово-перевертыш».

4.10. Проект «Графический редактор».

4.11. Проект «Системы координат».

4.12. Проект «Анимация».

Глава 5

Моделирование и формализация

5.1. Окружающий мир как иерархическая система

Микро-, макро- и мегамир. Мы живем в **макромире**, т. е. в мире, который состоит из объектов, по своим размерам сравнимых с человеком. Обычно макрообъекты разделяют на неживые (камень, льдина, бревно и т. д.), живые (растения, животные, 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**.

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**.

Все многообразие объектов мега-, макро- и микромира состоит из **вещества**, при этом все материальные объекты взаимодействуют друг с другом и поэтому обладают **энергией**. Поднятое над поверхностью земли тело обладает механической энергией, нагретый чайник — тепловой, заряженный проводник — электрической, а ядра атомов — атомной.

Окружающий мир можно представить в виде иерархического ряда объектов: элементарных частиц, атомов, молекул, макротел, звезд и галактик. При этом на уровнях молекул и макротел в этом иерархическом ряду образуется ответвление — другой ряд, связанный с живой природой.

В живой природе также существует иерархия: одноклеточные — растения и животные — популяции животных.

Вершиной эволюции жизни на Земле является человек, который не может жить вне общества.

Каждый человек в отдельности и общество в целом изучают окружающий мир и накапливают знания, на основании которых создаются искусственные объекты.

Все вышесказанное можно отобразить в виде схемы на рис. 5.1.

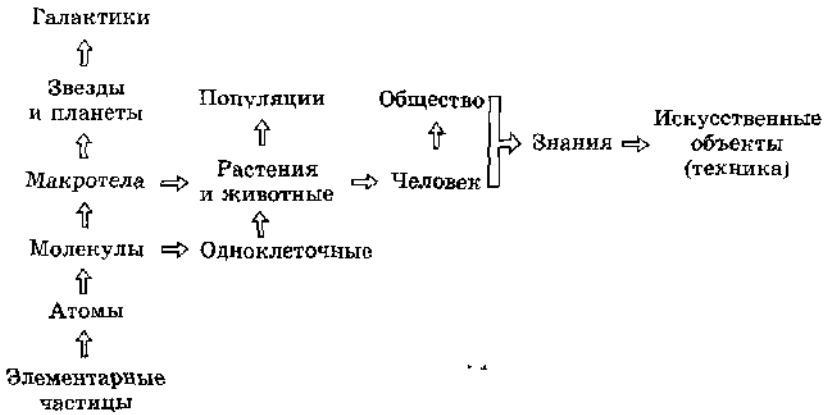


Рис. 5.1. Иерархическая система объектов окружающего мира

Системы и элементы. Каждый объект состоит из других объектов, т. е. представляет собой **систему**. Вместе с тем, каждый объект может входить в качестве **элемента** в систему более высокого структурного уровня. Является ли объект *системой или элементом системы, зависит от точки зрения (целей исследования).*

Система состоит из объектов, которые называются **элементами системы**.

Например, атом водорода можно рассматривать как систему, так как он состоит из положительно заряженного протона и отрицательно заряженного электрона.

Вместе с тем, атом водорода входит в молекулу воды, т. е. является элементом системы более высокого структурного уровня (рис. 5.2).

Рис. 5.2. Атом водорода и молекула воды

Целостность системы. Необходимым условием существования системы является ее **целостное функционирование.** Система является не набором отдельных объектов, а совокупностью взаимосвязанных элементов.

Например, если сложить в кучу устройства, которые входят в состав компьютера (процессор, модули оперативной памяти, системную плату, жесткий диск, корпус, монитор, клавиатуру и мышь), то они не образуют систему. Компьютер, т. е. целостно функционирующая система, образуется только после физического подключения устройств друг к другу, включения питания и загрузки операционной системы (рис. 5.3).

Если из системы удалить хотя бы один элемент, то она может перестать функционировать. Так, если удалить одно из устройств компьютера (например, процессор), компьютер выйдет из строя, т. е. прекратит свое существование как система.

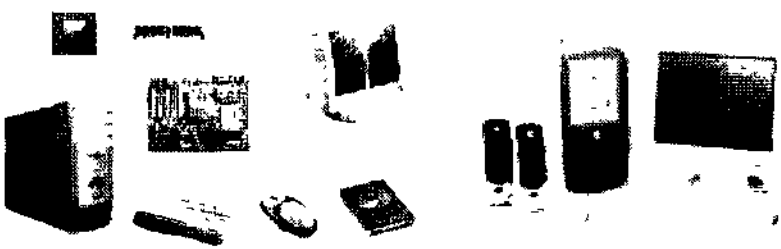


Рис. 5.3. Отдельные объекты (устройства) и целостная система (компьютер)

Взаимосвязь элементов в системах может иметь различную природу. В неживой природе взаимосвязь элементов осуществляется с помощью физических взаимодействий:

- в системах мегамира (например, в Солнечной системе) элементы взаимодействуют между собой посредством сил всемирного тяготения;
- в макротелах происходит электромагнитное взаимодействие между атомами;
- в атомах элементарные частицы связаны ядерными и электромагнитными взаимодействиями.

В живой природе целостность организмов обеспечивается химическими взаимодействиями между клетками, в обществе — социальными связями и отношениями между людьми, в технике — функциональными связями между устройствами и т. д.

Свойства системы. Каждая система обладает определенными свойствами, которые, в первую очередь, *зависят от* набора составляющих ее элементов. Так, свойства химических элементов зависят от строения их атомов.

Атом водорода состоит из двух элементарных частиц (протона и электрона), и соответствующий химический элемент является газом.

Атом лития состоит из трех протонов, четырех нейтронов и трех электронов, и соответствующий химический элемент является щелочным металлом (рис. 5.4).

Рис. 5.4. Атом водорода и атом лития

Свойства системы зависят также от **структуры** системы, т. е. от типа отношений и связей элементов системы между собой. Если системы состоят из одинаковых элементов, но обладают разными структурами, то их свойства могут существенно различаться. Например, алмаз, графит и углеродная нанотрубка состоят из одинаковых атомов (атомов углерода), *однако способы связей между* атомами (кристаллические решетки) существенно различаются (рис. 5.5).

В кристаллической решетке алмаза взаимодействие между атомами очень сильное по всем направлениям, поэтому он является самым твердым веществом на планете и существует в форме кристаллов.

В кристаллической решетке графита атомы размещены слоями, между которыми взаимодействие слабое, поэтому он легко крошится и используется в грифелях карандашей.

Углеродная нанотрубка представляет собой свернутую в цилиндр плоскость кристаллической решетки графита. Нанотрубки очень прочные на разрыв (хотя имеют толщину стенки в один атом углерода). Нить, сделанная из нанотрубок, *толщиной с человеческий волос* способна удерживать груз в сотни килограммов. Электрические свойства нанотрубок могут меняться, что делает их одним из основных материалов наноэлектроники.

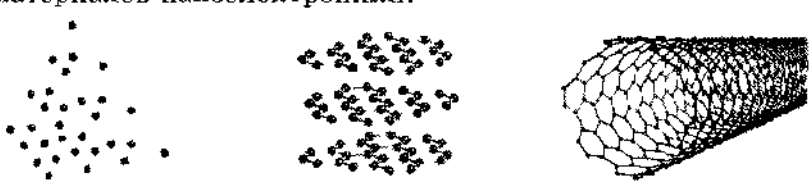


Рис. 5.5. *Кристаллические решетки алмаза, графита* и углеродная нанотрубка

Контрольные вопросы

1. Приведите примеры систем в окружающем мире.
2. Образуют ли систему устройства, из которых состоит компьютер: до сборки? после сборки? после включения компьютера?
3. От чего зависят свойства системы? Приведите примеры систем, состоящих из одних и тех же элементов, но обладающих различными свойствами.

5.2. Моделирование, формализация, визуализация

5.2.1. Моделирование как метод познания

Моделирование. Человечество в своей деятельности (научной, образовательной, технологической, художественной и др.) постоянно создает и использует модели объектов окружающего мира. Строгие правила построения моделей сформулировать невозможно, однако человечество накопило богатый опыт моделирования различных объектов и процессов.

Модели играют чрезвычайно важную роль в проектировании и создании различных технических устройств, машин и механизмов, зданий, электрических цепей и т. д. Без предварительного создания чертежа невозможно изготовить даже простую деталь, не говоря уже о сложном механизме. В процессе проектирования зданий и сооружений кроме чертежей часто изготавливают их макеты. Разработка электрической схемы обязательно предшествует созданию электрических цепей (рис. 5.6) и т. д.

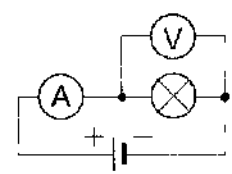


Рис. 5.6. Электрическая схема

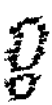
Развитие науки невозможно без создания теоретических моделей (теорий, законов, гипотез и т. д.), отражающих строение, свойства и поведение реальных объектов. Создание новых теоретических моделей иногда коренным образом меняет представление человечества об окружающем мире (например, такую роль сыграла гелиоцентрическая система мира Коперника). Истинность теоретических

моделей, т. е. их соответствие законам реального мира, проверяется с помощью опытов и экспериментов.

Все художественное творчество фактически является процессом создания моделей. Например, такой литературный жанр, как басня, переносит реальные отношения между людьми на отношения между животными и фактически создает модели человеческих отношений. Более того, практически любое литературное произведение может рассматриваться как модель реальной человеческой жизни. Моделями, в художественной форме отражающими реальную действительность, являются также живописные полотна, скульптуры, театральные постановки и т. д.

Моделирование — это метод познания, состоящий в создании и исследовании моделей.

Модель. Каждый объект имеет большое количество различных свойств. В процессе построения модели выделяются главные, **наиболее существенные** для проводимого исследования **свойства**. В процессе исследования аэродинамических качеств модели самолета в аэродинамической трубе важно, чтобы модель имела геометрическое подобие оригиналу, но не важен, например, ее цвет. При построении электрических схем — моделей электрических цепей необходимо учитывать порядок подключения элементов цепи друг к другу, но не важно их геометрическое расположение друг относительно друга и т. д.



Модель создается человеком в процессе познания окружающего мира и отражает **существенные** с точки зрения цели проводимого исследования (цели моделирования) **свойства** изучаемого объекта, явления или процесса.

Разные науки исследуют объекты под разными углами зрения и строят различные типы моделей. В физике изучаются процессы взаимодействия и изменения объектов, в химии — их химический состав, в биологии — строение и поведение живых организмов и т. д.

Рассмотрим в качестве примера человека, в разных науках он исследуется в рамках различных моделей. В рамках

механики его можно рассматривать как материальную точку, в химии — как объект, состоящий из различных химических веществ, в биологии — как систему, стремящуюся к самосохранению и т. д.

География, военное дело, судоходство и другие области человеческой деятельности невозможны без информационных моделей поверхности Земли в виде карт. Различные типы географических карт (политические, физические и т. д.) представляют собой информационные модели, отражающие различные особенности земной поверхности, т. е. один и тот же объект отражают несколько моделей (рис. 5.7).

Рис. 5.7. Модели земной поверхности — физическая и политическая карты мира

Для описания и исследования одного и того же объекта могут использоваться **несколько моделей**.

Вместе с тем, разные объекты могут описываться одной моделью. Например, для описания движения планет, движения автомобиля или движения мяча в определенных условиях (размеры объекта гораздо меньше его перемещений) можно использовать одну и ту же модель движения материальной точки.

Для описания и исследования разных объектов может использоваться **одна и та же модель**.

Никакая модель не может заменить сам объект. Но при решении конкретной задачи, когда нас интересуют определенные свойства изучаемого объекта, модель оказывается полезным, а подчас и единственным инструментом исследования.

Контрольные вопросы

1. Приведите примеры моделирования в различных областях деятельности.
2. Может ли объект иметь несколько моделей? Приведите примеры.
3. Могут ли разные объекты описываться одной и той же моделью? Если да, то приведите примеры.

5.2.2. Материальные и информационные модели

Все модели можно разбить на два больших класса: модели **материальные** и модели **информационные**.

Материальные модели. Материальные модели позволяют представить в материальной наглядной форме объекты, недоступные для непосредственного исследования (очень большие или очень маленькие объекты, очень быстрые или очень медленные процессы и др.). Например, макеты зданий и сооружений позволяют архитекторам выбрать наилучшие градостроительные решения, модели самолетов и кораблей позволяют инженерам выбрать оптимальную форму этих транспортных средств.

Материальные модели часто используются в процессе обучения. В курсе географии первые представления о нашей планете Земля мы получаем, изучая ее модель — глобус (рис. 5.8), в курсе физики изучаем работу двигателя внутреннего сгорания по его модели, в химии при изучении строения вещества используем модели молекул и кристаллических решеток, в биологии изучаем строение человека по анатомическим муляжам.

Рис. 5.8. Предметная модель — глобус Земли

Информационные модели. Информационные модели представляют объекты и процессы в **образной** или **знаковой** форме, а также в форме таблиц, блок-схем, графов и т. д.

Образные модели (рисунки, фотографии и др.) представляют собой зрительные образы объектов, зафиксированные на каком-либо носителе информации (бумаге, фото- и ки-

нопленке и др.). Широко используются образные информационные модели в обучении, где требуется классификация объектов по их внешним признакам (вспомните учебные плакаты по ботанике, биологии и физике).

Знаковые информационные модели строятся с использованием различных языков (знаковых систем). Знаковая информационная модель может быть представлена в форме текста (например, программы на языке программирования) или формулы (например, второго закона Ньютона $\vec{F} = m\vec{a}$).

Широко распространены информационные модели в форме таблиц. В периодической таблице химических элементов Д. И. Менделеева (рис. 5.9) химические элементы располагаются в ячейках таблицы по возрастанию атомных весов, а в столбцах — по количеству валентных электронов. Важно, что по положению в таблице можно определить некоторые физические и химические свойства элементов.

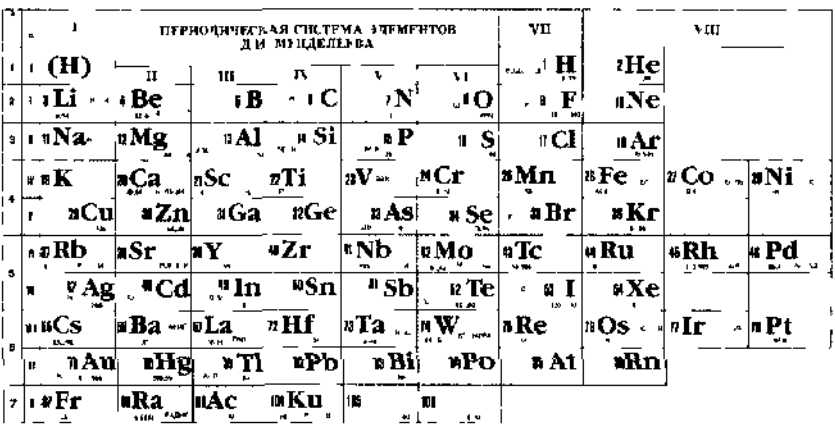


Рис. 5.9. Информационная модель — таблица элементов Менделеева

При построении информационных моделей некоторых типов одновременно используются система графических элементов и знаковая система. Так, в блок-схемах алгоритмов используются различные геометрические фигуры для обозначения элементов алгоритма и формальный алгоритмический язык для записи команд программы (рис. 5.10).

Важную роль играют информационные модели, которые отображают **иерархические системы**. В биологии весь животный мир рассматривается как иерархическая система (тип, класс, отряд, семейство, род, вид), в информатике используется иерархическая файловая система и т. д.

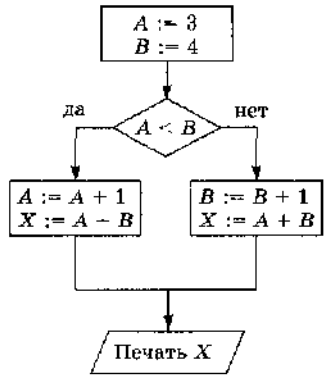


Рис. 5.10. Информационная модель — блок-схема алгоритма

В иерархической информационной модели объекты распределяются по уровням, от первого (верхнего) уровня до нижнего (последнего) уровня. На первом уровне может располагаться только один элемент, который является вершиной иерархической структуры. Основное отношение между уровнями состоит в том, что элемент более высокого уровня может состоять из нескольких элементов нижнего уровня, при этом каждый элемент нижнего уровня может входить в состав только одного элемента верхнего уровня.

Удобным способом наглядного представления иерархических информационных моделей являются **графы**. Элементы иерархической модели отображаются в графе *овалами (**вершинами графа**)*.

Элементы верхнего уровня находятся в отношении «состоять из» к элементам более низкого уровня. Такая связь между элементами отображается в форме **дуги графа** (направленной линии в форме стрелки).

Графы, имеющие одну вершину верхнего уровня, напоминают деревья, которые растут сверху вниз, поэтому называются **деревьями**. **Ребра дерева** могут связывать объекты только соседних иерархических уровней, причем каждый объект нижнего уровня может быть связан ребром только с одним объектом верхнего уровня.

Для описания исторического процесса смены поколений семьи используются информационные модели в форме **генеалогического дерева**. В качестве примера можно рассмотреть фрагмент генеалогического дерева династии Рюриковичей (рис. 5.11).

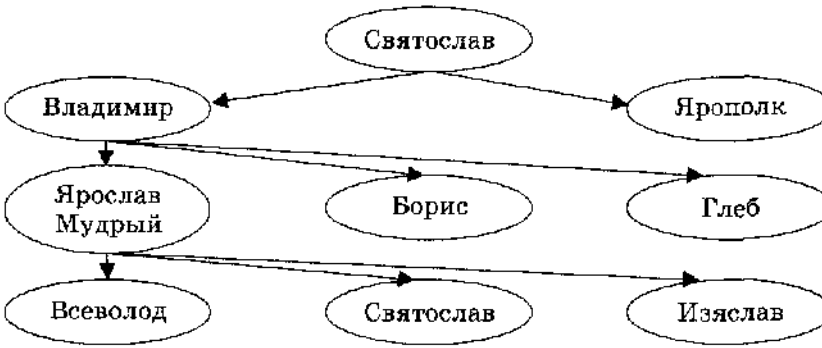


Рис. 5.11. Информационная модель — генеалогическое дерево Рюриковичей (X–XI века)

Контрольные вопросы

1. Приведите примеры материальных моделей.
2. Приведите примеры информационных моделей различных видов.

Задания для самостоятельного выполнения

5.1. *Задание с развернутым ответом.* Построить фрагмент иерархической модели животного мира.

5.2. *Задание с развернутым ответом.* Построить фрагмент модели генеалогического дерева вашей семьи.

5.2.3. Формализация и визуализация информационных моделей

На протяжении своей истории человечество использовало различные способы и инструменты для создания информационных моделей. Эти способы постоянно совершенствовались. Так, первые информационные модели создавались в форме наскальных рисунков. В настоящее время информационные модели обычно строятся и исследуются с использованием современных компьютерных технологий.

Описательные информационные модели. Описательные информационные модели отображают объекты, процессы и явления качественно, т. е. не используя количественных характеристик. Описательные информационные модели обычно строятся с использованием естественных языков и рисунков.

1.2.2. Знаковые системы Информатика и ИКТ-8

В истории науки известны многочисленные описательные информационные модели. Так, гелиоцентрическая модель мира Коперника на естественном языке формулировалась следующим образом:

- Земля вращается вокруг Солнца, а Луна вращается вокруг Земли;
- все планеты вращаются вокруг Солнца.

Однако более наглядно ее представление в виде рисунка (рис. 5.12).

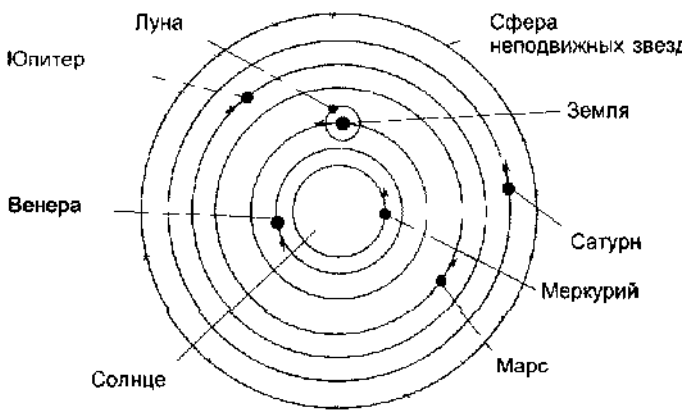


Рис. 5.12. Описательная модель гелиоцентрической системы мира Коперника

В физике явление электростатического взаимодействия двух зарядов описывается на естественном языке так: «Два одноименных заряда отталкиваются, а два разноименных притягиваются».

Для наглядности можно нарисовать линии напряженности электростатического поля и эквипотенциальные поверхности (рис. 5.13).

Электростатика Физика-8

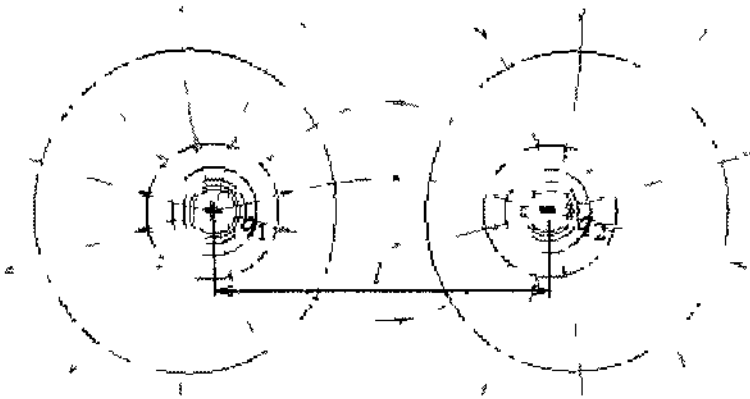


Рис. 5.13. Описательная модель взаимодействия электрических зарядов

В химии строение молекулы воды можно качественно описать на естественном языке: «Молекула воды состоит из атома кислорода и двух атомов водорода».

Для наглядности строение молекулы можно нарисовать (рис. 5.14).

Рис. 5.14 Описательная модель молекулы воды

Химия-8

Формализация информационных моделей. С помощью формальных языков строятся формальные информационные модели. Математика является широко используемым формальным языком. С использованием математических понятий и формул строятся **математические модели.** Математика включает различные формальные языки, с некоторыми из которых (алгебра и геометрия) вы знакомитесь в школе.

В естественных науках (физике, химии и др.) строятся формальные модели явлений и процессов. В большинстве случаев для этого применяется универсальный математический язык алгебраических формул. Однако в некоторых случаях используются специализированные формальные языки (в химии язык химических формул, в музыке нотная грамота и т. д.).

Ньютон формализовал гелиоцентрическую систему мира, открыв закон всемирного тяготения и законы механики и записав их в виде формул.

$$F = \gamma \frac{m_1 \, m_2}{r^2}$$

$$\vec{F} = m \, \vec{a}$$

В электростатике взаимодействие электрических зарядов описывается формулой закона Кулона.

$$F = k \cdot \frac{q_1 \cdot q_2}{r^2}$$

В химии строение молекулы воды описывается химической формулой.

$$H_2O$$

Процесс построения информационных моделей с помощью формальных языков называется **формализацией**.

В процессе познания окружающего мира человечество постоянно использует моделирование и формализацию. При изучении нового объекта сначала обычно строится его описательная информационная модель на естественном языке, затем она формализуется, т. е. выражается с использованием формальных языков.

Визуализация формальных моделей. В процессе исследования формальных моделей часто производится их визуализация. Для визуализации алгоритмов используются блок-схемы, пространственных соотношений между объектами — чертежи, моделей электрических цепей — электрические схемы. При визуализации формальных моделей с помощью анимации может отображаться динамика процесса, производиться построение графиков изменения величин и т. д.

В настоящее время широкое распространение получили **компьютерные интерактивные визуальные модели**. В таких моделях исследователь может менять начальные условия и параметры протекания процессов и наблюдать изменения в поведении модели.

В качестве примера визуализации формальной модели можно привести компьютерную визуальную интерактивную модель гидравлической машины (рис. 5.15).

В компьютерном эксперименте можно изменять площади поршней S_1 и S_2 и массы грузов m_1 и m_2 в обоих коленах и вывести формулу соотношения между площадями поршней и действующими на них силами.

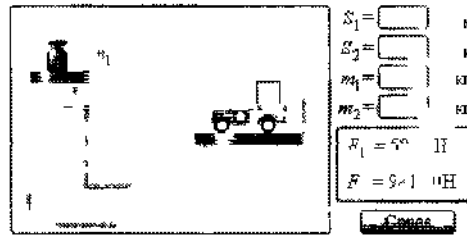


Рис. 5.15. Компьютерная визуальная интерактивная модель гидравлической машины

Контрольные вопросы

1. Приведите примеры описательных информационных моделей.
2. Приведите примеры формализованных информационных моделей.

Задания для самостоятельного выполнения

5.3. *Практическое задание.* Ознакомиться с визуализированными интерактивными моделями из различных предметных областей в Интернете по адресу http //www college ru

5.3. Основные этапы разработки и исследования моделей на компьютере

Использование компьютера для исследования информационных моделей различных объектов и систем позволяет изучить их изменения в зависимости от значений тех или иных свойств. Процесс разработки моделей и их исследование на компьютере можно разделить на несколько основных этапов.

Описательная информационная модель. На первом этапе исследования объекта или процесса обычно строится описательная информационная модель. Такая модель выделяет существенные, с точки зрения целей проводимого исследования, свойства объекта, а несущественными свойствами пренебрегает.

Формализованная модель. На втором этапе создается формализованная модель, т. е. описательная информационная модель записывается с помощью какого-либо формального языка. В такой модели с помощью формул, уравнений или неравенств фиксируются формальные соотношения между начальными и конечными значениями свойств объектов, а также накладываются ограничения на допустимые значения этих свойств.

Однако далеко не всегда удается найти формулы, явно выражающие искомые величины через исходные данные. В таких случаях используются приближенные математические методы, позволяющие получать результаты с заданной точностью.

Компьютерная модель. На третьем этапе необходимо формализованную информационную модель преобразовать в компьютерную модель, т. е. выразить ее на понятном для компьютера языке. Существуют различные пути построения компьютерных моделей, в том числе:

- создание компьютерной модели в форме проекта на одном из языков программирования;
- построение компьютерной модели с использованием электронных таблиц или других приложений: систем компьютерного черчения, систем управления базами данных, геоинформационных систем и т. д.

В процессе создания компьютерной модели полезно разработать удобный графический интерфейс, который позволит визуализировать формальную модель, а также реализовать интерактивный диалог человека с компьютером на этапе исследования модели.

Компьютерный эксперимент. Четвертый этап исследования информационной модели состоит в проведении компьютерного эксперимента. Если компьютерная модель существует в виде проекта на одном из языков программирования, ее нужно запустить на выполнение, ввести исходные данные и получить результаты.

Если компьютерная модель исследуется в приложении, например в электроиных таблицах, то можно построить диаграмму или график, провести сортировку и поиск данных или использовать другие специализированные методы обработки данных.

При использовании готовой компьютерной визуальной интерактивной модели необходимо ввести исходные данные, запустить модель на выполнение и наблюдать изменение объекта и характеризующих его величин.

В виртуальных компьютерных лабораториях можно проводить эксперименты с реальными объектами. Для этого к компьютеру присоединяются датчики измерения физических параметров (температуры, давления, силы и др.), данные измерений передаются в компьютер и обрабатываются специальной программой. Результаты эксперимента в виде таблиц, графиков и диаграмм отображаются на экране монитора и могут быть распечатаны.

Анализ полученных результатов и корректировка исследуемой модели. Пятый этап состоит в анализе полученных результатов и корректировке исследуемой модели. В случае расхождения результатов, полученных при исследовании информационной модели, с измеряемыми параметрами реальных объектов можно сделать вывод, что на предыдущих этапах построения модели были допущены ошибки или неточности.

Например, при построении описательной качественной модели могут быть неправильно отобраны существенные свойства объектов, в процессе формализации могут быть допущены ошибки в формулах и т. д. В этих случаях необходимо провести корректировку модели, причем уточнение модели может проводиться многократно, пока анализ результатов не покажет их соответствие изучаемому объекту.

Контрольные вопросы

1. Перечислите и опишите основные этапы разработки и исследования моделей на компьютере.
2. Какие программные средства обычно используются для создания компьютерных моделей?

5.4. Построение и исследование физических моделей

Рассмотрим процесс построения и исследования модели на конкретном примере движения тела, брошенного под углом к горизонту.

Физика-9

Содержательная постановка задачи «Бросание мячика в площадку». В процессе тренировок теннисистов используются *автоматы по бросанию мячика в определенное место площадки*. Требуется задать автомату необходимую скорость и угол бросания мячика для попадания в площадку определенной длины, находящуюся на известном расстоянии (рис. 5.16).

Качественная описательная модель. Сначала построим качественную описательную модель процесса движения тела с использованием физических объектов, понятий и законов, т. е. в данном случае идеализированную модель движения объекта. Из условия задачи можно сформулировать следующие основные предположения:

- мячик мал по сравнению с Землей, поэтому его можно считать материальной точкой;
- изменение высоты мячика мало, поэтому ускорение свободного падения можно считать постоянной величиной $g = 9{,}8$ м/с и движение по оси Y можно считать *равноускоренным*;
- скорость бросания тела мала, поэтому сопротивлением воздуха можно пренебречь и движение по оси X можно считать равномерным.

Формальная модель. Для формализации модели используем известные из курса физики формулы равномерного и равноускоренного движения. При заданных начальной скорости v_0 и угле бросания α значения координат дальности полета x и высоты y от времени можно описать следующими формулами:

$$\begin{aligned} x &= v_0 \cdot \cos\alpha \cdot t, \\ y &= v_0 \cdot \sin\alpha \cdot t - g \cdot t^2/2. \end{aligned} \qquad (5.1)$$

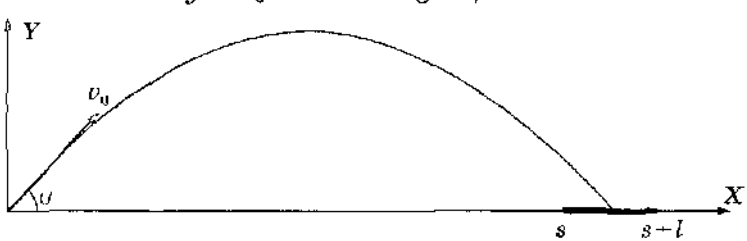


Рис. 5.16. Бросание мячика в площадку

Площадка расположена на поверхности Земли, поэтому из второй формулы (5.1) можно выразить время, которое понадобится мячику, чтобы достичь площадки:

$$v_0 \cdot \sin\alpha \cdot t - g \cdot t^2/2 = 0,$$
$$t \cdot (v_0 \cdot \sin\alpha - g \cdot t/2) = 0.$$

Значение времени $t = 0$ не имеет физического смысла, поэтому:

$$v_0 \cdot \sin\alpha - g \cdot t/2 = 0,$$
$$t = (2 \cdot v_0 \cdot \sin\alpha)/g.$$

Подставим полученное выражение для времени в формулу для вычисления координаты x:

$$x = (v_0 \cdot \cos\alpha \cdot 2 \cdot v_0 \cdot \sin\alpha)/g = (v_0^2 \cdot \sin 2\alpha)/g. \qquad (5.2)$$

Формализуем теперь условие попадание мячика в площадку. Пусть площадка расположена на расстоянии s и имеет длину l (см. рис. 5.16). Тогда попадание произойдет, если значение координаты x мячика будет удовлетворять условию в форме неравенства:

$$s \le x \le s + l.$$

Если $x < s$, то это означает «недолет», а если $x > s + l$, то это означает «перелет».

Компьютерная модель движения тела. На основе формальной модели, описывающей движение тела, брошенного под углом к горизонту, создадим компьютерную модель с использованием системы программирования. Используем систему объектно-ориентированного программирования Visual Basic 2005, так как она позволяет визуализировать траектории движения тела.

На основе данной формальной модели создадим также компьютерную модель с использованием электронных таблиц Microsoft Excel или OpenOffice.org Calc. Визуализируем траектории движения тела с использованием диаграммы типа *График*.

Контрольные вопросы

1. Какие основные предположения можно сделать при построении качественной описательной модели бросания мячика под углом к горизонту?
2. Чем отличается компьютерная модель от формальной модели?

5.5. Приближенное решение уравнений

Алгебра-9

На языке алгебры формальные модели записываются с помощью уравнений, точное решение которых основывается на поиске равносильных преобразований алгебраических выражений, позволяющих выразить переменную величину с помощью формулы.

Точные решения существуют только для некоторых уравнений определенного вида (линейные, квадратные, тригонометрические и др.), поэтому для большинства уравнений приходится использовать методы приближенного решения с заданной точностью (графические или численные).

Например, нельзя найти корень уравнения $x^8 - \sin x = 0$ путем равносильных алгебраических преобразований. Однако такие уравнения можно решать приближенно графическими и численными методами.

Построение графиков функций может использоваться для грубо приближенного решения уравнений. Для уравнений вида $f(x) = 0$, где $f(x)$ — некоторая непрерывная функция, корень (или корни) этого уравнения является точкой (или точками) пересечения графика функции с осью X.

Графическое решение таких уравнений можно осуществить путем построения компьютерных моделей:

- построением графика функции в системе объектно-ориентированного программирования Visual Basic;
- в электронных таблицах Microsoft Excel или OpenOffice.org Calc путем построения диаграммы типа *График*.

Контрольные вопросы

1. В каких случаях используются приближенные (графические) методы решения уравнений?

5.6. Экспертные системы распознавания химических веществ

Химия-9

Экспертные системы. Профессиональные экспертные системы достаточно широко используются в различных областях

науки и техники. Такие системы позволяют автоматически выявлять причины сбоев в работе сложных технических систем (например, космических кораблей), распознавать личность человека по его отпечаткам пальцев или радужной оболочке глаза и т. д.

Основная задача экспертных систем — распознавать объекты или состояния объекта. В процессе обучения встречается достаточно много учебных ситуаций, когда вам приходится выступать в роли эксперта и необходимо распознать тот или иной объект. Обычно такие задачи выполняются методом проб и ошибок, без осознания и фиксации стратегии поиска.

Создание учебной экспертной системы позволяет осознать и зафиксировать последовательность рассуждений или действий, которая приводит к распознаванию того или иного объекта среди некоторой совокупности.

Лабораторная работа по неорганической химии «Распознавание химических удобрений». Даны удобрения, химические реактивы и справочная таблица по взаимодействию удобрении с некоторыми реактивами (табл. 5.1) и предлагается распознать каждое из удобрений.

Таблица 5.1. **Свойства удобрений**

№	Внешний вид	Взаимодействие раствора удобрения с			Удобрение (результат распознавания)
		H_2SO_4	$BaCl_2$	раствором щелочи	
1	Белая, кристаллическая масса или гранулы	Выделяется бурый газ	—	Ощущается запах аммиака	Аммиачная селитра
2	Крупные бесцветные кристаллы	Выделяется бурый газ	Небольшое помутнение раствора	—	Натриевая селитра
3	Мелкие светло-серые кристаллы	—	Выпадает белый осадок	Ощущается запах аммиака	Сульфат аммония
4	Светло-серый порошок или гранулы	—	Выпадает белый осадок	—	Суперфосфат
5	Розовые кристаллы	—	—	—	Сильвинит
6	Бесцветные кристаллы	—	—	—	Калийная соль

Формальная модель экспертной системы «Распознавание удобрений». Экспертная система может быть представлена в виде алгоритма, состоящего из последовательности шагов с использованием алгоритмической структуры «ветвление». Можно построить различные алгоритмы поиска, однако необходимо стремиться к выбору оптимальной стратегии распознавания (достижения цели за минимальное число шагов). Такая стратегия будет реализована, если каждый шаг будет максимально уменьшать неопределенность (нести максимальное количество информации).

1.3.1. Количество информации как мера уменьшения неопределенности знания **Информатика и ИКТ-8**

Построим алгоритм (рис. 5.17), в котором на первом шаге разделим шесть веществ на две группы по условию *При взаимодействии с H_2SO_4 выделяется бурый газ.* Если условие:

- выполняется, то это вещества первой группы под номерами 1 и 2;
- не выполняется, то это вещества второй группы под номерами 3, 4, 5 и 6.

Для идентификации веществ первой группы достаточно проверить справедливость условия *При взаимодействии с раствором щелочи ощущается запах аммиака.* Если условие:

- выполняется, то это вещество «1. Аммиачная селитра»;
- не выполняется, то это вещество «2. Натриевая селитра».

Для идентификации веществ второй группы сначала необходимо проверить справедливость условия *При взаимодействии с $BaCl_2$ выпадает белый осадок* Если условие:

- выполняется, то это вещества 3 и 4;
- не выполняется, то это вещества 5 и 6.

Для идентификации веществ 3 и 4 достаточно проверить справедливость условия *При взаимодействии с раствором щелочи ощущается запах аммиака.* Если условие:

- выполняется, то это вещество «3. Сульфат аммония»;
- не выполняется, то это вещество «4. Суперфосфат».

Для идентификации веществ 5 и 6 достаточно проверить справедливость условия *Внешний вид — розовые кристаллы.* Если условие:

- выполняется, то это вещество «5. Сильвинит»;
- не выполняется, то это вещество «6. Калийная соль».

Целесообразно представить иерархическую модель экспертной системы в виде блок-схемы (рис. 5.17).

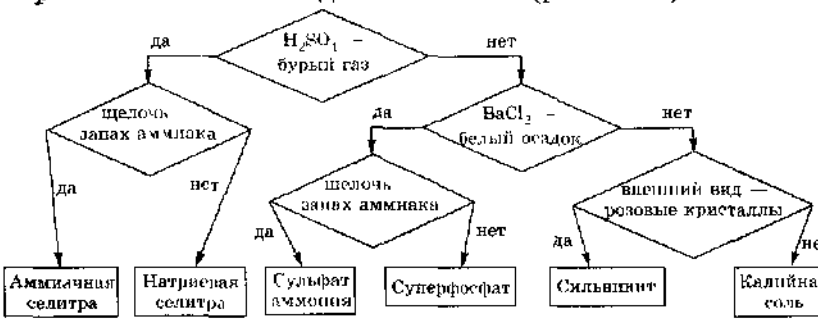


Рис. 5.17. Блок-схема экспертной системы «Распознавание удобрений»

Компьютерная модель экспертной системы на языке Visual Basic 2005. Создадим экспертную систему распознавания удобрений с использованием языка Visual Basic. Экспертная система будет задавать пользователю серии вопросов о результатах взаимодействия вещества с кислотой, щелочью и солью или о внешнем виде удобрений. Пользователь будет отвечать *Да* или *Нет* (на основании опытов или теоретических знаний). В результате нескольких серий вопросов будут определены названия всех удобрений.

При разработке сложного алгоритма целесообразно стараться выделить в нем последовательности действий, которые выполняют решение каких-либо подзадач и могут вызываться из основного алгоритма. Такие алгоритмы называются **вспомогательными** и в алгоритмических языках программирования реализуются в форме **подпрограмм**, которые вызываются из основной программы.

В объектно-ориентированном языке программирования Visual Basic 2005 вспомогательные алгоритмы реализуются с помощью **общих процедур**. Каждой общей процедуре дается уникальное название — **имя процедуры**. Запись общей процедуры производится следующим образом:

```
Sub ИмяПроцедуры(...)
программный код
End Sub
```

Запуск общих процедур не связывается с какими-либо событиями, а реализуется путем вызова по имени из других процедур.

Контрольные вопросы

1. Является ли единственным приведенный алгоритм экспертной системы распознания удобрений? Какие еще варианты алгоритма экспертной системы вы можете предложить?

5.7. Информационные модели управления объектами

В процессе функционирования сложных систем (биологических, технических и т. д.) важную роль играют информационные процессы управления.

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).

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.

Системы управления без обратной связи. В системах управления без обратной связи не учитывается состояние управляемого объекта и обеспечивается управление только по прямому каналу (от управляющего объекта к управляемому объекту). Информационную модель системы управления без обратной связи можно наглядно представить с помощью схемы на рис. 5.18.

В качестве примера системы управления без обратной связи рассмотрим процесс записи информации на гибкий диск, в котором контроллер дисковода (управляющий объ-

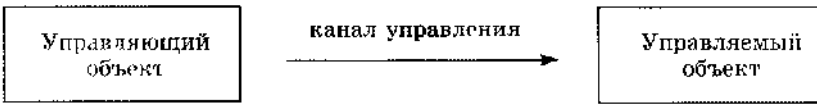


Рис. 5.18. Система управления без обратной связи

ект) изменяет положение магнитной головки дисковода (управляемый объект).

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 дискеты. Поэтому контроллер дисковода для установки магнитной головки над требуемой магнитной дорожкой дискеты просто перемещает ее вдоль радиуса дискеты.

Системы управления с обратной связью. В системах управления с обратной связью управляющий объект по прямому каналу управления производит необходимые действия над объектом управления, а по каналу обратной связи получает информацию о его реальных параметрах. Это позволяет осуществлять управление с гораздо большей точностью.

Информационную модель системы управления с обратной связью можно наглядно представить с помощью схемы на рис. 5.19.

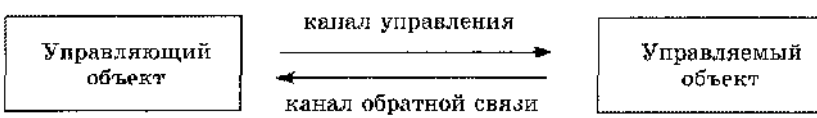


Рис. 5.19. Система управления с обратной связью

Примером системы управления с автоматической обратной связью является запись информации на жесткий диск. При записи информации на жесткий диск требуется особая точность установки магнитных головок, так как на рабочей поверхности пластин имеются десятки тысяч дорожек и необходимо учитывать их механические деформации (например, в результате изменения температуры). Контроллер жесткого диска (управляющий объект) по каналу обратной связи постоянно получает информацию о реальном положе-

нии магнитных головок (управляемый объект), а по каналу управления выставляет головки над поверхностью пластин с большой точностью.

Контрольные вопросы

1. Приведите примеры систем управления без обратной связи и с обратной связью.
2. В чем состоит различие между системами управления без обратной связи и системами управления с обратной связью?

Практические работы компьютерного практикума, рекомендуемые для выполнения в процессе изучения главы 5

Компьютерный практикум

5.1. Проект «Бросание мячика в площадку».

5.2. Проект «Графическое решение уравнения».

5.3. Проект «Распознавание удобрений».

5.4. Проект «Модели систем управления».

Глава 6

Информатизация общества

6.1. Информационное общество

Доиндустриальное общество. Человеческое общество по мере своего развития прошло этапы овладения веществом, затем энергией, и, наконец, информацией. В первобытно-общинном, рабовладельческом и феодальном обществах (в основе существования которых лежало ремесло) деятельность общества в целом и каждого человека в отдельности была направлена, в первую очередь, на овладение веществом.

На заре цивилизации (десятки тысяч лет до нашей эры) люди научились изготавливать простые орудия труда и охоты (каменный топор, стрелы и т. д.), в античности появились первые механизмы (рычаг и др.) и средства передвижения (колесницы, корабли), в средние века были изобретены первые сложные орудия труда и механизмы (ткацкий станок, часы).

Овладение энергией находилось в этот период на начальной ступени, в качестве источников энергии использовались Солнце, вода, огонь, ветер и мускульная сила человека.

С самого начала человеческой истории возникла потребность передачи и хранения информации. Для передачи информации сначала использовался язык жестов, а затем человеческая речь. Для хранения информации стали использоваться наскальные рисунки, а в IV тысячелетии до нашей эры появилась письменность и первые носители информации (шумерские глиняные таблички и египетские папирусы). История создания устройств для обработки числовой информации начиналась также еще в древности — с абака (счетной доски, являющейся прообразом счетов).

Индустриальное общество. Начиная примерно с XVII века, в процессе становления машинного производства, на первый план выходит проблема овладения энергией (машины и станки необходимо было приводить в движение). Сначала совершенствовались способы овладения энергией ветра и воды (ветряные мельницы и водяные колеса), а затем человечество овладело тепловой энергией (в середине XVIII века была изобретена паровая машина, а в конце XIX века — двигатель внутреннего сгорания).

В конце XIX века началось овладение электрической энергией, были изобретены электрогенератор и электродвигатель. И наконец, в середине XX века человечество овладело атомной энергией. В 1954 году в СССР была пущена в эксплуатацию первая атомная электростанция.

Овладение энергией позволило перейти к массовому машинному производству потребительских товаров, было создано индустриальное общество. Основными показателями развитости индустриального общества являлись количественные показатели, т. е. сколько было добыто угля и нефти, сколько произведено станков и т. д.

В этот период происходили также существенные изменения в способах хранения и передачи информации. В середине XV века было изобретено книгопечатание, что позволило сделать информацию доступной для гораздо большего количества людей. С конца XIX века для передачи информации на дальние расстояния по проводам стали широко использоваться телеграф и телефон, а в XX веке — электромагнитные волны (радио, телевидение).

Информационное общество. Первой попыткой автоматизированной обработки информации стало создание Чарльзом Бэббиджем в середине XIX века механической цифровой Аналитической машины. Однако лишь с середины XX века, с момента появления электронных устройств обработки и хранения информации (ЭВМ, а затем персонального компьютера), начался постепенный переход от индустриального общества к информационному.

В информационном обществе главным ресурсом является информация, именно на основе владения информацией о самых различных процессах и явлениях можно эффективно и оптимально строить любую деятельность.

Важно не только произвести большое количество продукции, но и сделать нужную продукцию в определенное время и с определенными затратами. В информационном

обществе повышается не только качество потребления, но и качество производства, человек, использующий информационные технологии, имеет лучшие условия труда, а труд становится творческим и интеллектуальным.

В качестве критериев развитости информационного общества можно выбрать три: наличие компьютеров, уровень развития компьютерных сетей и количество населения, занятого в информационной сфере, а также использующего информационные и коммуникационные технологии в своей повседневной деятельности.

Производство компьютеров. Первые электронно-вычислительные машины (ЭВМ), которые могли автоматически по заданной программе обрабатывать большие объемы информации, были созданы в 1946 году в США (ЭНИАК) и в 1950 году в СССР (МЭСМ). В 40–60-х годах XX века производство ЭВМ измерялось единицами, десятками и, в лучшем случае, сотнями штук. ЭВМ были очень дорогими и очень большими (занимали громадные залы) и поэтому оставались недоступными для массового потребителя.

Массовое производство сравнительно недорогих персональных компьютеров началось с середины 1970-х годов с компьютера Apple II (с этого компьютера начала свое существование фирма Apple). Количество произведенных персональных компьютеров начало составлять десятки тысяч в год, что по тем временам было колоссальным достижением.

В начале 1980-х годов приступила к массовому производству персональных компьютеров корпорация IBM (компьютеры так и назывались IBM Personal Computer — IBM PC). Достаточно скоро IBM-совместимые компьютеры стали выпускать многие фирмы, и их производство достигло сотен тысяч в год. Ежегодное производство персональных компьютеров постоянно росло и в 2007 году превысило 200 миллионов.

Персональный компьютер постоянно совершенствовался, его производительность возросла на три порядка, при этом, что очень важно, цена даже снизилась. Персональный компьютер стал доступен массовому потребителю, и теперь в развитых странах мира компьютер имеется на большинстве рабочих мест и в большинстве семей.

Компьютерные сети. В настоящее время существенной тенденцией в информатизации общества является переход от использования компьютеров в автономном режиме к использованию их в информационных сетях.

Информационные сети создают реальную возможность быстрого и удобного доступа пользователя ко всей информации, накопленной человечеством за свою историю. Электронная почта и телеконференции, поиск информации во Всемирной паутине и в файловых архивах, интерактивное общение, прослушивание радиостанций и просмотр телевизионных программ, покупки в Интернет-магазинах стали повседневной практикой многих пользователей компьютеров в развитых странах.

Развитие глобальных компьютерных сетей началось в 80-е годы прошлого века. В 1981 году в сети Интернет насчитывалось лишь 213 компьютеров, к концу 1980-х годов количество подключенных к сети компьютеров возросло до 150 тысяч, наиболее быстрый рост происходил за последнее десятилетие, и к началу 2007 года их количество превысило 400 миллионов.

Статистические данные о росте Интернета http://www.isc.org

По количеству имеющихся серверов Интернета можно судить о степени информатизации отдельных стран. Наибольшее количество серверов зарегистрировано в доменах административного типа (около 253 миллионов серверов), значительная часть которых зарегистрирована в США, на втором месте, с большим отставанием, Япония (31 миллион серверов), Россия занимает в этом списке 20-е место (около 2,4 миллиона серверов).

Количество постоянных пользователей ресурсов и услуг Интернета во всех странах мира составляет примерно один миллиард человек. В России количество пользователей растет быстрыми темпами, и в 2007 году составило примерно 25 миллионов человек.

Население, занятое в информационной сфере. По данным ООН, в 90-е годы XX века количество работников, занятых в информационной сфере (для которых обработка информации является основной производственной функцией), возросло примерно на 25%, тогда как количество занятых в сельском хозяйстве и промышленности сократилось, соответственно, на 10 и 15%.

Компьютеры и информационные технологии интенсивно проникают и в сферу материального производства. Инженер, фермер, специалисты других традиционных про-

фессий все чаще имеют на своем рабочем месте компьютер и используют информационные и коммуникационные технологии в своей профессиональной деятельности.

С развитием коммуникационных технологий и мобильной связи все большее количество людей осуществляют свою производственную деятельность дистанционно, т. е. работая дома, а не в офисе (в США более 10 миллионов человек). Все большее распространение получает дистанционное образование и поиск работы через Интернет. В 2000 году оборот мирового рынка информационных и коммуникационных технологий составил около 1 триллиона долларов. При этом на закупку аппаратных средств было потрачено менее половины этой суммы, большая часть была вложена в разработку программного обеспечения, проектирование компьютерных сетей и т. д.

Информационное общество — это общество, в котором большая часть населения занята получением, переработкой, передачей и хранением информации.

Курс информатики и информационно-коммуникационных технологий (ИКТ) играет особую роль в эпоху перехода от индустриального общества к информационному, так как готовит выпускников школы к жизни и деятельности в информационном обществе.

Контрольные вопросы

1. Какую роль играли вещество, энергия и информация на различных этапах развития общества?
2. По каким основным параметрам можно судить о степени развитости информационного общества и почему?
3. Как изменяется содержание жизни и деятельности людей в процессе перехода от индустриального общества к информационному?

Интернет

Задания для самостоятельного выполнения

6.1. Найдите в Интернете данные о росте количества пользователей и серверов.

6.2. Информационная культура

Количество информации в современном обществе стремительно нарастает, человек оказывается погруженным в море информации. Для того чтобы в этом море «не утонуть», необходимо обладать **информационной культурой**, т. е. знаниями и умениями в области информационных и коммуникационных технологий, а также быть знакомым с юридическими и этическими нормами в этой сфере.

Процесс информатизации общества меняет традиционные взгляды на перечень умений и навыков, необходимых для социальной адаптации. Возьмем традиционный навык письма. На заре цивилизации (Шумер, Египет), в античном мире (Эллада, Римская империя и др.) и в средние века (до изобретения книгопечатания) навык каллиграфического письма был залогом успешного продвижения по социальной лестнице. В индустриальном обществе (до изобретения персональных компьютеров) навыки письма ручкой также были необходимы для любого члена общества. В настоящее время, на пороге информационного общества социальная значимость навыка письма ручкой снижается и, наоборот, социальная значимость навыков ввода информации с помощью клавиатуры и *работы с графическим интерфейсом приложений с помощью* мыши возрастает.

Создание и редактирование документов с помощью компьютера, т. е. овладение **офисными информационными технологиями**, становится в информационном обществе *социально необходимым умением — достаточно просмотреть* объявления о приеме на работу.

Современные информационные технологии позволяют включать в состав документа любые мультимедийные объ-

екты (графику, звук, анимацию, видео). Дома вы можете привести в порядок фотоархив семьи, отсканировав старые фотографии и поместив их в упорядоченном виде в компьютерный фотоальбом, в процессе обучения вы можете подготовить реферат с иллюстрациями, в процессе профессиональной деятельности — подготовить компьютерную презентацию о деятельности вашей фирмы. **Умение работать с мультимедиа документами, создавать компьютерные презентации** становится важным в информационном обществе.

В современном информационном обществе вряд ли необходимы навыки традиционного черчения на ватмане. Вместо этого полезно получить первоначальное представление о ***назначении и возможностях компьютерных систем автоматизированного проектирования* (САПР)**. Такие системы позволят вам быстро рассмотреть различные варианты планировки интерьера дома или квартиры, создать чертеж или схему.

Использование **электронных таблиц** сделает более простым и наглядным исследование и построение графиков функций в математике, планирование и ведение домашнего бюджета, автоматизирует заполнение многочисленных квитанций оплаты за квартиру, электроэнергию и телефон.

Необходимость упорядочить информацию, например, о людях, с которыми вы контактируете, требует использования записной книжки. Однако часто удобнее использовать для хранения такой информации компьютерную базу данных «Записная книжка». При поиске информации в современной библиотеке или в Интернете необходимо иметь навыки поиска информации в базах данных. В информационном обществе очень полезным является умение создавать **базы данных**, а также вести в них поиск данных.

Квалифицированный пользователь компьютера может на основе использования **языков визуального объектно-ориентированного программирования** создавать необходимые ему специализированные приложения, создавать и исследовать модели различных объектов и процессов, разрабатывать динамические Web-сайты и т. д.

Современному человеку необходимо овладеть **коммуникативной культурой**, т. е. умениями создавать и посылать электронные письма, находить нужную информацию во Всемирной паутине или в файловых архивах, участвовать в чатах и т. д. Необходимым условием успешной профессиональной деятельности становится создание и публикация в

Интернете Web-сайтов с информацией о деятельности организации или предприятия.

Информационная культура состоит не только в овладении определенным комплексом знаний и умений в области информационных и коммуникационных технологий, но и предполагает знание и **соблюдение юридических и этических норм и правил.** Законы запрещают использование пиратского компьютерного обеспечения и пропаганду насилия, наркотиков и порнографии в Интернете. Общение с помощью электронной почты или в чатах, участие в телеконференциях предполагают соблюдение определенных правил: отвечать на письма и не рассылать знакомым и незнакомым людям многочисленные рекламные сообщения (спам), не отклоняться от темы обсуждения в телеконференциях и чатах и т. д.

Контрольные вопросы

1. Каковы основные компоненты информационной культуры, которые необходимы человеку для жизни в информационном обществе?

6.3. Перспективы развития информационных и коммуникационных технологий (ИКТ)

Развитие новых информационных и коммуникационных технологий имеет общие законы. Большинство новых технологий проходит в процессе своего развития пять этапов, однако некоторые технологии развиваются очень быстро и «пропускают» некоторые этапы, другие же, наоборот, периодически возвращаются на начальный этап развития.

Лучше всего этапы развития ИКТ можно представить в графической форме. По оси абсцисс отложим время в этапах, однако следует учитывать, что различные технологии проходят этапы своего развития за различное время (от 2 до 10 лет), т. е. шкала оси времени для разных технологий неодинакова. По оси ординат отложим уровень оценки технологии обществом, что носит довольно субъективный характер (рис. 6.1).

Первый этап. «Восход надежд», когда появляются теоретические обоснования и первые экспериментальные реализации новой технологии. Обществу кажется, что данная новая технология разрешит многие проблемы.

Примером такой технологии являются нанотехнологии. Действительно, современные транзисторы уже имеют размеры в несколько десятков атомов, и дальнейшая миниатюризация должна привести размеры транзисторов к размерам одного атома или молекулы. Конкретной технологической реализацией нанотехнологий являются углеродные нанотрубки, которые уже используются для создания первых экспериментальных нанотранзисторов.

Второй этап. «Пик завышенных ожиданий», когда разработчики и средства массовой информации внушают обществу высокую ценность новой технологии и эффективность первых промышленных образцов.

Примером такой технологии является IP-телевидение (трансляция телеканалов через Интернет). Однако у этой технологии много альтернативных конкурентов (эфирное, кабельное и спутниковое телевидение), недостаточное качество изображения (периодическое «рассыпание») и сравнительно высокая цена.

Третий этап. «Котловина разочарований», когда широко разрекламированная новая технология теряет свою привлекательность в глазах конечных потребителей. В процессе использования первых массовых экземпляров новой технологии выявляются конструктивные недостатки.

Примером такой технологии являются Интернет-видеоконференции, когда пользователи, подключенные к широкополосному Интернету, могут в реальном времени общаться не только с помощью традиционного чата, но и слышать и видеть друг друга.

Четвертый этап. «Подъем жизнестойкости», когда на основе новых исследований оптимизируется технологический процесс и начинается массовое серийное производство.

Примером такой технологии является платформа Microsoft .NET, которая позволяет создавать приложения с использованием различных языков объектно-ориентированного программирования. Важно, что на этом этапе технологию .NET поддерживают и другие фирмы, разработчики систем программирования (Borland, Novell и др.).

Пятый этап. «Плато продуктивности», когда массовое серийное производство изделий по новой технологии нахо-

дит массовый устойчивый спрос потребителей и приносит стабильную прибыль производителям.

Примером такой технологии является широкополосный, т. е. высокоскоростной, доступ в Интернет. Технологии широкополосного доступа стали необходимым обычным явлением для пользователя. Необходимо отметить, что технология широкополосного доступа в Интернет может использовать различные методы и каналы связи.

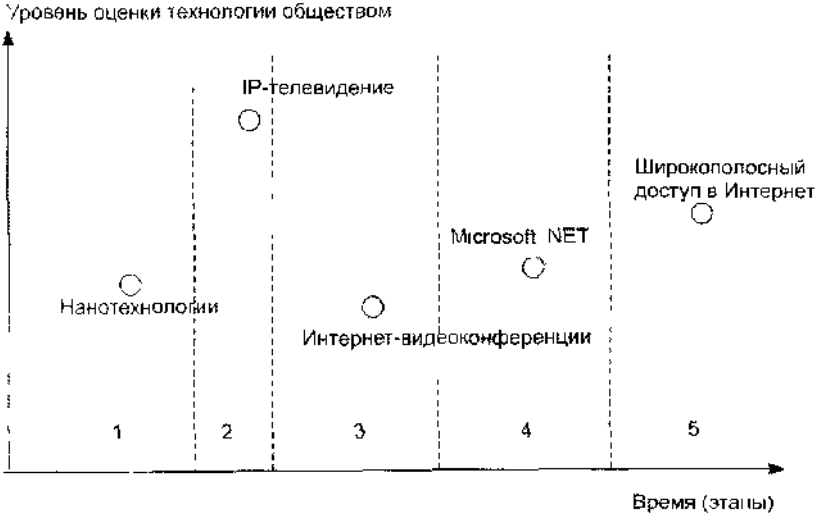


Рис. 6.1. Развитие информационных и коммуникационных технологий

Контрольные вопросы

1. Назовите технологии, соответствующие различным этапам развития ИКТ.

Практические работы к главе 1 «Кодирование и обработка графической и мультимедийной информации»

	Установить: • векторный графический редактор OpenOffice.org Draw, • растровый графический редактор GIMP; • программу разработки презентаций OpenOffice.org Impress; • редактор flash-анимации Macromedia Flash. • звуковой редактор Audacity, • программу разработки презентаций Microsoft PowerPoint	Windows-CD Дистрибутив Microsoft Office
	Установить: • векторный графический редактор OpenOffice.org Draw, • растровый графический редактор GIMP; • программу разработки презентаций OpenOffice.org Impress; • звуковой редактор Audacity, • систему захвата цифровых фото digiKam, • систему захвата и редактирование цифрового видео KINO	Linux-DVD

Практическая работа 1.1
Кодирование графической информации

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться:

- устанавливать различные графические режимы экрана монитора;
- устанавливать цвет путем задания числовых кодов интенсивностей базовых цветов палитры RGB (красного, зеленого и синего).

Задание 1. Установить графический режим экрана монитора:

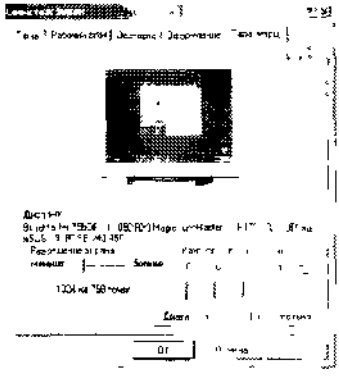
- с наиболее возможным высоким разрешением экрана;
- с наиболее возможной глубиной цвета.

Задание 2. В графическом редакторе последовательно установить цвета (см. табл. 1.3. Кодировка цветов при глубине цвета 24 бита) с использованием палитр цветов RGB, CMYK и HSB. Цвета устанавливать путем введения числовых кодов базовых цветов в соответствующие текстовые поля.

1.1.3. Палитры цветов в системах цветопередачи RGB, CMYK и HSB

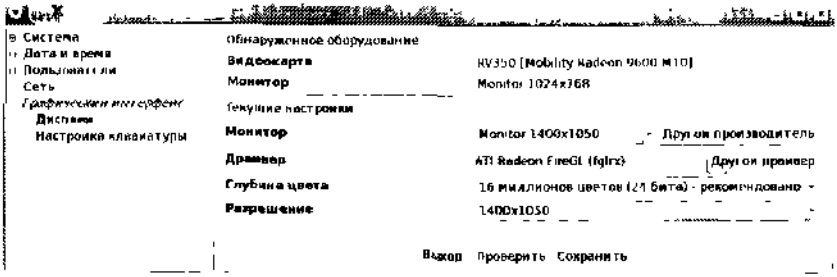
Задание 1. Установка графического режима экрана монитора в операционной системе Windows

1. В операционной системе Windows щелкнуть правой кнопкой мыши по *Рабочему столу*, появится диалоговое окно *Свойства: Экран*. Выбрать вкладку *Параметры*, которая предоставляет возможность установить графический режим экрана. Разрешающую способность экрана установить с помощью ползунка *Разрешение экрана*. Глубину цвета установить с помощью раскрывающегося списка *Качество цветопередачи*.



Установка графического режима экрана монитора в операционной системе Linux

1. В операционной системе Linux ввести команду [*Настройки-Центр управления системой*].
В левой части появившегося диалогового окна выбрать пункт *Дисплей* и в правой части диалогового окна установить разрешающую способность экрана и глубину цвета.



Задание 2. Установка цвета в графическом редакторе с использованием системы цветопередачи RGB в векторном редакторе OpenOffice.org Draw

1. В операционной системе Windows или Linux запустить интегрированное офисное приложение OpenOffice и ввести команду [*Файл-Создать-Рисунок*].

2. Нарисовать восемь одинаковых фигур (например, прямоугольников).

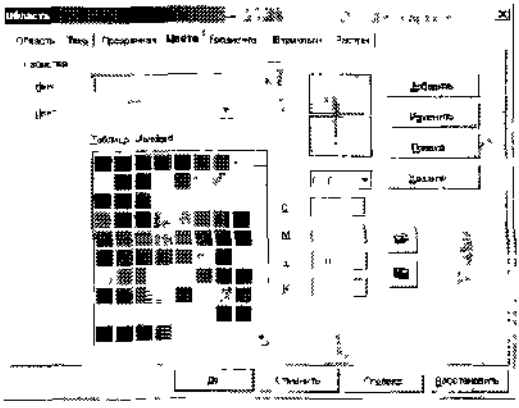
   Для каждой фигуры зададим цвет из таблицы.

3. Выделить фигуру и ввести команду [*Формат-Область...*].

   На появившемся диалоговом окне *Область* выбрать вкладку *Цвета*.

   С помощью раскрывающегося списка выбрать систему цветопередачи RGB.

   Задать цвет путем установки в полях со счетчиком интенсивностей базовых цветов.



4. Выполнить пункт 3 для остальных фигур.
Будут получены восемь фигур, закрашенные цветами, указанными в таблице.

Графический файл цвета.odg хранится в папке \\IKT9\Graph **Windows-CD**

Практическая работа 1.2

Редактирование изображений в растровом графическом редакторе

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux, сканер или цифровая камера.

Цель работы. Научиться получать цифровые растровые изображения и применять к ним различные графические эффекты.

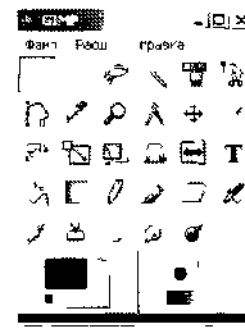
Задание. Получить с помощью сканера или цифровой камеры растровое изображение (например, обложку учебника) и преобразовать его с помощью эффектов просмотра через линзу и загнутой страницы.

Редактирование растрового изображения с помощью растрового графического редактора GIMP

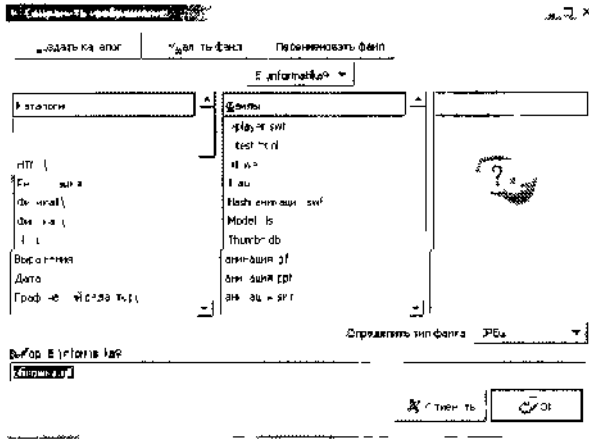
1. Получить с помощью сканера или цифровой камеры растровое изображение обложки учебника и сохранить его в файле обложка.bmp.

В случае отсутствия сканера и цифровой камеры воспользоваться готовым изображением обложка.bmp, хранящимся на Windows-CD в папке ..\IIKT9\Graph.

2. Запустить редактор GIMP командой [*Пуск Программы· GIMP GIPM2*]. *Появится Панель инструментов* графического редактора, содержащая меню редактора.



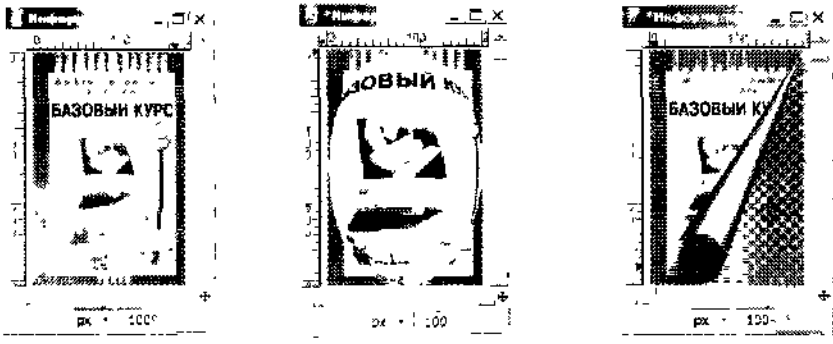
3. С помощью меню редактора ввести команду [*Файл-Открыть...*].
С помощью диалогового окна *Загрузить изображение* открыть графический файл обложка.bmp.



4. В окне изображения графического редактора откроется растровое изображение обложки учебника.

5. Для преобразования изображения обложки в вид его просмотра через линзу с помощью меню окна изображения ввести команду [*Фильтры Эффекты стекол-Линза...*].
В появившемся диалоговом окне *Эффект линзы* выбрать параметры коэффициента преломления и щелкнуть по кнопке *Да*.

6. Для преобразования изображения обложки в загнутую страницу с помощью меню окна изображения ввести команду [*Фильтры-Искажения-Загибание страницы...*]. В появившемся диалоговом окне *Эффект загнутой страницы* выбрать положение и ориентацию загиба и щелкнуть по кнопке *Да*.
7. Получим исходное изображение и полученные с использованием эффектов примерно такие растровые изображения в формате TIFF:



Файлы обложка*.tif хранятся в папке ..\IKT9\Graph **Windows-CD**

Практическая работа 1.3

Создание рисунков в векторном графическом редакторе

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться использовать различные возможности векторных редакторов: рисовать графические примитивы, линии и стрелки, вставлять растровые изображения и текст, использовать градиентную заливку, осуществлять группировку объектов, сохранять файлы в различных графических форматах.

Задание 1. Нарисовать функциональную схему компьютера, используя широкие возможности векторного графического редактора.

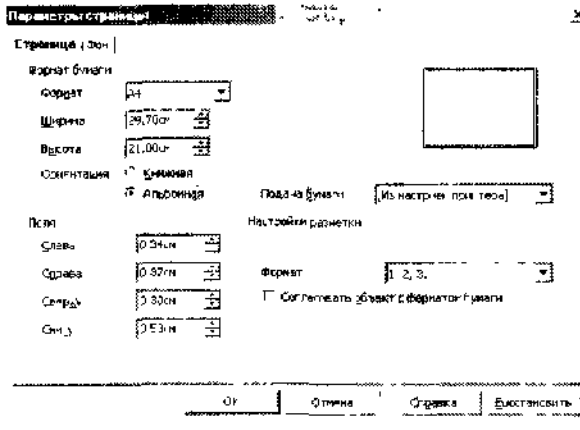
Задание 2. Сохранить полученный рисунок в векторном и растровом графических форматах.

Задание 1. Рисование функциональной схемы компьютера

1. В операционной системе Windows или Linux запустить интегрированный пакет OpenOffice.org и ввести команду [*Файл-Создать Рисунок*].

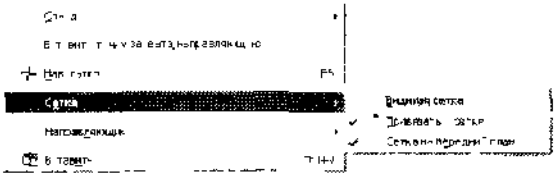
Установим размеры, поля и ориентацию области рисования.

2. Щелкнуть в области рисования правой кнопкой мыши и выбрать в контекстном меню команду [*Слайд-Параметры страницы...*].
   В появившемся окне на вкладке *Страница* выбрать формат области рисования, ее поля и ориентацию.



Для большей точности рисования привяжем рисуемые фигуры к сетке и сделаем сетку видимой.

3. Вызвать правым щелчком в области рисования ее контекстное меню и активизировать его пункты, задающие параметры сетки.



Нарисуем функциональную схему компьютера, состоящую из шести прямоугольников: *Процессор*, *Оперативная память*, *Магистраль*, *Устройства ввода*, *Долговременная память* и *Устройства вывода*.

4. На панели *Рисование*, которая находится в нижней части окна приложения, нажать кнопку *Прямоугольник* и нарисовать шесть прямоугольников в области рисования.
5. Выделить последовательно каждый прямоугольник и с помощью контекстного меню установить цвет и вид контура и заливки.

Введем в прямоугольники названия устройств компьютера.

6. На панели инструментов нажать кнопку *Текст* и ввести названия устройств компьютера.

Вставим в некоторые прямоугольники изображения соответствующих устройств компьютера.

7. Выделить прямоугольник и ввести команду [*Вставка-Изображение-Из файла...*]. В появившемся диалоговом окне *Вставить графический объект* найти в иерархической системе папок изображение процессора и щелкнуть по кнопке *ОК*.
   Аналогично вставить изображение модуля оперативной памяти.

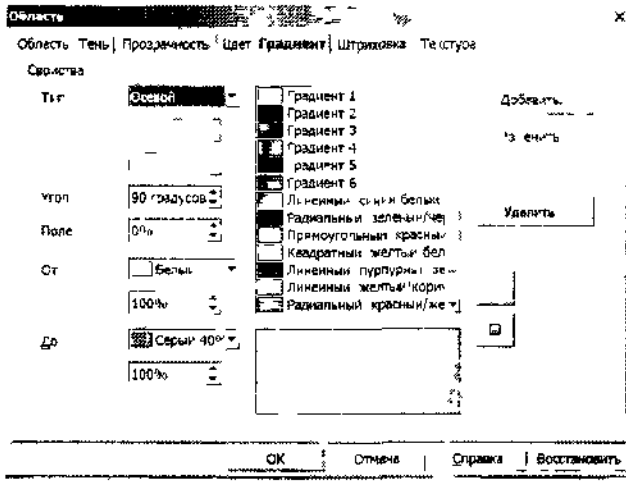
Вставим в функциональную схему соединительные стрелки.

8. На панели инструментов нажать на стрелку рядом с кнопкой *Соединительные линии*. На появившейся панели выбрать объект *Прямая соединительная линия со стрелками* и нарисовать стрелку, соединяющую магистраль с процессором.
9. Выделить стрелку и ввести команду [*Правка-Копировать*], а затем четыре раза — [*Правка-Вставить*]. Появившимися четырьмя стрелками соединить магистраль с устройствами компьютера.

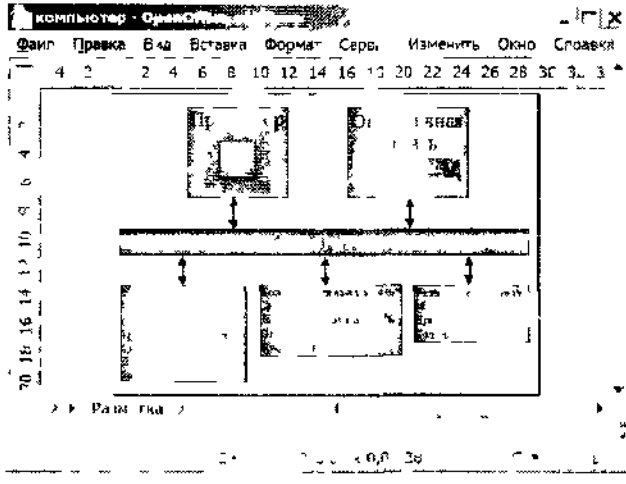
Подберем дизайн функциональной схемы компьютера, установим для каждого прямоугольника подходящий тип градиентной заливки.

10. Последовательно выделить прямоугольники и в контекстном меню выбрать пункт *Площадь...*

В появившемся диалоговом окне *Область* выбрать вкладку *Градиент* и на ней тип градиентной заливки *Осевой*, угол *90 градусов* и цвета *Белый* и *Серый*.



Получим примерно такую функциональную схему компьютера:



Для удобства изменения размера или перемещения функциональной схемы компьютера, состоящей из отдельных объектов, эти объекты целесообразно сгруппировать.

11. Последовательно выделить мышью все объекты при нажатой клавише *{Shift}* и ввести команду *[Группировка]*.

Задание 2. Сохранение рисунка в векторном и растровом графических форматах

Сохраним созданный векторный рисунок в собственном формате редактора OpenOffice.org Draw, а затем экспортируем его в растровый формат JPEG.

1. Ввести команду [*Файл Сохранить как...*], и в окне *Сохранить как* выбрать формат *Рисунок OpenDocument (odg)* и имя файла, например *компьютер*. Будет сохранен векторный графический файл компьютер.odg.
2. Ввести команду [*Файл Экспорт...*], и в окне *Экспорт* выбрать любой растровый формат (например, *JPEG*). Файл будет сохранен в этом формате: компьютер.jpg.

Файлы компьютер.odg и компьютер.jpg хранятся в папке ..\IIKT9\Graph **Windows-CD**

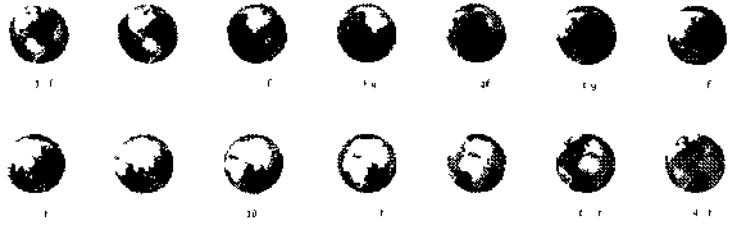
Практическая работа 1.4
Анимация

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться создавать анимацию в презентациях, GIF- и flash-анимацию.

Задание 1. Создать в презентации анимационное движение Земли вокруг Солнца.

Задание 2. Создать GIF-анимацию «Вращении Земли» из набора растровых GIF-изображений, показывающих последовательные положения Земли.



Задание 3. Создать flash-анимацию последовательного преобразования синего квадрата в зеленый треугольник и красный круг.

Рис. 1.22. Последовательность кадров flash-анимации преобразования синего квадрата в зеленый треугольник и красный круг

Задание 1. Анимация в презентации

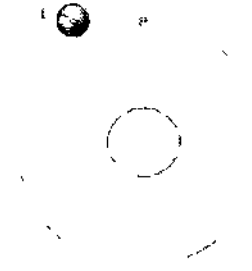
1. В операционной системе Windows запустить программу разработки презентаций Microsoft PowerPoint и ввести команду [*Файл-Создать*].
   Или:
   в операционной системе Windows или Linux запустить программу разработки презентаций OpenOffice.org Impress и ввести команду [*Файл Создать Презентацию...*].

2. На слайде нарисовать желтый круг (Солнце) и ввести команду [*Вставка Рисунок Из файла...*].
   В диалоговом окне *Добавление рисунка* в иерархической файловой системе на диске Windows-CD в папке ..\IIRT9\Graph\ выбрать файл Земля.jpg.

Создадим анимационное движение Земли вокруг Солнца и вращение Земли путем задания анимационных эффектов.

3. Ввести команду [*Показ слайдов-Эффекты анимации*]. Появится диалоговое окно *Настройка анимации*.

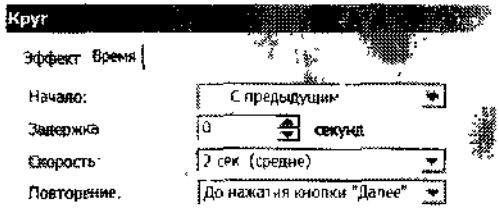
4. Выделить объект *Земля*, щелкнуть по кнопке *Добавить эффект* и ввести команду [*Пути перемещения Круг*].
   Переместить объект *Солнце* так, чтобы оно оказалось в центре круга (траектории перемещения объекта *Земля*).



5. Выделить объект *Земля*, щелкнуть по кнопке *Добавить эффект* и ввести команду [*Выделение Вращение*].

Настроим анимационные эффекты так, чтобы они начинались одновременно и заканчивались с переходом на следующий слайд.

6. В диалоговом окне *Настройка анимации* выделить по очереди анимационные эффекты и в контекстном меню ввести команду [*Параметры эффектов...*].
   В появившемся диалоговом окне на вкладке *Время* установить параметры анимационного эффекта.



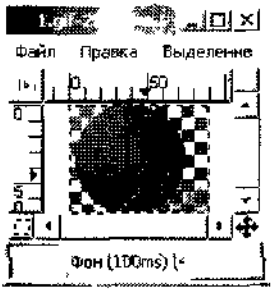
7. Запустить презентацию на выполнение командой [*Показ слайдов-Начать показ*] и наблюдать вращение Земли вокруг Солнца и собственной оси.

Файлы Анимация.odp и Анимация.ppt хранятся в папке ..\IIKT9\Graph **Windows-CD**

Задание 2. Создание GIF-анимации с помощью растрового графического редактора GIMP

Загрузим в растровый редактор набор растровых изображений, показывающих последовательные положения Земли.

1. В операционной системе Windows или Linux запустить редактор GIMP командой [*Пуск Программы-GIMP-GIPM2*].
   С помощью команды [*Файл-Открыть...*] последовательно загрузить в окна изображений файлы 1.gif – 14.gif из папки ..\IIKT9\GIF-анимация\.

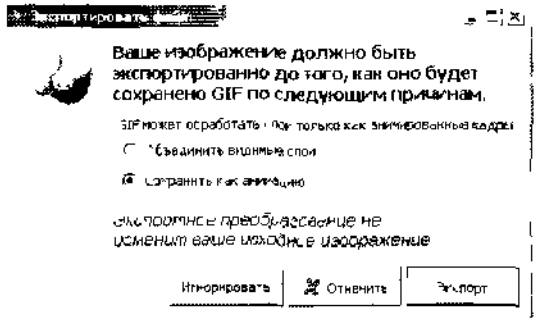


Последовательно скопируем изображения 2.gif – 14.gif в окно с изображением 1.gif.

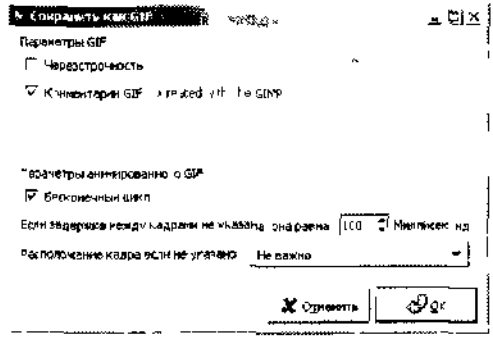
2. В окне изображений 2.gif – 14.gif вводить команду [*Правка-Копировать*].
   В окне изображения 1.gif вводить команду [*Правка-Вставить в*].

Сохраним полученное изображение как GIF-анимацию.

3. В окне изображения 1.gif выбрать команду [*Файл-Сохранить как...*].
   В появившемся диалоговом окне *Экспортировать файл* установить переключатель в положение *Сохранить как анимацию* и щелкнуть по кнопке *Экспорт*.



4. В появившемся диалоговом окне *Сохранить как GIF* установить флажок *Бесконечный цикл* и с помощью счетчика установить задержку между кадрами в миллисекундах.



GIF-анимация «Вращение Земли» хранится в папке ..\IIKT9\GIF-анимация **Windows-CD**

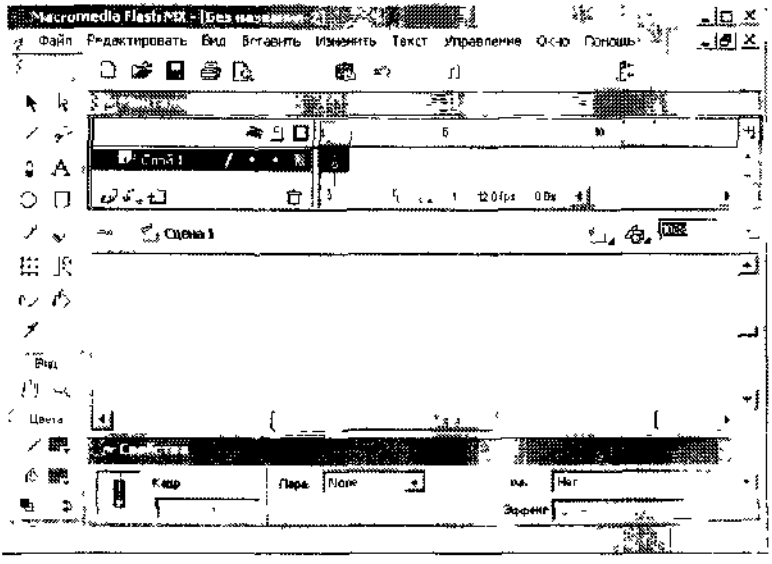
Задание 3. Создание flash-анимации с помощью редактора Macromedia Flash

1. В операционной системе Windows запустить редактор Macromedia Flash командой [*Пуск Программы Macromedia Macromedia Flash*].

В верхней части появившегося окна редактора находится *Монтажный стол*, на котором размещена *Монтажная линейка*, содержащая последовательность пустых кадров.

В середине находится *Окно рабочего поля*, в котором создаются кадры.

Внизу расположено диалоговое окно *Свойства*, которое позволяет устанавливать тип анимации.



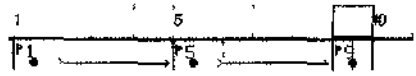
Слева размещены панели *Инструменты* и *Цвета*, которые используются при рисовании объектов.

Выберем ключевые кадры (например, 1, 5 и 9) и нарисуем на них синий квадрат, зеленый треугольник и красный круг.

2. Выбрать кадр 1 (ключевой по умолчанию) и в левой части *Окна рабочего поля* нарисовать синий квадрат.
3. Выбрать кадр 5, щелкнуть правой кнопкой мыши и преобразовать его в ключевой с помощью команды контекстного меню *Преобразовать в ключевой кадр*.
   В центре *Окна рабочего поля* нарисовать зеленый треугольник.
4. Выбрать кадр 9, щелкнуть правой кнопкой мыши и преобразовать его в ключевой с помощью команды контекстного меню *Преобразовать в ключевой кадр*.
   В правой части *Окна рабочего поля* нарисовать красный круг.

Установим тип анимационного перехода между ключевыми кадрами, который позволит автоматически создать промежуточные кадры.

5. Выбрать ключевой кадр 1 и в диалоговом окне *Свойства* с помощью раскрывающегося списка *Пара:* выбрать пункт *Shape*.
Повторить действия п. 4 и 5 для ключевого кадра 5.
На *Монтажной линейке* последовательность кадров приобретет салатовый цвет и между ключевыми кадрами появятся стрелки.



6. Для просмотра полученной анимации ввести команду [*Управление-Проиграть*].
7. Для просмотра полученной анимации по кадрам последовательно вводить команду [*Управление-Шаг вперед*].

Файл Анимация.fla хранится в папке ..\ИКТ9\Flash-анимация\ Windows-CD

Практическая работа 1.5

Кодирование и обработка звуковой информации

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux, *звуковой платой, подключенным микрофоном и динамиками.*

Цель работы. Научиться оцифровывать звук, редактировать звуковые записи и сохранять звуковые файлы в различных форматах.

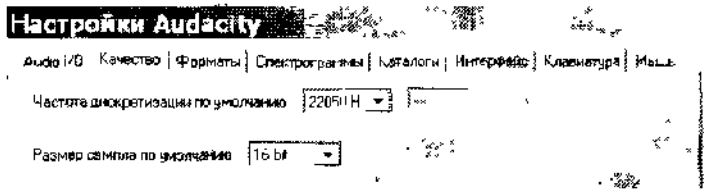
Задание. Записать оцифрованный звук, отредактировать запись, наложить две записи, применить звуковые эффекты и сохранить звуковые файлы в различных форматах.

Кодирование и обработка звуковой информации с помощью звукового редактора Audacity

1. В операционной системе Windows или Linux запустить звуковой редактор Audacity.

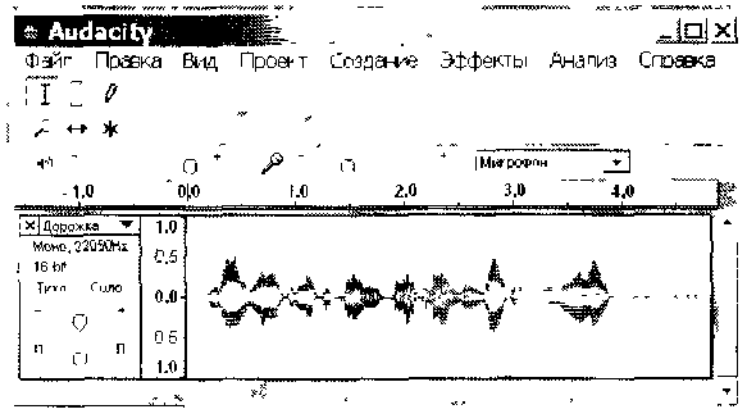
Установим частоту дискретизации звука 22050 Гц и глубину кодирования звука 16 битов.

2. В окне приложения ввести команду [*Правка-Настроить...*]. В диалоговом окне *Настройки Audacity* на вкладке *Качество* в соответствующих полях ввести частоту дискретизации и глубину кодирования.



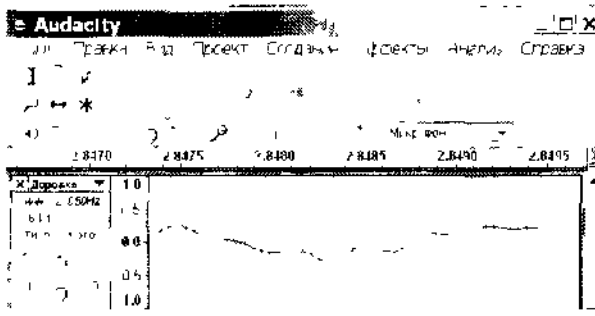
Запишем оцифрованный звук.

3. В окне приложения на панели инструментов щелкнуть по кнопке *Записывать* и с помощью микрофона начать запись звука. Для остановки записи щелкнуть по кнопке *Остановить*.
4. В окне приложения появится графическое отображение зависимости громкости записанного оцифрованного звука от времени.



Ознакомимся с точками оцифровки, отображенными на графике зависимости громкости звука от времени.

5. В окне приложения несколько раз ввести команду [*Вид-Приблизить*]. Шкала времени будет существенно растянута, и на графике станут видны точки оцифровки звука.



Осуществим редактирование оцифрованного звука: перенесем начальный фрагмент записи в ее окончание.

6. На графическом отображении звуковой дорожки выделить с помощью мыши ее начальный фрагмент и ввести команду [*Правка-Вырезать*].
   Установить курсор на конец записи и ввести команду [*Правка-Вставить*].
   Прослушать отредактированную запись, щелкнув на панели инструментов по кнопке *Воспроизвести*.

Осуществим наложение (микширование) двух записей.

7. Записать вторую дорожку оцифрованного звука (см. п. 3) или открыть существующий звуковой файл командой [*Файл Открыть...*].
   Прослушать наложение двух записей, щелкнув на панели инструментов по кнопке *Воспроизвести*.

Применим к записи звуковые эффекты (*Разворот*, *Смена скорости*, *Эхо* и другие).

8. Установить курсор на начало записи и ввести команды [*Эффекты-Разворот*], [*Эффекты-Смена скорости...*], [*Эффекты Эхо...*] и другие.
   Прослушать результаты применения звуковых эффектов, щелкнув на панели инструментов по кнопке *Воспроизвести*.

Сохраним оцифрованный звук в звуковом файле.

9. Для сохранения оцифрованного звука в собственном формате звукового файла AUP ввести команду [*Файл-Сохранить проект как...*] и задать имя файла и его местоположение в файловой системе.
10. Для сохранения оцифрованного звука в универсальном формате звукового файла WAV ввести команду [*Файл-Экспортировать как WAV...*] и задать имя файла и его местоположение в файловой системе.

11. Сравнить информационные объемы звуковых файлов, сохраненных в различных форматах.

Файлы 1.aup и 1.wav хранится в папке ..\IIKT9\Звук **Windows-CD**

Практическая работа 1.6
Захват цифрового фото и создание слайд-шоу

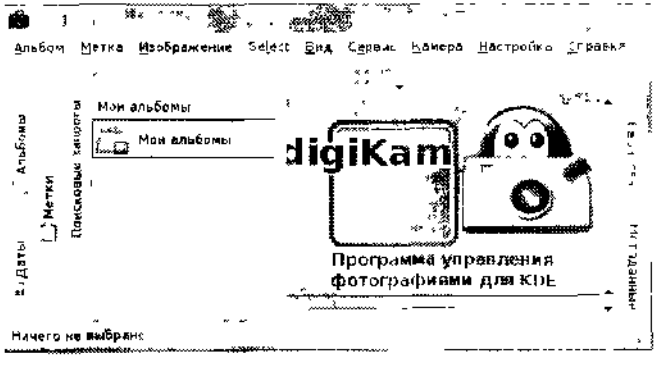
Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Linux и цифровой фотокамерой, подключенной к USB-порту (в операционной системе Windows надо установить программу захвата и редактирования цифровых фотографий, полученную с камерой).

Цель работы. Научиться захватывать снимки с цифровых фотокамер и создавать слайд-шоу.

Задание. Захватить фото с цифровой фотокамеры и создать слайд-шоу.

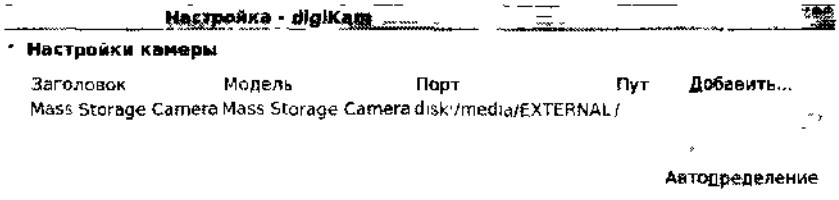
Захват цифровых фото и создание слайд-шоу с использованием системы digiKam

1. Запустить систему захвата цифровых фотографий командой [*Графика-Работа с фотографиями digiKam*]. В появившемся окне приложения выбрать пункт *Камера*, чтобы выбрать модель подключенной цифровой камеры.



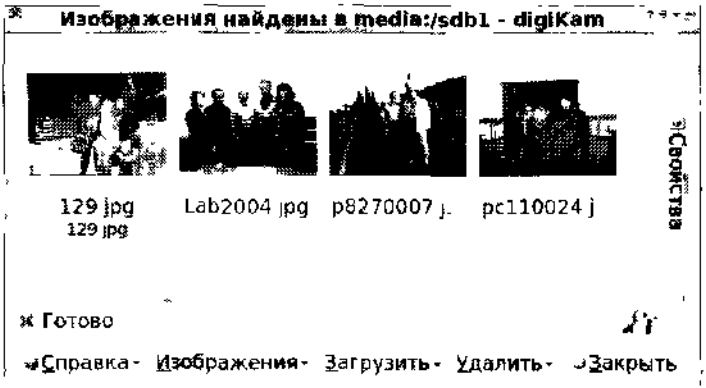
2. В появившемся окне *Параметры камеры* из списка выбрать модель подключенной цифровой камеры.

3. Если в списке такой камеры нет, щелкнуть по кнопке *Автоопределение*. В верхней строке появится модель *Mass Storage Camera* и путь к ее флэш-карте disk./media/EXTERNAL.

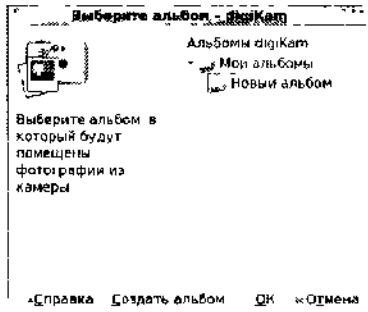


Выполним захват цифровых фотографий, т. е. их копирование с карты памяти цифровой камеры на жесткий диск компьютера.

4. В окне программы ввести команду [*Камера-Media Brouse-EXTERNAL*]. Появится окно, содержащее изображения, найденные на флэш-карте камеры.



5. Щелкнуть по кнопке *Загрузить* и выбрать пункт *Загрузить все*. Появится окно *Выберите альбом*. Щелкнуть по кнопке *Создать альбом* и ввести его имя (например, *Новый альбом*).



Создадим слайд-шоу из загруженных с камеры фотографий.

6. В окне приложения digiKam ввести команду [*Вид-Слайд-шоу-Все*]. На экране монитора компьютера будут последовательно появляться фотографии в полноэкранном режиме.

Практическая работа 1.7

Захват и редактирование цифрового видео с использованием системы нелинейного видеомонтажа

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Linux и цифровой видеокамерой, подключенной к DV-порту (в операционной системе Windows надо использовать программу нелинейного видеомонтажа Windows Movie Maker).

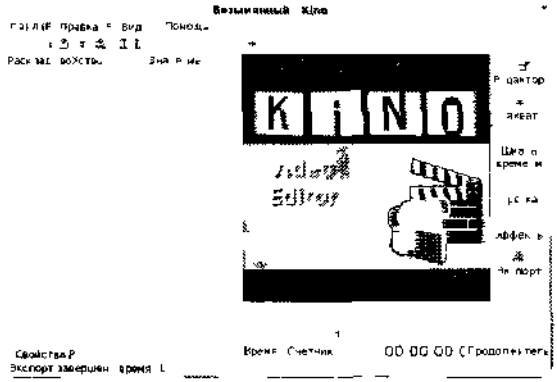
Цель работы. Научиться захватывать и редактировать цифровые видеозаписи.

Задание. Оцифровать, отредактировать и сохранить видеофильм.

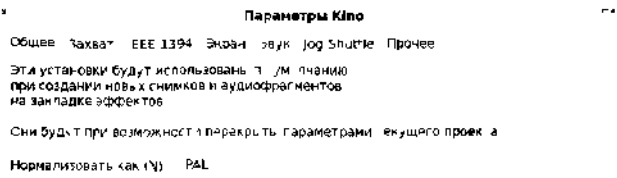
Захват и редактирование цифрового видео с использованием системы нелинейного видеомонтажа KINO

Выполним захват цифрового видео с заданным качеством, т. е. считывание видеозаписи с магнитной кассеты цифровой видеокамеры и сохранение в видеофайле на жестком диске компьютера.

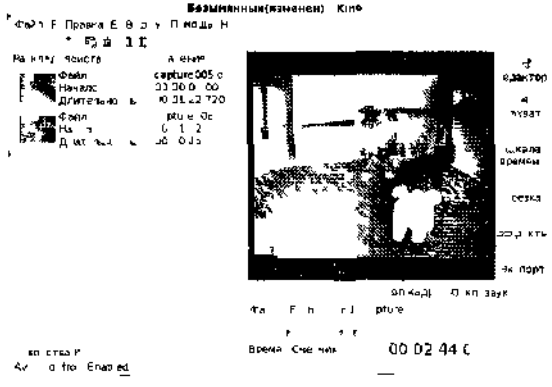
1. Запустить систему видеомонтажа KINO, в диалоговом окне которой можно выбрать режим оцифровки, монтажа и сохранения видеофайла: *Редактор*, *Захват*, *Шкала времени*, *Срезка*, *Эффекты* и *Экспорт*.



2. В окне программы ввести команду [*Правка Настройки*]. Появившееся диалоговое окно *Параметры KINO* позволяет на вкладках выбрать параметры захвата, редактирования и сохранения видеофайла.

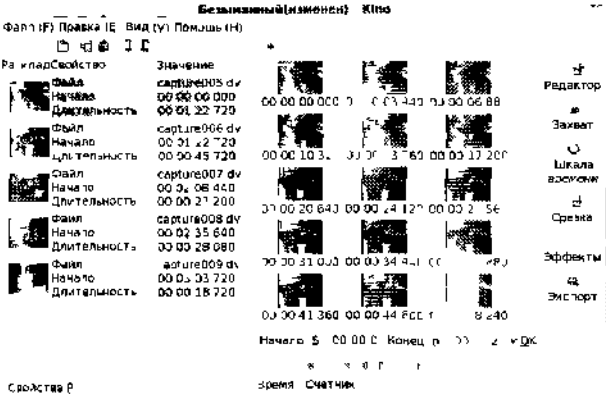


3. В окне программы выбрать пункт *Захват*. Для начала оцифровки видеофрагмента щелкнуть по кнопке *Захват*, а для окончания захвата щелкнуть по кнопке *Стоп*. В левой части окна приложения отобразятся данные об оцифрованных видеофрагментах.



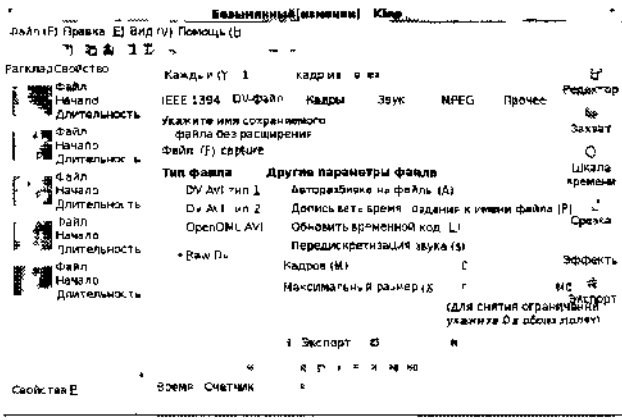
Видеофайл автоматически разбивается на сцены, которые можно увидеть в окне *Шкала времени*.

4. В окне системы видеомонтажа выбрать режим *Шкала времени* и увидеть сцены выбранного видеофрагмента.
С помощью счетчика *Время:* можно установить длительность сцен от отдельных кадров до часов.
В окне системы видеомонтажа выбрать режим *Срезка*.
С помощью меню приложения *Правка* можно редактировать оцифрованный видеофильм.



Сохраним смонтированный цифровой видеофильм с требуемым качеством на жестком диске компьютера.

5. В окне системы видеомонтажа выбрать режим *Экспорт*.
Выбрать место сохранения и качество цифрового видео *(DV AVI или MPEG)*.
Щелкнуть по кнопке *Экспорт*.



Практические работы к главе 2 «Кодирование и обработка текстовой информации»

	Установить: • текстовый редактор OpenOffice.org Writer; • текстовый редактор Hieroglyph; • англо-русский словарь SV-Translator; • систему оптического распознавания документов FineReader; • текстовый редактор Microsoft Word	Windows-CD Дистрибутив Microsoft Office
	Установить: • текстовый редактор OpenOffice.org Writer; • редактор формул OpenOffice.org Math	Linux-DVD

Практическая работа 2.1

Кодирование текстовой информации

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться определять числовые коды символов и осуществлять перекодировку русскоязычного текста в текстовом редакторе.

Задание 1. В текстовом редакторе определить числовые (шестнадцатеричные) коды нескольких символов в кодировке *Unicode (Юникод)*.

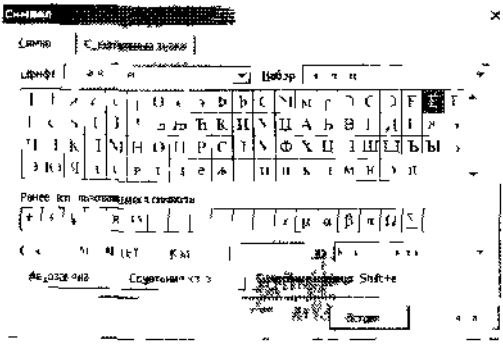
Задание 2. В текстовом редакторе Hieroglyph представить слово «Кодировка» в пяти различных кодировках: *Windows*, *MS-DOS*, *КОИ-8*, *Mac*, *ISO*.

Определение числового кода символа с помощью текстовых редакторов Microsoft Word и OpenOffice.org Writer

1. В операционной системе Windows запустить текстовый редактор Microsoft Word командой [*Программы Microsoft Word*] или текстовый редактор OpenOffice.org Writer командой [*Программы OpenOffice OpenOffice Writer*].
   Или:
   в операционной системе Linux запустить текстовый редактор OpenOffice.org Writer командой [*Офис OpenOffice Writer*].

Определим числовой код символа в текстовом редакторе Microsoft Word.

2. В текстовом редакторе Microsoft Word ввести команду [*Вставка Символ...*]. На экране появится диалоговое окно *Символ*. Центральную часть диалогового окна занимает фрагмент таблицы символов.



3. Для определения числового кода знака кириллицы с помощью раскрывающегося списка *Набор:* выбрать пункт *кириллица*.
4. Для определения шестнадцатеричного числового кода символа в кодировке *Unicode* с помощью раскрывающегося списка *из:* выбрать тип кодировки *Юникод (шестн.)*.
5. В таблице символов выбрать символ (например, заглавную букву Ё). В текстовом поле *Код знака:* появится его шестнадцатеричный числовой код (в данном случае 0401).

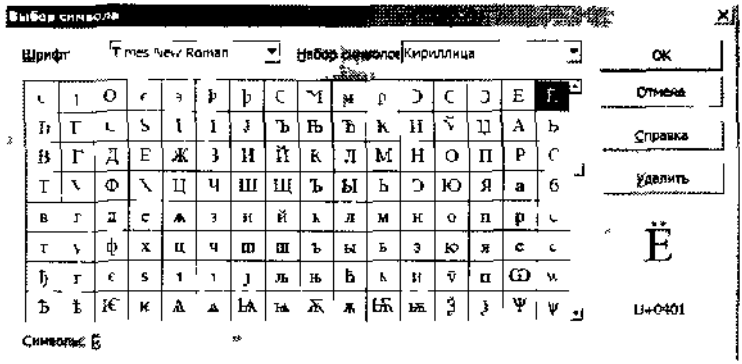
Перевод числового кода символа из шестнадцатеричной системы счисления в десятичную систему счисления можно

осуществить с помощью программного калькулятора NumLock Calculator.

Практическая работа 3.1. Задание 1. Перевод чисел из шестнадцатеричной системы счисления в десятичную с помощью программного калькулятора NumLock Calculator

Определим числовой код символа в текстовом редакторе OpenOffice.org Writer.

6. В текстовом редакторе OpenOffice.org Writer ввести команду [*Вставка Специальные символы...*]. На экране появится диалоговое окно Выбор символа. Центральную часть диалогового окна занимает фрагмент таблицы символов.



7. Для определения числового кода знака кириллицы с помощью раскрывающегося списка Набор символов: выбрать пункт кириллица.
8. В таблице символов выбрать символ (например, заглавную букву Ё). В правом нижнем углу диалогового окна появится его шестнадцатеричный числовой код (в данном случае 0401).

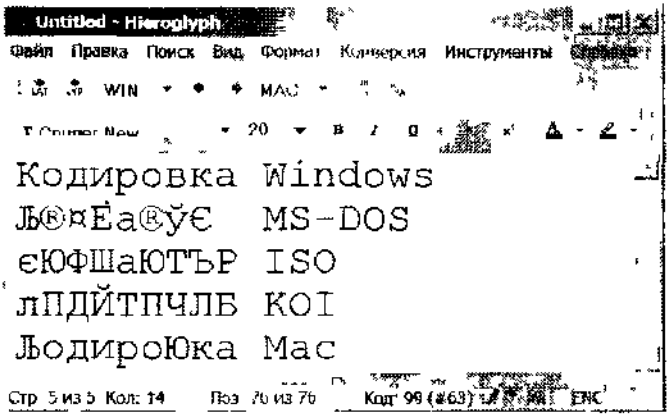
Перевод числового кода символа из шестнадцатеричной системы счисления в десятичную систему счисления можно осуществить с помощью программного калькулятора KCalc.

Практическая работа 3.1. Задание 2. Перевод чисел из шестнадцатеричной системы счисления в десятичную с помощью программного калькулятора KCalc

Перекодирование русскоязычного текста в текстовом редакторе Hieroglyph

1. В операционной системе Windows запустить текстовый редактор Hieroglyph командой [*Программы Hieroglyph*].

2. В раскрывающемся списке исходных кодировок выбрать кодировку *WIN* и ввести текст: «Кодировка Windows».
3. Скопировать текст четыре раза.
   Последовательно выделить строки, выбрать для каждой конечную кодировку в раскрывающемся списке (*DOS*, *KOI*, *Mac* и *ISO*), нажать кнопку *Перевод кодировки*. Для каждой кодировки отредактировать ее название.



4. В результате получим пять строк символов в различных кодировках, соответствующих одной и той же последовательности числовых кодов.

Файл перекодировка.txt хранится в папке ..\ИКТ9\Текст

Windows-CD

Практическая работа 2.2

Вставка в документ формул

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться вставлять в документ физические и математические формулы.

Задание 1. Вставить в документ формулу закона Ома с использованием *Редактора формул* (*Microsoft Equation*), встроенного в текстовый редактор Microsoft Word.

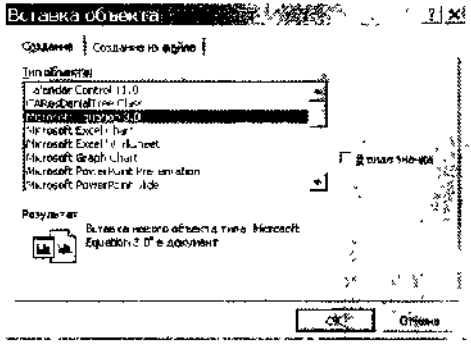
Задание 2. Вставить в документ формулу закона Ома с использованием редактора формул OpenOffice.org Math, встроенного в интегрированное офисное приложение OpenOffice.org.

Задание 1. Вставка в документ формул с использованием *Редактора формул*, встроенного в текстовый редактор Microsoft Word

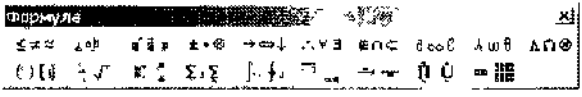
1. В операционной системе Windows запустить текстовый редактор Microsoft Word командой [*Программы-Microsoft Word*]. Создать новый документ с помощью команды [*Файл-Создать...*].

   Вызовем *Редактор формул (Microsoft Equation)*.
2. Для вставки в документ формулы необходимо ввести команду [*Вставка-Объект...*], появится диалоговое окно *Вставка объекта*.

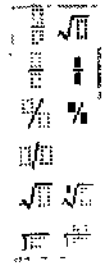
   На вкладке *Создание* в списке *Тип объекта:* выбрать пункт *Microsoft Equation 3.0* и щелкнуть по кнопке *OK*.



3. В тексте документа появится рамка для ввода формулы, а в окне документа появится панель инструментов *Редактора формул*.



   Вставим в документ формулу закона Ома.



4. Внутри рамки для ввода формул ввести на латинской клавиатуре левую часть формулы и знак равенства $I=$.

   На панели инструментов *Редактора формул* щелкнуть по кнопке *Шаблоны дробей и радикалов*.

   На открывшейся панели выбрать кнопку с изображением дроби и щелкнуть по ней мышью.

   В рамке для ввода формулы появится заготовка дроби, в которую ввести знаки U и R.

5. В результате в рамке для ввода формул появится формула закона Ома. $I = \frac{U}{R}$

Задание 2. Вставка в документ формул с использованием редактора формул OpenOffice.org Math, встроенного в интегрированное офисное приложение OpenOffice.org

1. В операционной системе Windows или Linux запустить текстовый редактор OpenOffice.org Writer соответственно командой [*Программы-OpenOffice-OpenOffice Writer*] или [*Офис-OpenOffice Writer*].

   Вызовем редактор формул OpenOffice.org Math.

2. Для вставки в документ формулы ввести команду [*Вставка-Объект Формула...*], появится панель *Выбор*.

   На панели *Выбор* выбрать шаблон формулы *a/b*.

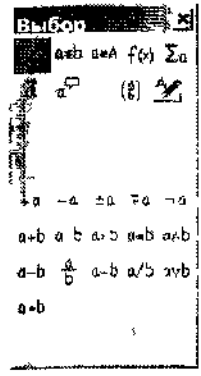
   В документе появится заготовка формулы, а в окне OpenOffice.org Math — код:

   `<?> over <?>`

   Ввести в код формулы конкретные переменные закона Ома:

   `I = U over R`

   В документе появится формула закона Ома: $I = \frac{U}{R}$.



Практическая работа 2.3
Форматирование символов и абзацев

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться устанавливать в документе различные параметры форматирования символов и абзацев.

Задание 1. В текстовом редакторе Microsoft Word ввести в документ строки и отформатировать их по указанному в самих строках образцу (шрифт, размер, начертание и цвет):

Times New Roman, 14, курсив, красный, 10^2;

Arial, 8, полужирный подчеркнутый, зеленый, 10_2;

Courier New, 10, полужирный курсив, синий.

Задание 2. В текстовом редакторе OpenOffice.org Writer ввести в документ абзацы и отформатировать их по указанному в самих абзацах образцу (шрифт, выравнивание, отступы первой строки, отступы абзаца целиком, междустрочные интервалы и интервалы между абзацами):

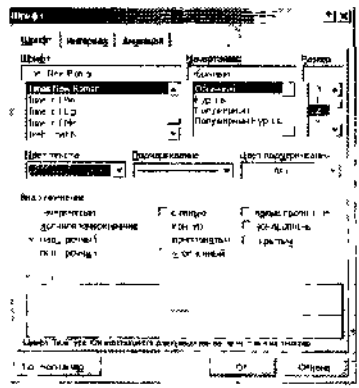
Абзац с выравниванием по ширине, отступ слева 6 см, шрифт Times New Roman, размер 12 пт, обычный.

Абзац с выравниванием по центру, шрифт Arial, размер 14 пт, полужирный.

Абзац с выравниванием по левому краю, отступ первой строки 1,25 см, шрифт Courier New, размер 10 пт, курсив, подчеркнутый.

Задание 1. Форматирование символов в текстовом редакторе Microsoft Word

1. В операционной системе Windows запустить текстовый редактор Microsoft Word командой [*Программы-Microsoft Word*]. Создать новый документ с помощью команды [*Файл-Создать...*].
2. Ввести в документ строки, указанные в задании 1.
3. Для форматирования шрифта ввести команду [*Формат-Шрифт...*], откроется диалоговое окно *Шрифт*.
   На вкладке *Шрифт* с помощью раскрывающихся списков установить параметры форматирования: шрифт, размер, начертание, цвет символов и варианты подчеркивания.



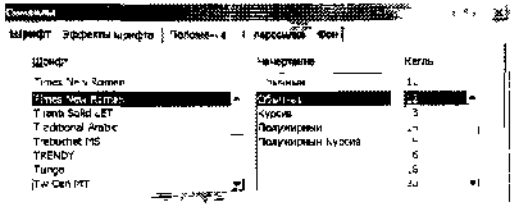
4. Установить верхний 10^2 и нижний 10_2 индексы с помощью флажков *надстрочный* и *подстрочный* группы *Видоизменение*.

Задание 2. Форматирование абзацев в текстовом редакторе OpenOffice.org Writer

1. В операционной системе Windows или Linux запустить текстовый редактор OpenOffice.org Writer соответственно командой [*Программы-OpenOffice-OpenOffice Writer*] или [*Офис-OpenOffice Writer*].
2. Внести в документ абзацы, указанные в задании 2.

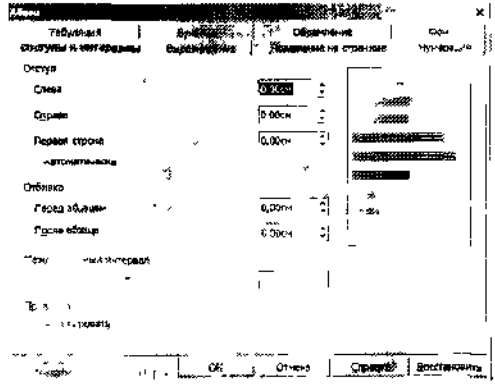
   Отформатируем шрифт.
3. Для форматирования шрифта ввести команду [*Формат-Символы...*].

   В появившемся диалоговом окне *Символы* на вкладке *Шрифт* установить с помощью списков гарнитуру шрифта, начертание и размер.



   Отформатируем абзацы.
4. Для форматирования абзаца ввести команду [*Формат-Абзац...*].

   В появившемся диалоговом окне *Абзац* на вкладке *Отступы и интервалы* установить с помощью счетчиков отступы абзаца, отступы и интервалы между абзацами и строками.
5. На вкладке *Выравнивание* установить параметры выравнивания абзацев.



Практическая работа 2.4

Создание и форматирование списков

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться создавать маркированные, нумерованные и многоуровневые списки.

Задание 1. Создать и отформатировать маркированный список (см. рис. 2.11) в текстовом редакторе Microsoft Word.

Задание 2. Создать и отформатировать нумерованный список (см. рис. 2.10) в текстовом редакторе Microsoft Word.

Задание 3. Создать и отформатировать многоуровневый список (см. рис. 2.12) в текстовом редакторе OpenOffice.org Writer.

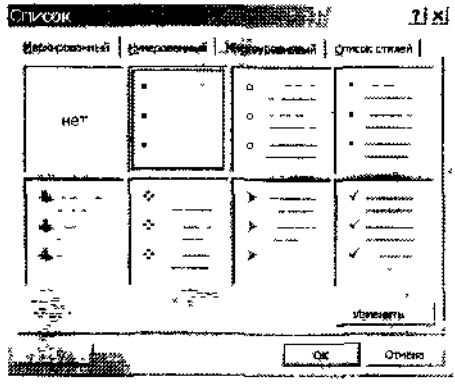
Задание 1. Создание и форматирование маркированных списков в текстовом редакторе Microsoft Word

1. В операционной системе Windows запустить текстовый редактор Microsoft Word командой [*Программы-Microsoft Word*]. Создать новый документ с помощью команды [*Файл Создать...*].

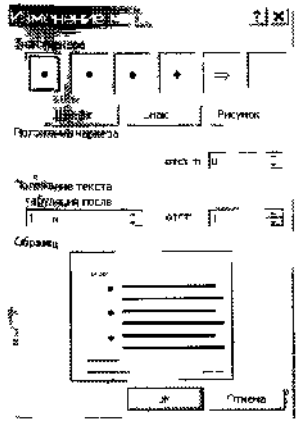
Вставим в документ и отформатируем маркированный список.

2. Ввести команду [*Формат-Список...*], и в диалоговом окне *Список* на вкладке *Маркированный* с помощью графического переключателя выбрать внешний вид списка.

   Для детальной установки параметров списка щелкнуть по кнопке *Изменить*.



3. В диалоговом окне *Изменение маркированного списка* установить:
   - вид маркера с помощью графического переключателя *Знак маркера*;
   - отступ списка от края области текста с помощью счетчика *Положение маркера отступ:*;
   - отступ текста списка от маркера с помощью счетчиков *Положение текста табуляция после:* и *отступ:*.



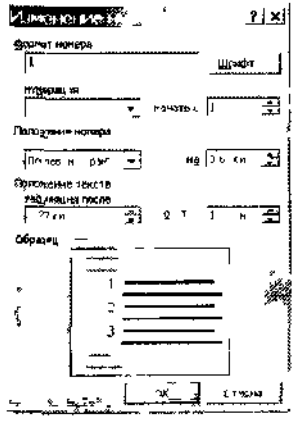
Задание 2. Создание и форматирование нумерованных списков в текстовом редакторе Microsoft Word

1. В операционной системе Windows запустить текстовый редактор Microsoft Word командой [*Программы Microsoft Word*]. Создать новый документ с помощью команды [*Файл Создать...*].

Вставим в документ и отформатируем нумерованный список.

2. Ввести команду [*Формат-Список...*], и в диалоговом окне *Список* на вкладке *Нумерованный* с помощью графического переключателя выбрать внешний вид списка.

   Для детальной установки параметров списка щелкнуть по кнопке *Изменить*.



3. В диалоговом окне *Изменение нумерованного списка* установить:
   - формат номера, нажав на кнопку *Шрифт*;
   - вид нумерации (арабские или римские цифры, русские или латинские буквы и т. д.) с помощью раскрывающегося списка *нумерация:*;
   - начало нумерации с помощью счетчика *начать с:*;

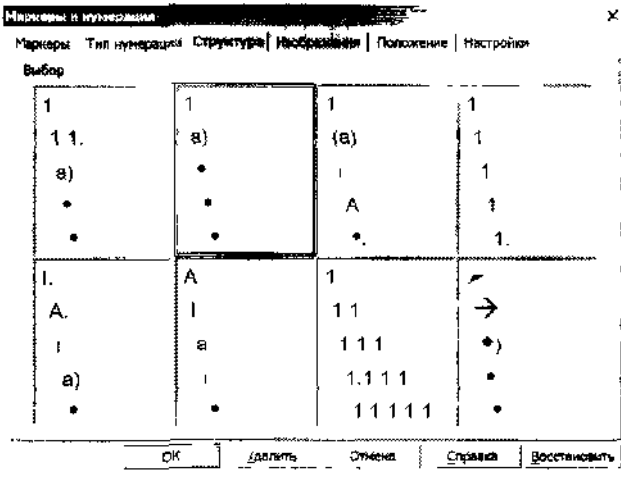
- отступ списка от края области текста с помощью счетчика *Положение номера на:*;
- отступ текста списка от номера с помощью счетчиков *Положение текста табуляция после:* и *отступ:*.

Задание 3. Создание и форматирование многоуровневых списков в текстовом редакторе OpenOffice.org Writer

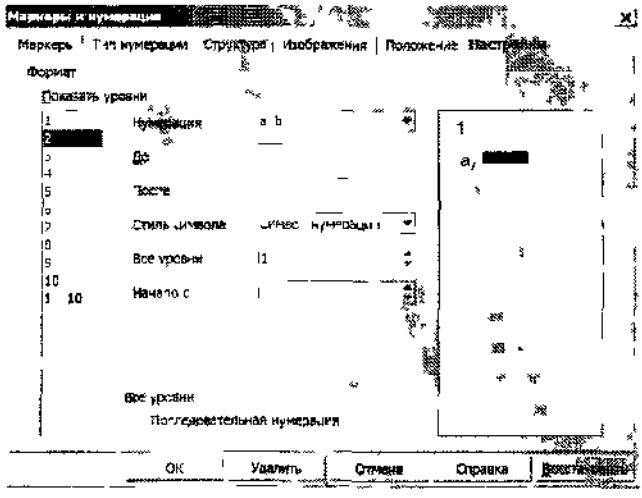
1. В операционной системе Windows или Linux запустить текстовый редактор OpenOffice.org Writer соответственно командой [*Программы-OpenOffice-OpenOffice Writer*] или [*Офис-OpenOffice Writer*].

Вставим в документ и отформатируем многоуровневый список. На первом уровне находится нумерованный список из двух элементов, в каждый из которых вложен нумерованный список из двух элементов второго уровня, во второй элемент которого вложен маркированный список из одного элемента третьего уровня.

2. Ввести команду [*Формат-Маркеры и нумерация...*], и в диалоговом окне *Маркеры* и нумерация на вкладке *Структура* выбрать внешний вид многоуровневого списка.
   Для детальной установки параметров списка перейти на вкладку *Настройки*.

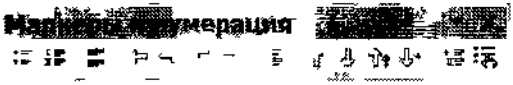


3. На вкладке *Настройки* установить для каждого уровня списка порядок нумерации списка и другие параметры.



В документе, где создается список, можно для каждого пункта многоуровневого списка повысить или понизить уровень, переместить пункт вверх или вниз.

4. Выделить пункт многоуровневого списка и на панели *Маркеры* и нумерация с помощью горизонтальных стрелок повысить или понизить уровень пункта списка, а с помощью вертикальных стрелок переместить его вверх или вниз.



Практическая работа 2.5

Вставка в документ таблицы, ее форматирование и заполнение данными

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться вставлять в документ таблицы, настраивать их внешний вид и вставлять данные (текст, изображения, числа и формулы).

Задание 1. Вставить в документ таблицу, настроить ее внешний вид и заполнить данными, включая вычисления по формуле (см. табл. 2.6) в текстовом редакторе Microsoft Word.

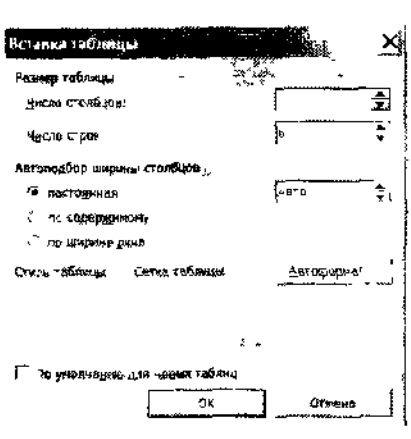
Задание 2. Вставить в документ таблицу, настроить ее внешний вид и заполнить данными, включая вычисления по формуле (см. табл. 2.6) в текстовом редакторе OpenOffice.org Writer.

Задание 1. Вставка в документ таблицы, ее форматирование и вставка формулы в текстовом редакторе Microsoft Word

1. В операционной системе Windows запустить текстовый редактор Microsoft Word командой [*Программы-Microsoft Word*]. Создать новый документ с помощью команды [*Файл-Создать...*].

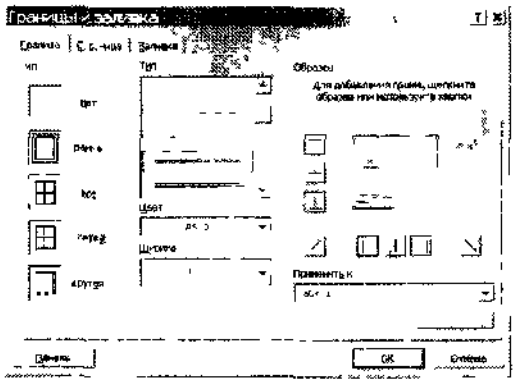
Вставим в документ таблицу, состоящую из определенного количества строк и столбцов.

2. Вставить в документ таблицу при помощи команды [*Таблица-Вставить таблицу...*].
В диалоговом окне *Вставка таблицы* с помощью счетчиков указать количество столбцов (3) и строк (6) в создаваемой таблице.



Отформатируем таблицу по образцу (см. табл. 2.6).

3. Ввести команду [*Формат-Границы и заливка...*]. Диалоговое окно *Границы и заливка* позволяет выбрать требуемые параметры.
На вкладке *Граница* можно задать тип границы (*Нет*, *Сетка*, *Рамка* и др.), тип и ширину линий границы.
На вкладке *Заливка* можно задать цвет фона ячеек или выбрать узор.

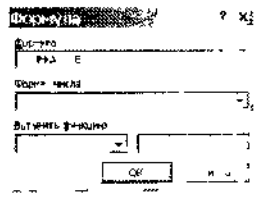


Введем в таблицу данные (текст, изображения и числа).

4. Вставить в ячейки таблицы данные.

Вставим в последнюю ячейку правого столбца таблицы формулу суммирования, позволяющую определить стоимость компьютера.

5. Ввести команду [*Таблица-Формула...*]. В диалоговом окне *Формула* ввести в поле *Формула* функцию =SUM(ABOVE), суммирующую числа, находящиеся в вышележащих ячейках.

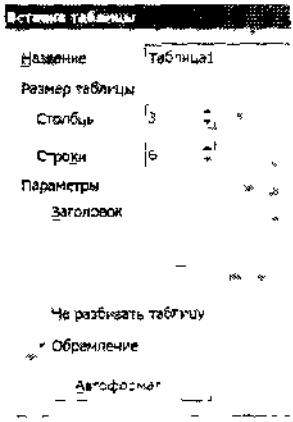


Задание 2. Вставка в документ таблицы, ее форматирование и вставка формулы в текстовом редакторе OpenOffice.org Writer

1. В операционной системе Windows или Linux запустить текстовый редактор OpenOffice.org Writer соответственно командой [*Программы-OpenOffice.org-OpenOffice.org Writer*] или [*Офис-OpenOffice.org Writer*].

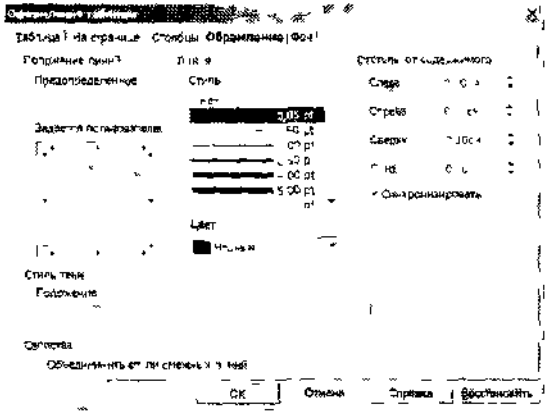
Вставим в документ таблицу, состоящую из определенного количества строк и столбцов.

2. Вставить в документ таблицу при помощи команды [*Таблица-Вставить-Таблица...*].
В диалоговом окне *Вставка таблицы* с помощью счетчиков указать количество столбцов (3) и строк (6) в создаваемой таблице.



Отформатируем таблицу по образцу (см. табл. 2.6).

3. Ввести команду [*Таблица Свойства таблицы...*]. Появится диалоговое окно *Оформление таблицы*, позволяющее выбрать требуемые параметры.
На вкладке *Обрамление* можно задать тип границы (*Нет*, *Сетка*, *Рамка* и др.), тип и ширину линий границы.
На вкладке *Фон* можно задать цвет фона ячеек или выбрать узор.



Введем в таблицу данные (текст, изображения и числа).

4. Вставить в ячейки таблицы данные.

Вставим в последнюю ячейку правого столбца таблицы формулу суммирования, позволяющую определить стоимость компьютера.

5. Ввести команду [*Таблица-Формула...*].
Ввести в поле функцию, суммирующую числа, находящиеся в вышележащих ячейках:
`=sum(<C2>+<C3>+<C4>+<C5>)`

Практическая работа 2.6

Перевод текста с помощью компьютерного словаря

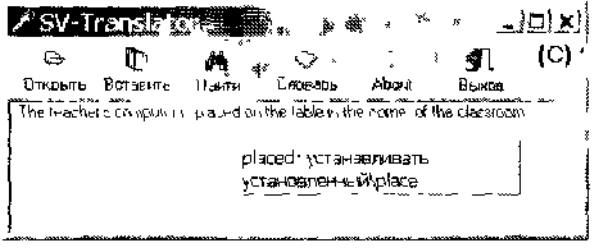
Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows.

Цель работы. Научиться использовать компьютерные словари для перевода текстов.

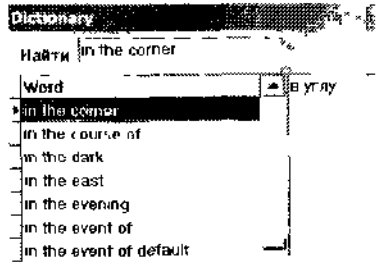
Задание. Перевести с английского языка на русский язык с помощью компьютерного словаря предложение, например: «The teacher's computer is placed on the table in the corner of the classroom».

Перевод текста при помощи компьютерного словаря SV-Translator

1. В операционной системе Windows запустить текстовый редактор Microsoft Word командой [*Программы-Microsoft Word*]. Создать новый документ с помощью команды [*Файл-Создать...*].
2. Ввести английское предложение, выделить его и скопировать в *буфер обмена* командой [*Правка-Копировать*].
3. Запустить компьютерный англо-русский словарь SV-Translator командой [*Программы-Svtrans*]. Вставить предложение из *буфера обмена* командой [*Вставить*].
4. Последовательно подвести указатель мыши к словам английского предложения и с помощью появляющихся русских переводов перевести предложение целиком.



5. Для перевода английских текстов можно также использовать встроенный словарь, который запускается командой [*Словарь*].
   В поле *Найти:* ввести английское слово или словосочетание, например «in the corner». В правом поле появится перевод «в углу».



Практическая работа 2.7

Сканирование и распознавание «бумажного» текстового документа

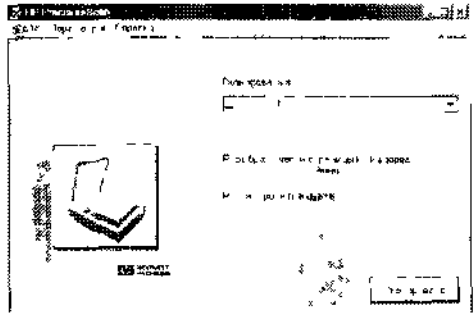
Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows, сканер.

Цель работы. Научиться сканировать «бумажные» тексты и преобразовывать их в компьютерные текстовые документы с помощью систем оптического распознавания.

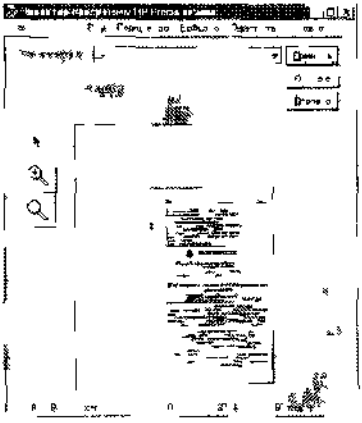
Задание. Отсканировать и преобразовать в компьютерный текстовый документ страницу учебника.

Сканирование и распознавание «бумажного» текстового документа

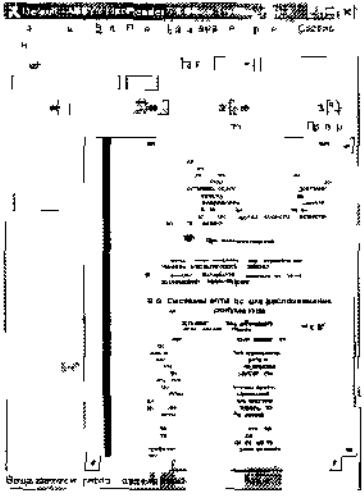
1. В операционной системе Windows запустить систему оптического распознавания документов FineReader командой [*Программы-ABBYY FineReader*].
2. В окне системы оптического распознавания щелкнуть по кнопке *Сканировать*.
   В появившемся окне сканера щелкнуть по кнопке *Сканировать*.



3. В окне *Проектор программы* (сканера) появится отсканированное изображение текстовой страницы.
Для передачи изображения в систему оптического распознавания щелкнуть по кнопке *Принять*.



4. В окне системы оптического распознавания появится отсканированное изображение текстовой страницы.
Для преобразования графического изображения страницы в текстовый файл щелкнуть по кнопке *Распознать*.
После окончания процесса распознавания ввести команду [*Файл Сохранить текст как..*], выбрать место сохранения, имя и тип полученного текстового файла.



5. Открыть полученный документ в текстовом редакторе и исправить возможные ошибки, допущенные в процессе распознавания.

Практические работы к главе 3 «Кодирование и обработка числовой информации»

	Установить: • электронный калькулятор NumLock Calculator; • электронные таблицы OpenOffice.org Calc; • электронные таблицы Microsoft Excel	Windows-CD Дистрибутив Microsoft Office
	Установить: • электронный калькулятор KCalc; • электронные таблицы OpenOffice.org Calc	Linux-DVD

Практическая работа 3.1

Перевод чисел из одной системы счисления в другую с помощью калькулятора

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться с помощью калькулятора переводить целые числа из шестнадцатеричной системы счисления в десятичную для определения десятичного кода символа.

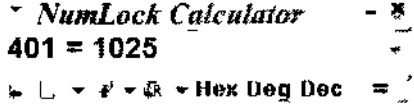
Задание 1. В операционной системе Windows перевести шестнадцатеричный код символа в десятичный с помощью программного калькулятора NumLock Calculator.

Задание 2. В операционной системе Linux перевести шестнадцатеричный код символа в десятичный с помощью программного калькулятора KCalc.

Для перевода чисел из шестнадцатеричной системы счисления в десятичную необходимо установить в качестве исходной системы счисления шестнадцатеричную, а в качестве конечной — десятичную.

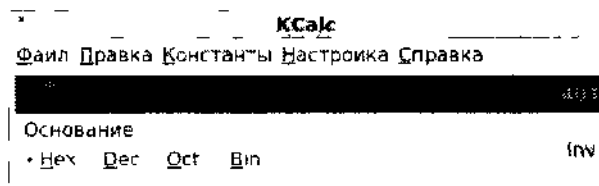
Задание 1. Перевод чисел из шестнадцатеричной системы счисления в десятичную с помощью программного калькулятора NumLock Calculator

1. В операционной системе Windows запустить электронный калькулятор NumLock Calculator командой [*Программы NumLock Calculator*].
2. С помощью меню ввести команды [*Формат чисел в выражении Шестнадцатеричный*] и [*Формат результата Десятичный*].
   Ввести шестнадцатеричный код символа и нажать кнопку {=}.

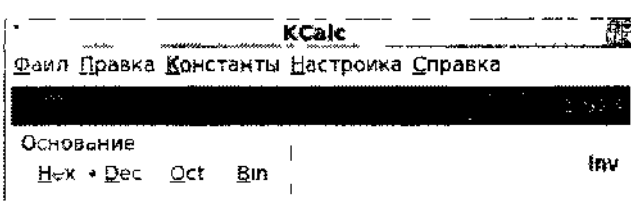


Задание 2. Перевод чисел из шестнадцатеричной системы счисления в десятичную с помощью программного калькулятора KCalc

1. В операционной системе Linux запустить электронный калькулятор KCalc командой [*Служебные Калькулятор (KCalc)*].
2. С помощью переключателя установить шестнадцатеричную систему (положение *Hex*) и ввести шестнадцатеричный код символа.



3. С помощью переключателя установить десятичную систему (положение *Dec*) и получить десятичный код символа.



Практическая работа 3.2

Относительные, абсолютные и смешанные ссылки в электронных таблицах

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Window или Linux.

Цель работы. Научиться использовать в формулах электронной таблицы относительные, абсолютные и смешанные ссылки.

Задание. Какой вид приобретут формулы, хранящиеся в диапазоне ячеек C1:C3 при их копировании в диапазон ячеек E2:E4?

	A	B	C	D	E
1			=A1+B1		
2			=$A$1+$B$1		
3			=$A1+B$1		
4					

Копирование формул, содержащих относительные, абсолютные и смешанные ссылки в электронных таблицах Microsoft Excel и OpenOffice.org Calc

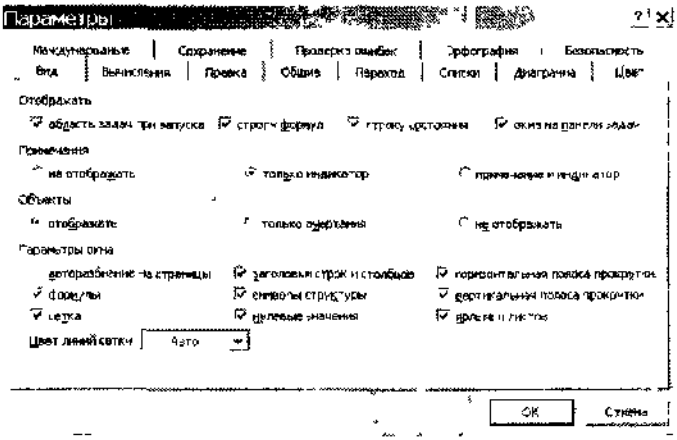
1. В операционной системе Windows запустить электронные таблицы Microsoft Excel командой [*Программы-Microsoft Excel*] или электронные таблицы OpenOffice.org Calc командой [*Программы-OpenOffice-OpenOffice Calc*].
   Или:
   в операционной системе Linux запустить электронные таблицы OpenOffice.org Calc командой [*Офис-OpenOffice Calc*].
   В созданном документе присвоить листу имя *Ссылки*.

Введем в ячейки диапазона C1:C3 формулы, содержащие относительные, абсолютные и смешанные ссылки.

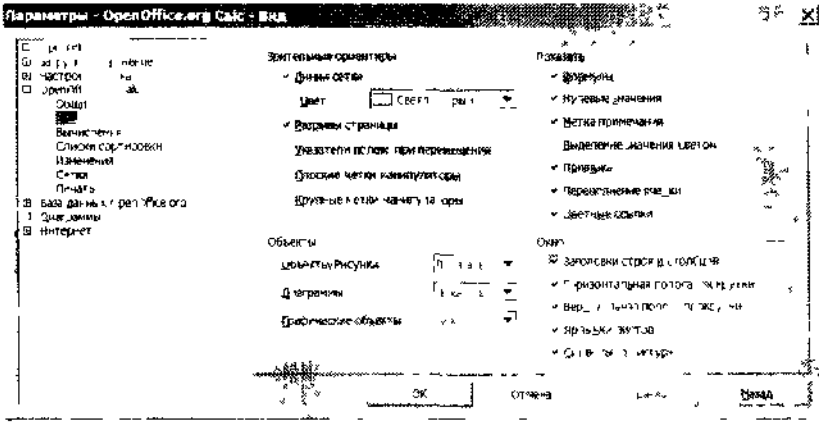
2. Ввести:
   - в ячейку C1 формулу =A1+B1, содержащую относительные ссылки;
   - в ячейку C2 формулу =$A$1+$B$1, содержащую абсолютные ссылки;
   - в ячейку C3 формулу =$A1+B$1, содержащую смешанные ссылки.

Отобразим формулы в ячейках.

3. В электронных таблицах Microsoft Excel для отображения в ячейках не чисел, а формул ввести команду [*Сервис-Параметры...*] и в появившемся диалоговом окне *Параметры* на вкладке *Вид* в разделе *Параметры* окна установить флажок *Формулы*.

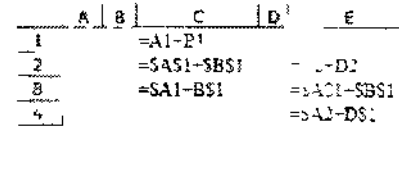


В электронных таблицах OpenOffice.org Calc для отображения в ячейках не чисел, а формул ввести команду [*Сервис- Параметры...*] и в появившемся диалоговом окне *Параметры OpenOffice Calc Вид* в разделе *Показать* установить флажок *Формулы*.



Скопируем формулы из диапазона ячеек C1:C3 в диапазон ячеек E2:E4.

4. Выделить диапазон ячеек C1:C3 и ввести команду [*Правка-Копировать*]. Выделить диапазон ячеек E2:E4 и ввести команду [*Правка-Вставить*].



Лист Ссылки находится в файлах Calc.xls и Calc.ods, которые хранятся в папке ..\IIKT9\Calc **Windows-CD**

Практическая работа 3.3

Создание таблиц значений функций в электронных таблицах

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться создавать таблицы значений функций в заданном диапазоне значений аргумента и с заданным шагом его изменения.

Задание. В электронных таблицах Microsoft Excel и OpenOffice.org Calc создать таблицы значений квадратичной функции $y = x^2 - 3$ и функции квадратного корня $y = \sqrt{x+4}$ на отрезке $[-4; 4]$ с шагом 1.

Создание таблиц значений функций в электронных таблицах Microsoft Excel и OpenOffice.org Calc

1. В операционной системе Windows запустить электронные таблицы Microsoft Excel командой [*Программы Microsoft Excel*] или электронные таблицы OpenOffice.org Calc командой [*Программы-OpenOffice OpenOffice Calc*].
Или:
в операционной системе Linux запустить электронные таблицы OpenOffice.org Calc командой [*Офис-OpenOffice Calc*].
В созданном документе присвоить листу имя *Функция*.

Введем значения аргумента функции в первую строку электронных таблиц.

2. В ячейку A1 ввести название строки значений аргумента (например, x), а в ячейку B1 минимальное значение аргумента (число -4).

В ячейку C1 ввести формулу =B1+1.
Выделить диапазон ячеек C1:J1 и скопировать формулу во все ячейки этого диапазона с помощью команды [*Правка-Заполнить-Вправо*].

Введем значения функции $y = x^2 - 3$ во вторую строку электронной таблицы. Ввод формулы для вычисления функции произвести с клавиатуры.

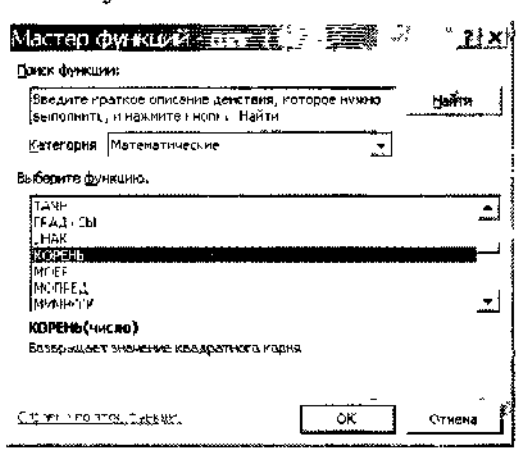
3. В ячейку A2 ввести название строки значений функции (например, $y = x^2-3$).
В ячейку B2 ввести формулу =B1^2–3.
Выделить диапазон ячеек B2:J2 и скопировать формулу во все ячейки этого диапазона с помощью команды [*Правка-Заполнить-Вправо*].

Введем значения функции $y = \sqrt{x+4}$ в третью строку электронной таблицы.

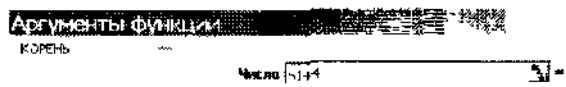
4. В ячейку A3 ввести название строки значений функции (например, *y =КОРЕНЬ(x+4)*).

В электронных таблицах Microsoft Excel для ввода функций воспользуемся *Мастером функций*. (По форме он несколько отличается от *Мастера функций* электронных таблиц OpenOffice.org Calc.)

5. Выделить ячейку B3 и ввести команду [*Вставка-Функция...*]. В диалоговом окне *Мастер функций – шаг 1 из 2* в списке *Категория:* выбрать вариант *Математические*, а в списке *Выберите функцию:* выбрать вариант *Корень*.
Нажать кнопку *ОК*.



6. На появившейся панели *Аргументы функции* в поле *Число* вести B1+4.
Нажать кнопку *ОК*.



7. Выделить диапазон ячеек B3:J3 и скопировать формулу во все ячейки этого диапазона с помощью команды [*Правка Заполнить Вправо*]. В результате будет получена таблица значений функций.
8. Для отображения в ячейках чисел с заданной точностью выделить диапазон ячеек и ввести команду [*Формат-Ячейки...*].
В появившемся диалоговом окне *Формат ячеек* на вкладке *Число* выбрать в списке *Числовые форматы* формат *Числовой* и установить с помощью счетчика *Число десятичных знаков:* 1.

	A	B	C	D	E	F	G	H	I	J
1	*x*	-4	-3	-2	-1	0	1	2	3	4
2	*y = x^2-3*	13	6	1	-2	-3	-2	1	6	13
3	*y =КОРЕНЬ(x-4)*	0.0	1.0	1.4	1.7	2.0	2.2	2.4	2.6	2.8

Лист *Функция* находится в файлах Calc.xls и Calc.ods, которые хранятся в папке ..\IIKT9\Calc **Windows-CD**

Практическая работа 3.4
Построение диаграмм различных типов

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться строить линейчатые и круговые диаграммы, а также диаграммы типа график.

Задание 1. В электронных таблицах построить на листе с данными линейчатую диаграмму с вертикальными столбцами (гистограмму) с легендой, позволяющую сравнить численность населения в семи наиболее населенных странах мира.

	A	B
1	**Страна**	**Население, млн**
2	Китай	1273
3	Индия	1030
4	США	279
5	Индонезия	228
6	Бразилия	175
7	Россия	146
8	Бангладеш	131

Задание 2. В электронных таблицах построить круговую диаграмму без легенды, позволяющую наглядно представить долю стоимости каждого устройства в общей стоимости компьютера.

	A	B
1	**Устройство**	**Стоимость, руб.**
2	Системная плата	2200
3	Процессор	2100
4	Оперативная память	1000
5	Жесткий диск	2500
6	Монитор	5500
7	Дисковод DVD-RW	900
8	Корпус	1000
9	Клавиатура	450
10	Мышь	150

Задание 3. В электронных таблицах построить на листе с данными графики квадратичной функции $y = x^2 - 3$ и функции квадратного корня $y = \sqrt{x+4}$ с легендой.

Задание 1. Построение линейчатой диаграммы с вертикальными столбцами с легендой в электронных таблицах Microsoft Excel

1. В операционной системе Windows запустить электронные таблицы Microsoft Excel командой [*Программы-Microsoft Excel*].

В созданном документе присвоить листу имя *Линейчатая диаграмма*.

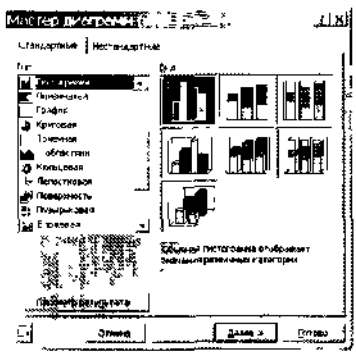
На листе с данными построим линейчатую диаграмму с вертикальными столбцами (гистограмму) с легендой.

2. Вставить в электронные таблицы данные из таблицы, содержащей численность населения некоторых стран мира.

Для создания диаграмм используется *Мастер диаграмм*. Он позволяет создавать диаграмму по шагам с помощью серии диалоговых окон.

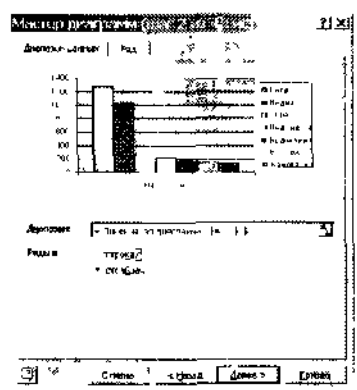
3. Выделить диапазон ячеек A1:B8, содержащий исходные данные. Запустить *Мастер диаграмм* с помощью команды [*Вставка-Диаграмма...*].

4. В появившемся диалоговом окне *Мастер диаграмм (шаг 1 из 4)* в списке *Тип:* необходимо выбрать *Гистограмма*. Гистограммы могут быть различных видов (плоские, объемные и т. д.), в окне *Вид:* выбрать плоскую диаграмму. Щелкнуть по кнопке *Далее>*.



Определим, в строках или столбцах хранятся названия категорий и ряд данных, а также уточним, в какой строке или столбце содержатся категории.

5. В появившемся диалоговом окне *Мастер диаграмм (шаг 2 из 4)* на вкладке *Диапазон данных* с помощью переключателя *Ряды в:* выбрать *столбцах*.



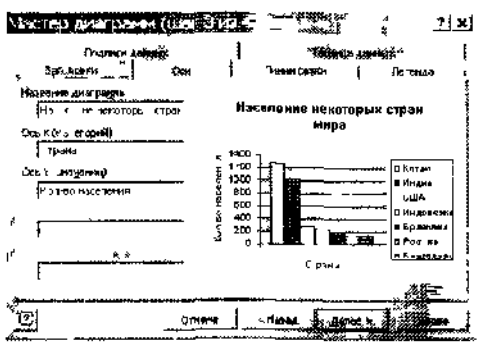
В окне появится изображение диаграммы, в которой исходные данные для рядов данных и категорий берутся из столбцов таблицы.

Справа от диаграммы появляется *Легенда*, которая содержит необходимые пояснения к диаграмме.

Щелкнуть по кнопке *Далее>*.

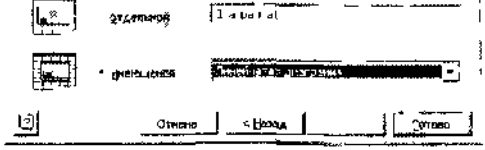
Настроим внешний вид диаграммы: введем заголовок диаграммы (например, *Население некоторых стран мира*) и названия оси категорий (например, *Страна*) и оси значений (например, *Кол во населения*) и определим наличие горизонтальных линий сетки. Для идентификации столбцов вместо вывода под столбцами названий стран (категорий) удобнее использовать легенду.

6. В появившемся окне *Мастер диаграмм (шаг 3 из 4)* на вкладке *Заголовки* ввести в соответствующие поля название диаграммы, а также названия оси категорий и оси значений.
На других вкладках можно уточнить детали отображения диаграммы (шрифт, цвета, подписи и т. д.).
Щелкнуть по кнопке *Далее>*.

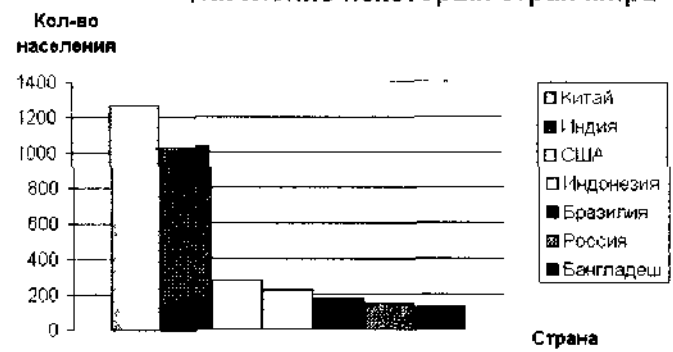


Выберем вариант размещения диаграммы (например, на листе с данными).

7. На появившейся диалоговой панели *Мастер диаграмм (шаг 4 из 4)* с помощью переключателя *Поместить диаграмму на листе:* выбрать *имеющемся*.
Щелкнуть по кнопке *Готово*.



8. В результате на листе с данными *Линейчатая диаграмма* получим гистограмму с легендой, в которой высота столбцов пропорциональна численности населения в странах.



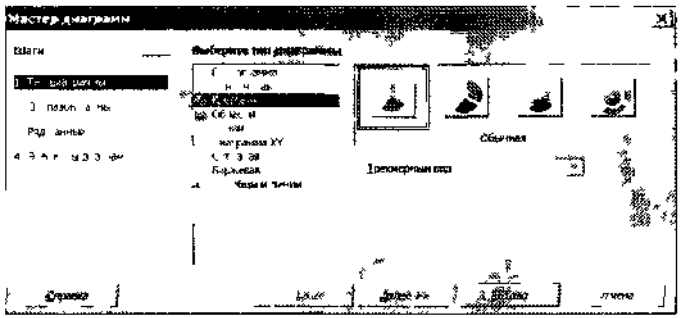
Лист *Линейчатая диаграмма* находится в файлах Calc.xls и Calc.ods, которые хранятся в папке ..\IIKT9\Calc **Windows-CD**

Задание 2. Построение круговой диаграммы без легенды в электронных таблицах OpenOffice.org Calc

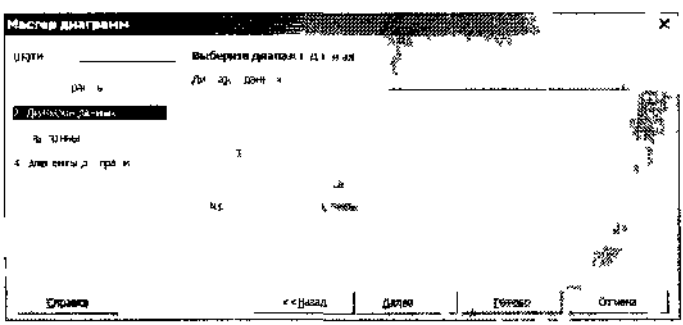
1. В операционной системе Windows или Linux запустить электронные таблицы OpenOffice.org Calc соответственно командой [*Программы-OpenOffice-OpenOffice Calc*] или [*Офис-OpenOffice Calc*].
   В созданном документе присвоить листу имя *Круговая диаграмма*.
2. Вставить в электронные таблицы данные из таблицы, содержащей стоимость устройств компьютера.

Рассмотрим использование *Мастера диаграмм* для построения круговой диаграммы, позволяющей наглядно представить долю стоимости каждого устройства в общей стоимости компьютера.

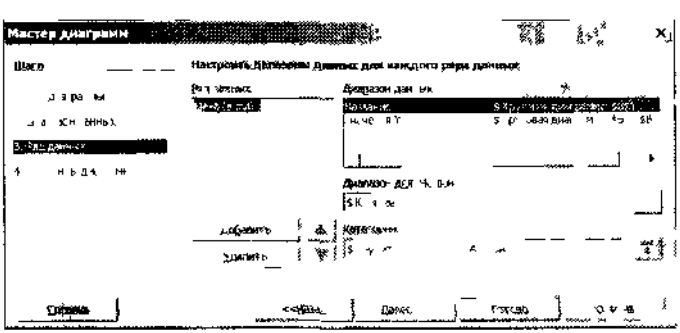
3. Выделить диапазон ячеек A1:B10, содержащий исходные данные.
4. Запустить *Мастер диаграмм* с помощью команды [*Вставка Диаграмма...*]. В появившемся диалоговом окне *Мастер диаграмм* на шаге *Тип диаграммы* необходимо выбрать *Круговая*, а в окне выбрать плоскую диаграмму.



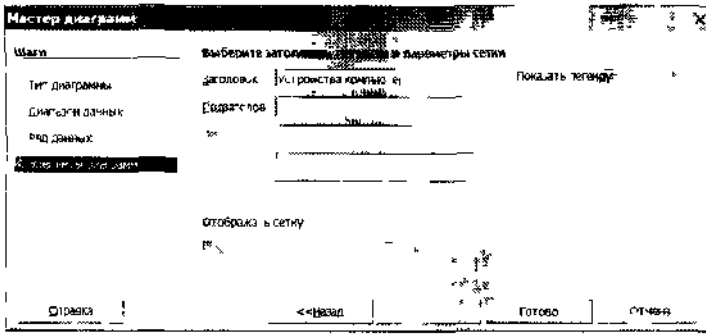
5. На шаге *Диапазон данных* с помощью переключателя установить *Ряды данных в столбцах*.
В окне появится изображение диаграммы, в которой исходные данные для рядов данных и категорий берутся из столбцов таблицы.



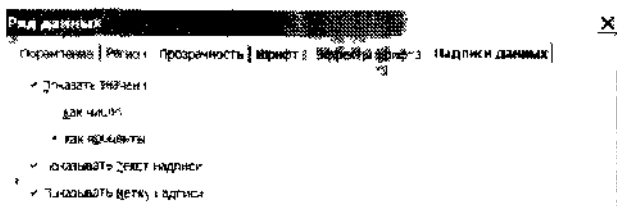
6. На шаге *Ряд данных* установить, в каких столбцах размещены ряды данных и категорий.



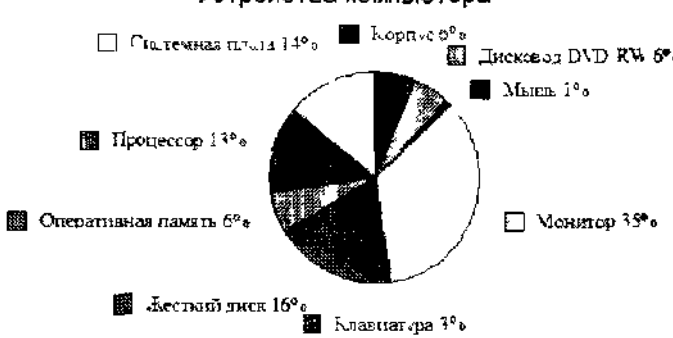
7. На шаге *Элементы диаграмм* ввести заголовок диаграммы и снять флажок *Показать легенду*.



8. В контекстном меню диаграммы выбрать пункт *Свойства объекта* и в окне *Ряд данных* на вкладке *Надписи данных* установить параметры надписей секторов круговой диаграммы.



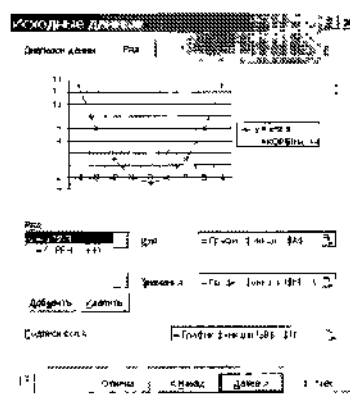
9. В результате получим круговую диаграмму без легенды, но зато с указанием процентов стоимости отдельных устройств в общей стоимости компьютера.



Лист *Круговая диаграмма* находится в файлах Calc.xls и Calc.ods, которые хранятся в папке ..\IIKT9\Calc **Windows-CD**

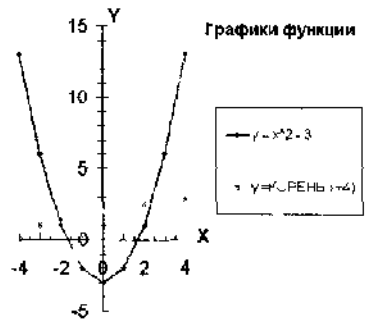
Задание 3. Построение диаграммы типа график с легендой в электронных таблицах Microsoft Excel

1. В операционной системе Windows запустить электронные таблицы Microsoft Excel командой [*Программы-Microsoft Excel*].
   В созданном документе присвоить листу имя *График функции*.
2. Для вставки в электронные таблицы данных воспользоваться результатами практической работы 3.3 «Создание таблиц значений функций в электронных таблицах».
   Скопировать данные с листа *Функция* на лист *График функции*.
3. Выделить диапазон ячеек B1:J3, содержащий в качестве исходных данных значения функций. Запустить *Мастер диаграмм* с помощью команды [*Вставка-Диаграмма...*].
4. В появившемся диалоговом окне *Мастер диаграмм (шаг 1 из 4)* в списке *Тип:* необходимо выбрать *График*, а в окне *Вид*: выбрать *график с маркерами*.
   Щелкнуть по кнопке *Далее>*.
5. В появившемся диалоговом окне *Мастер диаграмм (шаг 2 из 4)* на вкладке *Диапазон данных* с помощью переключателя *Ряды в:* выбрать *строках*.
   На вкладке *Ряд* выбрать для *подписей категорий значения* аргумента: в текстовое окно *Подписи оси X:* ввести данные путем щелчка по кнопке со стрелочкой и выделения в таблице диапазона ячеек B1:J1.
   Щелкнуть по кнопке *Далее>*.



6. В появившемся диалоговом окне *Мастер диаграмм (шаг 3 из 4)* на вкладке *Заголовки* ввести в соответствующие поля название диаграммы, а также названия оси категорий и оси значений.
   На других вкладках можно уточнить детали отображения диаграммы (формат осей, наличие и вид сетки и т. д.).
   Щелкнуть по кнопке *Далее>*.

7. В появившемся диалоговом окне *Мастер диаграмм (шаг 4 из 4)* с помощью переключателя *Поместить диаграмму на листе:* выбрать *имеющемся.* Щелкнуть по кнопке *Готово.*
8. В результате на листе *График функции* будут построены два графика функций $y = x^2 - 3$ и $y = \sqrt{x} + 4$, маркеры которых имеют координаты y, равные значениям рядов данных, и координаты x, равные значениям ряда категорий.



Лист *График функции* находится в файлах Calc.xls и Calc.ods, которые хранятся в папке ..\IIKT9\Calc

Windows-CD

Практическая работа 3.5

Сортировка и поиск данных в электронных таблицах

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows и Linux.

Цель работы. Научиться осуществлять в электронных таблицах сортировку данных в выделенном столбце, вложенную сортировку записей базы данных по нескольким столбцам и поиск данных.

Задание 1. В электронные таблицы внести данные из табл. 3.8 и произвести сортировку данных по убыванию в столбцах A и C, содержащих числа и даты, а также сортировку по возрастанию в столбцах B и D, содержащих текст и время

Таблица 3.8. Сортировка чисел, текста, дат и времени в столбцах

Задание 2. В электронные таблицы внести базу данных «Процессоры» из таблицы и произвести вложенную сортировку по возрастанию для числового поля *Частота* и числового поля *Технология.*

Процессоры

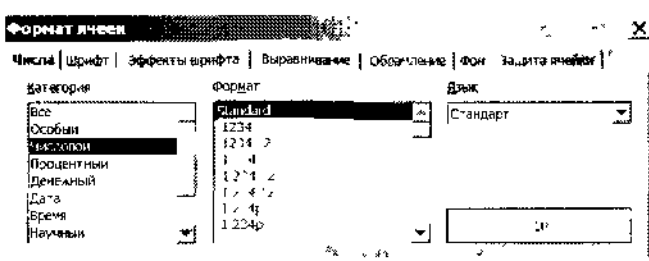
№	Тип процессора	Частота (ГГц)	Технология (мк)
1	Intel Core 4 Quad	2,6	0,065
2	Intel Core 2 Duo	3,0	0,065
3	Intel Celeron	2,8	0,09
4	Intel Pentium 4	3,0	0,065
5	AMD Athlon	3,0	0,09
6	AMD Sempron	2,0	0,09

Задание 3. В электронных таблицах осуществить поиск записей в базе данных «Процессоры» с помощью составного фильтра, состоящего из двух условий: для поля *Частота (ГГц)* условие *=3*, для поля *Технология (мк)* условие *=0,065*.

Задание 1. Сортировка данных в столбцах электронных таблиц OpenOffice.org Calc

1. В операционной системе Windows или Linux запустить электронные таблицы OpenOffice.org Calc соответственно командой [*Программы-OpenOffice-OpenOffice Calc*] или командой [*Офис-OpenOffice Calc*].
   В созданном документе присвоить листу имя *Сортировка данных*.

   Внесем в электронную таблицу данные из табл. 3.8.
2. Выделить столбец A и ввести команду [*Формат-Ячейка...*].
   В появившемся диалоговом окне *Формат ячеек* установить на вкладке *Числа* с помощью списка *Категория* числовой тип данных.
   С помощью аналогичной процедуры установить требуемый тип данных для столбцов B, C и D.



Произведем сортировку строк по убыванию в столбцах, содержащих числа и даты, и сортировку строк по возрастанию в столбцах, содержащих текст и время.

3. Выделить столбец A с числовыми данными и ввести команду [*Данные-Сортировка...*].
   В появившемся диалоговом окне *Сортировка* на вкладке *Условия сортировки* выбрать в раскрывающемся списке *Столбец A* и установить переключатель в положение *По убыванию*.

4. Для столбцов B, C и D повторить пункт 3 с учетом направления сортировки.
   В результате данные будут отсортированы в одних столбцах по возрастанию, а в других — по убыванию.

	A	B	C	D
1	5	я	среда, Январь 12, 2005	4:30
2	1	5	среда, Март 03, 2004	8:30
3	0	$	понедельник, Январь 12, 2004	12:30
4	5	бг	понедельник, Март 03, 2003	16:30
5	-10	бит	суббота, Январь 01, 2000	20:30

Лист *Сортировка данных* находится в файлах Calc.xls и Calc.ods, которые хранятся в папке ..\IIKT9\Calc **Windows-CD**

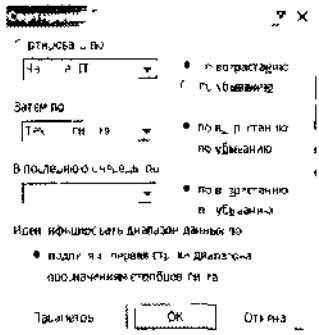
Задание 2. Вложенная сортировка записей базы данных в электронных таблицах Microsoft Excel

1. В операционной системе Windows запустить электронные таблицы Microsoft Excel командой [*Программы-Microsoft Excel*].
   В созданном документе присвоить листу имя *Вложенная сортировка*.

Внесем в электронные таблицы базу данных «Процессоры». Осуществим в электронных таблицах вложенную сортировку базы данных.

2. Заполнить столбцы A, B, C и D данными. Выделить их и ввести команду [*Данные-Сортировка...*]

3. В появившемся диалоговом окне *Сортировка диапазона* в списке *Сортировать по* выбрать *Частота (ГГц)* и установить переключатель в положение *по возрастанию*.
В списке *Затем по* выбрать *Технология (мк)* и установить переключатель в положение *по возрастанию*.
Установить переключатель *Идентифицировать диапазон данных по* в положение *подписям*.



4. После щелчка по кнопке *ОК* будет осуществлена вложенная сортировка по двум столбцам.

	A	B	C	D
	№		Частота (ГГц)	Технология (мк)
1		Тип процессора		
2		AMD Sempron		
3		Intel Core 2 Quad		
4		Intel Celeron		
5		Intel Core 2 Duo		
6		Intel Pentium 4		0,065
7		AMD Athlon		0,09

Лист *Вложенная сортировка* находится в файлах *Calc.xls* и *Calc.ods*, которые хранятся в папке ..\IIKT9\Calc **Windows-CD**

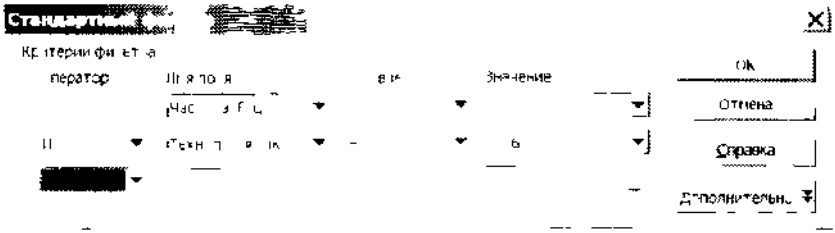
Задание 3. Поиск данных с помощью составного фильтра в электронных таблицах OpenOffice.org Calc

1. В операционной системе Windows или Linux запустить электронные таблицы OpenOffice.org Calc соответственно командой [*Программы OpenOffice OpenOffice Calc*] или [*Офис OpenOffice Calc*].

В созданном документе присвоить листу имя *Поиск*.

Осуществим поиск записей базы данных с помощью фильтра, состоящего из трех условий.

2. Выделить столбцы с данными и ввести команду [*Данные Фильтр Стандартный фильтр*].
   В появившемся диалоговом окне *Стандартный фильтр* в раскрывающихся списках *Имя поля* последовательно ввести заголовки столбцов *Частота (ГГц)* и *Технология (мк)*.
   Фильтр должен искать записи, удовлетворяющие одновременно двум условиям, поэтому в списке *Оператор* выбрать *И*.
   В раскрывающихся списках *Условие* ввести условие =.
   В раскрывающихся списках *Значение* последовательно ввести 3 и 0,065.



3. В результате в базе данных будут найдены и показаны в таблице две записи (№ 2 и № 4), удовлетворяющие заданному фильтру.

	A	B	C	D
1	№	Тип процессора	Частота (ГГц)	Технология (мк)
3	2	Intel Core 2 Duo	3,0	0,65
5	4	Intel Pentium 4	3,0	0,065

Лист *Поиск* находится в файлах Calc.xls и Calc.ods, которые хранятся в папке ..\IIKT9\Calc\ Windows-CD

Практические работы к главе 4 «Алгоритмизация и основы объектно-ориентированного программирования»

	Установить: • систему объектно-ориентированного программировая Visual Basic 2005; • систему алгоритмического программирования Basic, входящую в OpenOffice.org	VisualStudio-CD Windows-CD
	Установить: • систему объектно-ориентированного программирования Gambas; • систему алгоритмического программирования Basic, входящую в OpenOffice.org	Linux-DVD

Практическая работа 4.1

Знакомство с системами объектно-ориентированного и алгоритмического программирования

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Знакомство с системами объектно-ориентированного и алгоритмического программирования.

Задание. Познакомиться с системами объектно-ориентированного программирования Visual Basic 2005 (Windows) и Gambas (Linux) и системой алгоритмического программирования Basic, входящей в интегрированный пакет OpenOffice.org.

Знакомство с системой объектно-ориентированного программирования Visual Basic 2005

1. Запустить систему объектно-ориентированного программирования Visual Basic 2005 командой [*Программы-Visual Basic 2005 Express Edition*].

После запуска системы программирования ввести команду [*Файл-Создать проект...*] ([*File-New project...*]). В появившемся диалоговом окне *Новый проект* (*New project*) (рис. 4.1.1) выбрать тип создаваемого проекта *Приложение Windows* (*Windows Application*) и в текстовое поле *Имя* (*Name*) ввести название проекта, например *Первый проект*. Щелкнуть по кнопке *OK*.

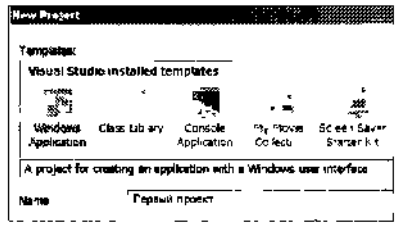


Рис. 4.1.1. Создание проекта в системе программирования Visual Basic 2005

После этого появится окно системы программирования, включающее несколько окон (рис. 4.1.2).

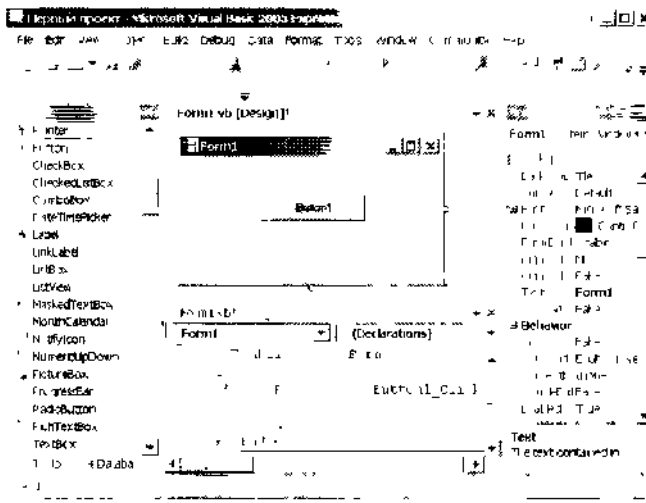


Рис. 4.1.2. Система объектно-ориентированного программирования Visual Basic 2005

Визуальное конструирование графического интерфейса проекта выполняется в окне *Конструктор форм* (*Designer*). Оно располагается в центре окна системы программирования и содержит форму (в данном случае `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 объектов* (*ToolBox*), которая размещается в левой части окна системы программирования.

С формой связан программный код проекта, для ввода и редактирования которого служит окно *Программный код* (*Code*) (на рис. 4.1.2 размещено под окном *Конструктор форм*).

Для перехода в окно *Программный код* применяется команда [*Вид-Код*] ([*View-Code*]), а для обратного перехода в окно конструирования графического интерфейса *Конструктор форм* применяется команда [*Вид-Конструктор*] ([*View-Designer*]).

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.

Этапы разработки проекта. Создание проектов в системе объектно-ориентированного визуального программирования Visual Basic 2005 реализуется в режиме [*design*]. Создание проекта можно разделить на несколько этапов.

- Создание графического интерфейса проекта. В окне *Конструктор форм* с использованием *Панели объектов* на форму помещаются элементы управления, которые должны обеспечить взаимодействие проекта с пользователем.
- Установка значений свойств объектов графического интерфейса. С помощью окна *Свойства* задаются значения свойств элементов управления, помещенных ранее на форму.
- Создание и редактирование программного кода. В окне *Редактор кода* производится ввод и редактирование программного кода проекта.
- Сохранение проекта. Так как проекты включают в себя несколько файлов, необходимо каждый проект сохранять в отдельной папке. Сохранение проекта производится командой [*Файл-Сохранить все*] ([*File-Save all*]).

В появившемся диалоговом окне *Сохранить проект* (*Save Project*) (рис. 4.1.3) в текстовом поле *Имя:* (*Name:*) можно уточнить имя проекта. Ввести путь к папке проекта можно в текстовом поле *Расположение:* (*Location:*) или выбрать ее расположение в файловой системе после щелчка по кнопке *Обзор...* (*Browse...*).

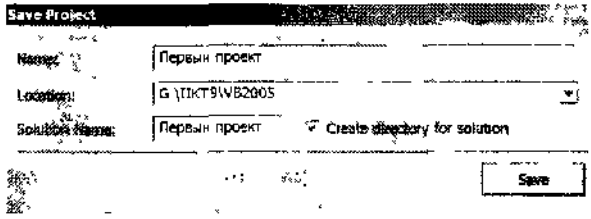


Рис. 4.1.3. Сохранение проекта

Выполнение проекта. Загрузка проекта в систему программирования Visual Basic 2005 производится путем активизации в папке проекта основного файла проекта (файла с расширением vbproj).

Запуск проекта на выполнение производится командой [*Отладка-Начать*] ([*Debug-Start*]) или щелчком по кнопке ▶ на панели инструментов окна системы программирования. После этого система программирования переходит в режим выполнения проекта [*run*], в котором редактирование графического интерфейса или программного кода невозможно. В процессе выполнения проекта производится его компиляция в приложение, которое имеет расширение exe и находится в папке ...\bin.

Для окончания выполнения проекта и перехода в режим конструирования проекта [*design*] необходимо ввести команду [*Отладка-Остановить отладку*] ([*Debug-Stop Debugging*]) или щелкнуть по кнопке ■ на панели инструментов окна системы программирования.

Знакомство с системой объектно-ориентированного программирования Gambas

1. Запустить систему объектно-ориентированного программирования Gambas командой [*Программы-Средства разработки-Gambas*].

После этого появится окно системы программирования, включающее несколько окон (см. рис. 4.1.4).

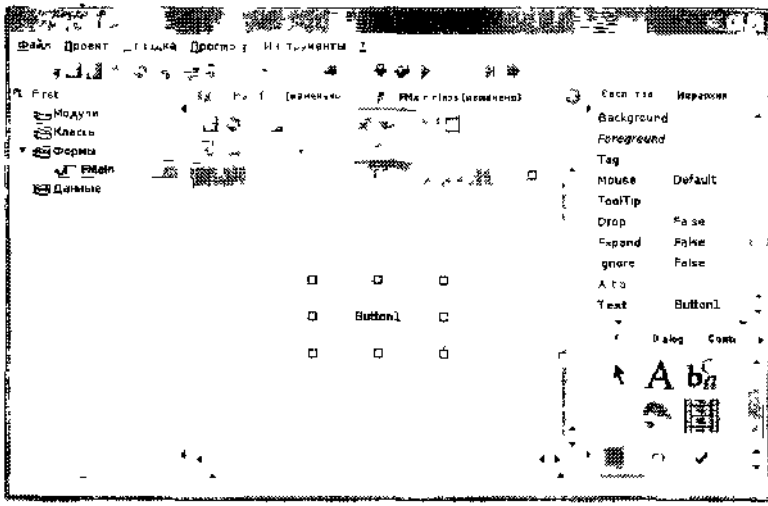


Рис. 4.1.4. Система объектно-ориентированного программирования Gambas

Визуальное конструирование графического интерфейса проекта выполняется в центре окна системы программирования (на вкладке *FMain form*) и содержит **форму**, 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 объектов*, которая вызывается командой [*Просмотр Палитра компонентов*] и размещается в правой нижней части окна системы программирования.

С формой связан программный код проекта, для ввода и редактирования которого служит окно *Программный код* (размещено на вкладке *FMain class*).

Для перехода с конструирования графического интерфейса на редактирование программного кода и обратно активизируется соответствующая вкладка.

Справа располагается окно *Свойства*. 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.

Этапы разработки проекта. Создание проектов в системе объектно-ориентированного визуального программирования Gambas можно разделить на несколько этапов.

- Создание графического интерфейса проекта. В окне *Конструктор форм* с использованием *Панели объектов* на форму помещаются элементы управления, которые должны обеспечить взаимодействие проекта с пользователем.
- Установка значений свойств объектов графического интерфейса. С помощью окна *Свойства* задаются значения свойств элементов управления, помещенных ранее на форму.
- Создание и редактирование программного кода. В окне *Редактор кода* производится ввод и редактирование программного кода проекта.
- Сохранение проекта. Так как проекты включают в себя несколько файлов, необходимо каждый проект сохранять в отдельной папке. Сохранение проекта производится командой [*Файл-Сохранить все*].
- Компиляция проекта в приложение. Компиляция проекта в приложение производится командой [*Проект-Компилировать*].

Выполнение проекта. Открытие проекта в системе программирования Gambas производится путем активизации в папке проекта *основного файла проекта* (*файла с расширением* vbproj).

Запуск проекта на выполнение производится командой [*Отладка Старт*] или щелчком по кнопке ▶ на панели инструментов окна системы программирования. После этого система программирования переходит в режим выполнения проекта, в котором редактирование графического интерфейса или программного кода невозможно.

Для окончания выполнения проекта и перехода в режим конструирования проекта необходимо ввести команду [*Отладка-Стоп*] или щелкнуть по кнопке ■ на панели инструментов окна системы программирования.

Знакомство с системой алгоритмического программирования OpenOffice.org Basic

1. В операционной системе Windows или Linux запустить один из компонентов интегрированного офисного приложения OpenOffice.org (например, OpenOffice.org Writer) командой [*Программы OpenOffice.org Writer*].
2. В меню приложения ввести команду [*Сервис Макросы-Управление макросами-OpenOffice.org Бэйсик*].

Появится диалоговое окно *Макрос OpenOffice.org Basic*. Нажать кнопку *Создать*.

3. В появившемся диалоговом окне интерпретатора Basic ввести программный год на алгоритмическом языке Basic, между служебными словами **Sub** и **End Sub**.

4. Сохранение программы на языке Basic производится нажатием кнопки *Сохранить Basic*.
5. Вставка программы на языке Basic производится нажатием кнопки *Вставить код на Basic*.
6. Запуск на выполнение программы на языке Basic производится нажатием кнопки *Выполнить Basic*.

Практическая работа 4.2

Проект «Переменные»

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться использовать переменные разных типов в системах объектно-ориентированного и алгоритмического программирования.

Задание. Создать проект, в котором объявить переменные различных типов, присвоить переменным A и B значения, переменным разных типов C, D и F присвоить значения арифметического выражения A/B, вывести значения переменных C, D и F.

Проект «Переменные» на языках объектно-ориентированного программирования Visual Basic 2005 или Gambas

1. В операционной системе Windows запустить систему объектно-ориентированного программирования Visual Basic 2005 командой [*Программы-Visual Basic 2005 Express Edition*].

Или:
в операционной системе Linux запустить систему объектно-ориентированного программирования Gambas командой [*Программы-Средства разработки Gambas*].

Создадим графический интерфейс. Окно *Конструктор форм* (*Designer*) содержит форму (в данном случае Form1), являющуюся основой графического интерфейса проекта. На форму можно поместить различные элементы управления: метки (Label), кнопки (Button) и т. д. Пиктограммы элементов управления располагаются на *Панели объектов* (*ToolBox*), которая размещается в левой части окна системы программирования.

2. Поместить на форму (рис. 4.2.1 и 4.2.2):
   - три метки Label1, Label2 и Label3 для вывода значений переменных;
   - кнопку Button1 для запуска обработчика события.

С помощью окна *Свойства* (*Properties*) изменим установленные по умолчанию значения свойств управляющих элементов.

3. Изменить значение свойства Text:
   - форма — Form1 на Переменные;
   - кнопка — Button1 на Вычислить;

Объявим переменные. Щелкнем по кнопке, и в окне *Программный код* создадим обработчик события, реализующий линейный алгоритм:

- присвоить переменным A и B значения;
- присвоить переменным разных типов C, D и F значения арифметического выражения A/B;
- вывести значения переменных C, D и F на метки, присвоив их значения свойству Text.

```
Dim A, B As Byte, C As Integer, D As
Single, F As Double (As Float)
Private Sub Button1_Click(…)
A = 1
B = 3
C = A / B
D = A / B
F = A / B
Label1.Text = C
Label2.Text = D
Label3.Text = F
End Sub
```

4. Запустить проект на выполнение. После щелчка по кнопке с надписью *Вычислить* начнет выполняться обработчик события.
   Результат парадоксален с точки зрения математики, на метки выводятся различные значения арифметического выражения, зависящие от типа используемой переменной.

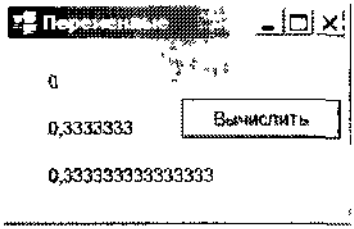


Рис. 4.2.1. Проект «Переменные» на языке Visual Basic 2005

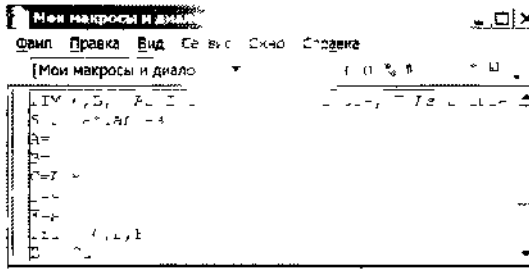
Рис. 4.2.2. Проект «Переменные» на языке Gambas

Проект «Переменные» на языке Visual Basic хранится в папке ..\IIKT9\VB2005\Переменные
Проект «Переменные» на языке Gambas хранится в папке ..\IIKT9\Gambas\Variables

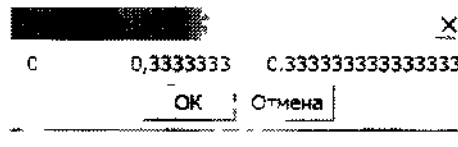
Windows-CD

Программа «Переменные» на языке алгоритмического программирования OpenOffice.org Basic

1. В операционной системе Windows или Linux запустить один из компонентов интегрированного офисного приложения OpenOffice.org (например, OpenOffice.org Writer) командой [*Программы-OpenOffice.org Writer*].
2. В меню приложения ввести команду [*Сервис-Макросы-Управление макросами-OpenOffice.org Бэйсик*].
   Появится диалоговое окно *Макрос OpenOffice.org Basic*. Нажать кнопку *Создать*.
3. В появившемся диалоговом окне интерпретатора Basic ввести программный код на алгоритмическом языке Basic, между служебными словами **Sub** и **End Sub**.



4. Запустить на выполнение программу на языке Basic нажатием кнопки *Выполнить Basic*.
Появится окно с результатами выполнения программы



Программа хранится в файле ..\IIKT9\Basic\Variables.bas **Windows-CD**

Практическая работа 4.3
Проект «Калькулятор»

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться вычислять с использованием четырех арифметических действия и математических функций в системах объектно-ориентированного и алгоритмического программирования.

Задание. Разработать проект «Калькулятор», который позволит производить четыре арифметических действия над числами (сложение, вычитание, умножение и деление), находить синус и квадратный корень

Проект «Калькулятор» на языках объектно-ориентированного программирования Visual Basic 2005 или Gambas

1. В операционной системе Windows запустить систему объектно-ориентированного программирования Visual Basic 2005 командой [*Программы Visual Basic 2005 Express Edition*].

Или:
в операционной системе Linux запустить систему объектно-ориентированного программирования Gambas командой [*Программы Средства разработки-Gambas*].

Работа над проектом начинается с создания графического интерфейса, для этого на форму помещаются элементы управления.

2. Разместить на форме (рис. 4.3.1 и 4.3.2):
   - два текстовых поля `TextBox1` и `TextBox2` для ввода числовых данных;
   - надпись `Label1` для вывода результата;
   - шесть кнопок `Button1`, `Button2`, `Button3`, `Button4`, `Button5` и `Button6` для запуска обработчиков событий: сложения, вычитания, умножения, деления двух чисел, вычисление синуса и квадратного корня числа.

Следующим шагом является создание программного кода обработчиков событий. Двойной щелчок мышью по кнопке вызывает окно *Программный код* с пустой заготовкой обработчика событий. Обработчик события, реализующий сложение чисел `Button1_Click`, должен присвоить значению свойства `Text` надписи `Label1` сумму числовых значений, введенных в текстовые поля `TextBox1` и `TextBox2`. Обработчики событий вычитания, умножения и деления создаются аналогично.

Для преобразования строковых значений свойства `Text` текстовых полей в десятичные числа воспользуемся функцией `Val()`, аргументом которой является строка, а значением — число.

3.
```
Private Sub Button1_Click( )
Label1.Text = Val(TextBox1.Text) +
Val(TextBox2.Text)
End Sub
```

Обработчик события, реализующий вычисление квадратного корня (аналогично — вычисления синуса) примет вид (в Gambas используется функция `Sqr()`).

4.
```
Private Sub Button6_Click( )
Label1.Text = Math.Sqrt(Val(TextBox1.Text))
End Sub
```

Графический интерфейс проекта можно сделать более понятным и привлекательным. Для этого необходимо в режиме конструирования проекта последовательно выделить

объекты графического интерфейса и с помощью диалогового окна *Свойства* (*Properties*) установить новые значения некоторых свойств для каждого объекта:

5. • для объекта *форма* `Form1` изменить значение свойства `Text` и цвет (значение свойства `BackColor`);
   • для объектов *кнопка* `Button1`, `Button2`, `Button3`, `Button4`, `Button5` и `Button6` изменить значение свойства `Text`;
   • для объектов *текстовое поле* `TextBox1` и `TextBox2` установить выравнивание текста по правому краю (значение свойства `TextAlign`) и шрифт (значение свойства `Font`);
   • для всех объектов увеличить шрифт (значение свойства `Font`).

Изменения значений свойств объектов с помощью диалогового окна *Свойства* (*Properties*) могут производиться различными способами. В большинстве случаев нужно просто стереть старое значение свойства и ввести новое. Однако для ввода значений некоторых свойств используются раскрывающиеся списки или диалоговые окна. Так, автоматическое изменение размера метки (значение свойства `AutoSize`) устанавливается с использованием списка, а цвет фона (значение свойства `BackColor`) и шрифт (значение свойства `Font`) устанавливается с использованием диалоговых окон.

6. Запустить проект на выполнение. Ввести число (например, 2008) в первое текстовое поле и щелкнуть по кнопке вычисления синуса.

На метку будет выведен результат.

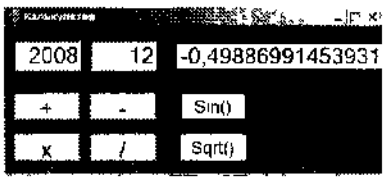


Рис. 4.3.1. Проект «Калькулятор» на языке Visual Basic

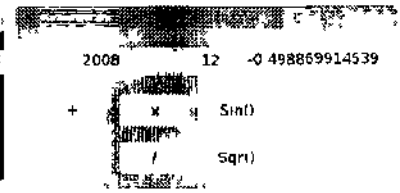


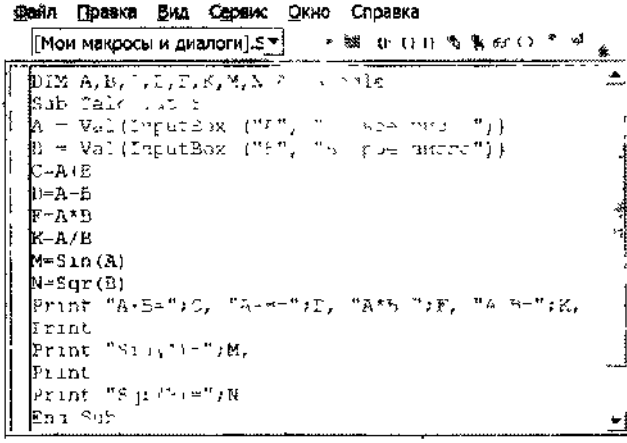
Рис. 4.3.2. Проект «Калькулятор» на языке Gambas

Проект «Калькулятор» на языке VisualBasic хранится в папке ..\IIKT9\VB2005\Калькулятор **Windows-CD**

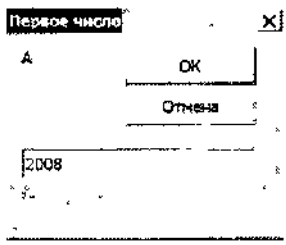
Проект «Переменные» на языке Gambas хранится в папке ..\IIKT9\Gambas\Calculator

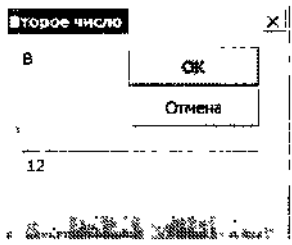
Программа «Калькулятор» на языке алгоритмического программирования OpenOffice.org Basic

1. В операционных системах Windows или Linux запустить один из компонентов интегрированного офисного приложения OpenOffice.org (например, OpenOffice.org Writer) командой [*Программы-OpenOffice.org Writer*].
2. В меню приложения ввести команду [*Сервис-Макросы-Управление макросами-OpenOffice.org Бэйсик*]. Появится диалоговое окно *Макрос OpenOffice.org Basic*. Нажать кнопку *Создать*.
3. В появившемся диалоговом окне интерпретатора Basic ввести программный год на алгоритмическом языке Basic, между служебными словами **Sub** и **End Sub**.

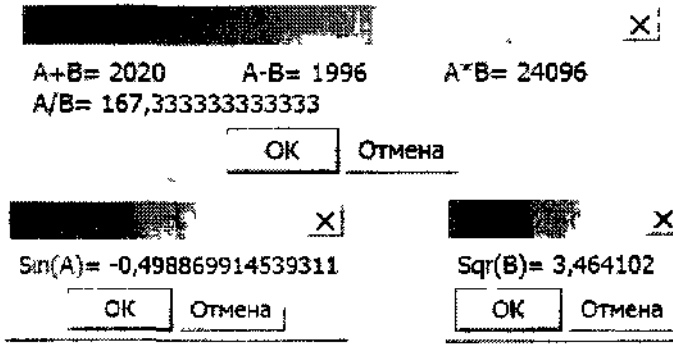


4. Запустить на выполнение программу на языке Basic нажатием кнопки *Выполнить Basic*.
В появившихся диалоговых окнах ввести первое (например, 2008) и второе (например, 12) числа и каждый раз нажимать на кнопку *OK*.





Появятся окна с результатами выполнения программы.



Программа хранится в файле ..\IIKT9\Basic\Calculator.bas **Windows-CD**

Практическая работа 4.4

Проект «Строковый калькулятор»

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться применять строковые функции в системах объектно-ориентированного и алгоритмического программирования.

Задание. Создать проект, который позволит производить *преобразования строк с использованием строковых* функций.

Проект «Строковый калькулятор» на языках объектно-ориентированного программирования Visual Basic 2005 или Gambas

1. В операционной системе Windows запустить систему объектно-ориентированного программирования Visual Basic 2005 командой [*Программы Visual Basic 2005 Express Edition*].
Или:
в операционной системе Linux запустить систему объектно-ориентированного программирования Gambas командой [*Программы-Средства разработки Gambas*].

Создадим графический интерфейс проекта.

2. Разместить на форме (рис. 4.4.1 и 4.4.2):
   - текстовое поле TextBox1 для ввода строкового аргумента Строка;
   - текстовое поле TextBox2 для ввода числового аргумента Позиция;
   - текстовое поле TextBox3 для ввода числового аргумента Длина;
   - надпись Label1 для вывода результата;
   - шесть кнопок для запуска обработчиков событий.

Создадим для каждой кнопки обработчик события, реализующий одну из строковых функций. Событийная процедура, реализующая функцию Mid(), будет иметь следующий вид.

3.
```
Private Sub Button1_Click( )
Label1.Text =
Microsoft.VisualBasic.Mid(TextBox1.Text,
Val(TextBox2.Text), Val(TextBox3.Text))
End Sub
```

Событийная процедура, реализующая функцию Asc(), будет иметь следующий вид.

4.
```
Private Sub Button5_Click( )
Label1.Text = Asc(TextBox1.Text)
End Sub
```

Событийные процедуры вырезания левой и правой подстрок, а также определения длины строки и преобразования строки в символ и символа в строку создаются аналогично.

5. Запустить проект, в первое поле ввести слово, (например, «информатика»), во второе поле — порядковый номер символа в слове (например, 3), в третье поле — количество вырезаемых символов (например, 5). Щелкнуть по кнопке *Mid()*. На метке появится вырезанная подстрока «форма».

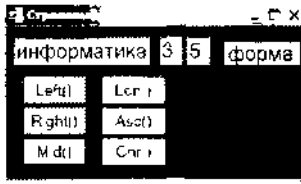


Рис. 4.4.1. Проект «Строковый калькулятор» на языке Visual Basic

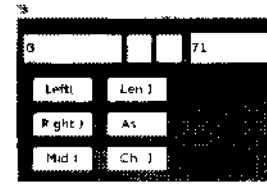


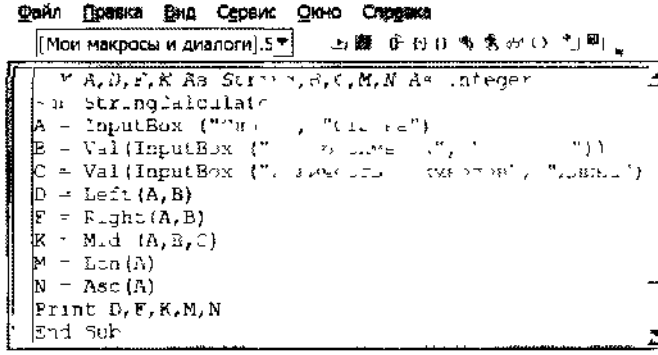
Рис. 4.4.2. Проект «Строковый калькулятор» на языке Gambas

Обратите внимание на то, что в языке Gambas функция `Asc()` возвращает десятичный числовой код символа в кодировке *ASCII*.

Проект «Строковый калькулятор» на языке Visual Basic хранится в папке ..\IIKT9\VB2005\Строковый калькулятор
Проект «Строковый калькулятор» на языке Gambas хранится в папке ..\IIKT9\Gambas\StringCalculator
Windows-CD

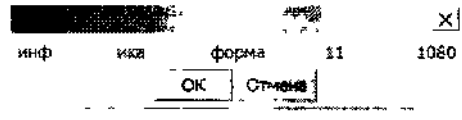
Программа «Строковый калькулятор» на языке алгоритмического программирования OpenOffice.org Basic

1. В операционной системе Windows или Linux запустить один из компонентов интегрированного офисного приложения OpenOffice.org (например, OpenOffice.org Writer) командой [*Программы OpenOffice.org Writer*].
2. В меню приложения ввести команду [*Сервис Макросы Управление макросами OpenOffice.org Бэйсик*]. Появится диалоговое окно *Макрос OpenOffice.org Basic*. Нажать кнопку *Создать*.
3. В появившемся диалоговом окне интерпретатора Basic ввести программный год на алгоритмическом языке Basic, между служебными словами **Sub** и **End Sub**.



4. Запустить на выполнение программу на языке Basic нажатием кнопки *Выполнить Basic*.
5. В последовательно появляющихся диалоговых окнах ввести слово (например, «информатика»), порядковый номер символа в слове (например, 3) и количество вырезаемых символов (например, 5).

Появится окно с результатами выполнения программы. Обратите внимание на то, что функция `Asc()` возвращает десятичный числовой код символа в кодировке *Unicode*.



Программа хранится в файле ..\IIKT9\Basic\StringCalculator.bas Windows-CD

Практическая работа 4.5

Проект «Даты и время»

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться применять оператор цикла с предусловием в системах объектно-ориентированного и алгоритмического программирования.

Задание. Разработать проект, в котором:

- на метку выводится текущее время;
- на метку выводится прошедшее (или оставшееся) количество дней с (до) какого-либо события.

Проект «Даты и время» на языках объектно-ориентированного программирования Visual Basic 2005 или Gambas

1. В операционной системе Windows запустить систему объектно-ориентированного программирования Visual Basic 2005 командой [*Программы-Visual Basic 2005 Express Edition*].

   Или:

   В оперативной системе Linux запустить систему объектно-ориентированного программирования Gambas командой [*Программы-Средства разработки-Gambas*].

   Создадим графический интерфейс проекта.

2. Разместить на форме (рис. 4.5.1 и 4.5.2):
   - метку `Label1` для вывода значений текущего времени;
   - метку `Label2` для вывода прошедшего (или оставшегося) количества дней с (до) какого-либо события;
   - объект `Timer1` для периодического обновления значения времени, который вызывает событие `Tick` через определенные пользователем интервалы времени.

Периодичность события `Tick` может быть задана в свойстве `Interval`, измеряемом в миллисекундах (может изменяться от 0 до 65535). Для того чтобы событие `Tick` происходило каждую секунду, необходимо свойству `Interval` присвоить значение 1000.

3. Выделить объект `Timer1` и с помощью диалогового окна *Свойства* присвоить свойству `Interval` значение `1000`, а свойству `Enabled` значение **`True`**.

Создадим обработчик события на языке Visual Basic. Дату необходимо задать в формате День/Месяц/Год.

4.
```
Dim Dat1, Dat2 As Date
Sub Timer1_Tick( )
Label1.Text = TimeOfDay
Dat1 = #3/1/1950#
Dat2 = Today
Label2.Text=DateDiff(DateInterval.Day,Dat1,Dat2)
End Sub
```

Создадим обработчик события на языке Gambas. Дату необходимо задать в формате Месяц/День/Год.

5.
```
Public Sub Timer1_Timer()
Dim Dat1, Dat2 As Date
Label1.Text = Time
Dat1 = "3/1/1950"
Dat2 = Now
Label2.Text = DateDiff(Dat1, Dat2, gb.Day)
End
```

6. Запустить проект. На одну метку с интервалом в одну секунду будет выводиться системное время компьютера, а на другую метку — количество дней, прошедших со дня рождения до текущей даты.

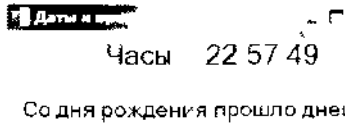


Рис. 4.5.1. Проект «Даты и время» на языке Visual Basic

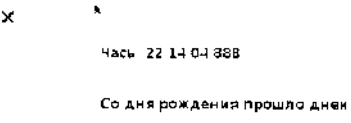


Рис. 4.5.2. Проект «Даты и время» на языке Gambas

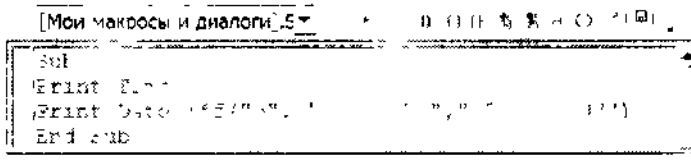
Обратите внимание, что в режиме выполнения проекта объект `Timer1` не виден.

Проект «Даты и время» на языке Visual Basic хранится в папке ..\IIKT9\VB2005\Даты и время
Проект «Даты и время» на языке Gambas хранится в папке ..\IIKT9\Gambas\Time

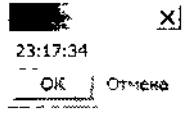
Windows-CD

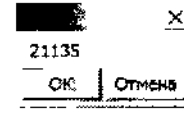
Программа «Даты и время» на языке алгоритмического программирования OpenOffice.org Basic

1. В операционной системе Windows или Linux запустить один из компонентов интегрированного офисного приложения OpenOffice.org (например, OpenOffice.org Writer) командой [*Программы-OpenOffice.org Writer*].
2. В меню приложения ввести команду [*Сервис-Макросы-Управление макросами-OpenOffice.org Бэйсик*]. Появится диалоговое окно *Макрос OpenOffice.org Basic*. Нажать кнопку *Создать*.
3. В появившемся диалоговом окне интерпретатора Basic ввести программный год на алгоритмическом языке Basic, между служебными словами **Sub** и **End Sub**.
   Функция `Time` возвращает значение текущего времени в формате Часы:Минуты:Секунды.
   Функция `DateDiff` возвращает количество дней между двумя датами, заданными в формате День/Месяц/Год.



4. Запустить на выполнение программу на языке Basic нажатием кнопки ▸ *Выполнить Basic*.
5. В диалоговых окнах последовательно появятся текущее время и количество дней, прошедших между двумя датами.





Программа хранится в файле ..\IKT9\Basic\Day.bas Windows-CD

Практическая работа 4.6

Проект «Сравнение кодов символов»

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться применять оператор ветвления в системах объектно-ориентированного и алгоритмического программирования.

Задание. Создать проект, который позволит определять больший из числовых кодов двух символов.

Проект «Сравнение кодов символов» на языках объектно-ориентированного программирования Visual Basic 2005 или Gambas

1. В операционной системе Windows запустить систему объектно-ориентированного программирования Visual Basic 2005 командой [*Программы-Visual Basic 2005 Express Edition*].
   Или:
   в операционной системе Linux запустить систему объектно-ориентированного программирования Gambas командой [*Программы Средства разработки-Gambas*].

   Создадим графический интерфейс проекта.
2. Разместить на форме (рис. 4.6.1 и 4.6.2):
   - текстовое поле `TextBox1` для ввода первого символа;
   - текстовое поле `TextBox2` для ввода второго символа;
   - метку `Label1` для вывода числового кода первого символа;
   - метку `Label2` для вывода числового кода второго символа;
   - метку `Label3` для вывода результата (большего числового кода);
   - кнопку `Button1` для запуска обработчика события.

Создадим для кнопки обработчик события, реализующий определение числового кода символов и выбор большего из них:

3.
```
Dim A, B As String
Private Sub Button1_Click( )
A = TextBox1.Text
B = TextBox2.Text
Label1.Text = Asc(A)
Label2.Text = Asc(B)
If Asc(A) > Asc(B) Then
    Label3.Text = Asc(A)
Else
    Label3 Text = Asc(B)
End If
End Sub
```

4. Запустить проект, ввести символ в первое поле, затем во второе.
Щелкнуть по кнопке *Сравнить*. На метках появятся числовые коды символов и больший числовой код.

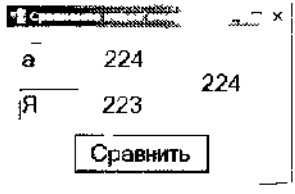


Рис. 4.6.1. Проект «Сравнение кодов символов» на языке Visual Basic

Рис. 4.6.2. Проект «Сравнение кодов символов» на языке Gambas

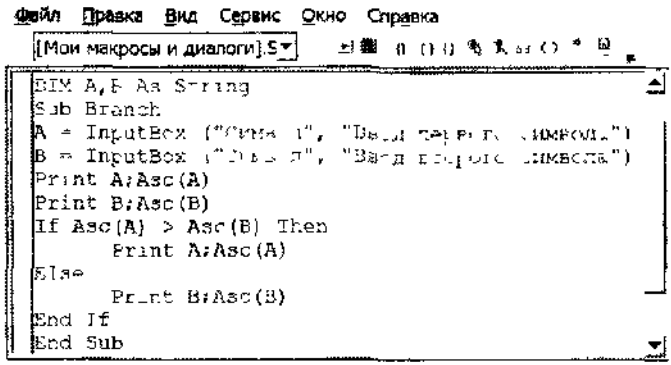
Обратите внимание на то, что в языке Gambas на метки выводятся десятичные числовые коды символов в кодировке *ASCII*.

Проект «Сравнение кодов символов» на языке Visual Basic хранится в папке .\IIKT9\VB2005\Сравнение кодов
Проект «Сравнение кодов символов» на языке Gambas хранится в папке .\IIKT9\Gambas\Branch

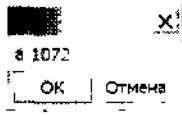
Windows-CD

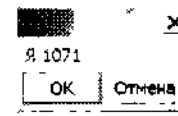
Программа «Сравнение кодов символов» на языке алгоритмического программирования OpenOffice.org Basic

1. В операционной системе Windows или Linux запустить один из компонентов интегрированного офисного приложения OpenOffice.org (например, OpenOffice.org Writer) командой [*Программы-OpenOffice.org Writer*].
2. В меню приложения ввести команду [*Сервис-Макросы-Управление макросами-OpenOffice.org Бэйсик*]. Появится диалоговое окно *Макрос OpenOffice.org Basic*. Нажать кнопку *Создать*.
3. В появившемся диалоговом окне интерпретатора Basic ввести программный код на алгоритмическом языке Basic, между служебными словами **Sub** и **End Sub**.



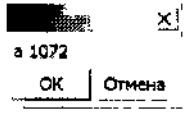
4. Запустить на выполнение программу на языке Basic нажатием кнопки *Выполнить Basic*. В последовательно появляющихся диалоговых окнах ввести первый символ (например, «а») и второй символ (например, «Я»).
5. Появятся окна с промежуточными результатами выполнения программы.





Обратите внимание на то, что функция `Asc()` возвращает десятичный числовой код символа в кодировке *Unicode*.

6. В результате появится окно с символом, соответствующим большему числовому коду.



Программа хранится в файле ..\IIKT9\Basic\Branch.bas Windows-CD

Практическая работа 4.7

Проект «Отметка»

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться применять оператор выбора в системах объектно-ориентированного и алгоритмического программирования.

Задание. Создать проект, который позволит выставлять отметку в зависимости от количества ошибок.

Проект «Отметка» на языке объектно-ориентированного программирования Visual Basic 2005 или Gambas

1. В операционной системе Windows запустить систему объектно-ориентированного программирования Visual Basic 2005 командой [*Программы-Visual Basic 2005 Express Edition*].
Или:
в операционной системе Linux запустить систему объектно-ориентированного программирования Gambas командой [*Программы-Средства разработки-Gambas*].

Создадим графический интерфейс проекта.

2. Разместить на форме (рис. 4.7.1 и 4.7.2).
Поместить на форму:
- текстовое поле `TextBox1` для ввода количества ошибок;
- надпись `Label1` для вывода отметки;
- кнопку `Button1` для запуска обработчика событий;
- две надписи для вывода поясняющих текстов.

В обработчике событий объявим переменную N, значением которой будет являться количество ошибок, как переменную типа **Byte** (количество ошибок не может быть отрицательным и вряд ли может быть больше 255). Присвоим переменной N значение свойства Text текстового поля TextBox1.

В операторе **Select Case** в зависимости от значения переменной N будем присваивать значению свойства Text надписи Label1 определенные отметки:

3.
```
Dim N As Byte
Private Sub Button1_Click( )
N = TextBox1.Text
Select Case N
Case 0
Label1.Text = "Отлично"
Case 1
Label1.Text = "Хорошо"
Case 2
Label1.Text = "Удовлетворительно"
Case Else
Label1.Text = "Плохо"
End Select
End Sub
```

4. После запуска проекта на выполнение необходимо ввести в текстовое поле количество ошибок и щелкнуть по кнопке *Отметка*, на надпись будет выведена соответствующая отметка.

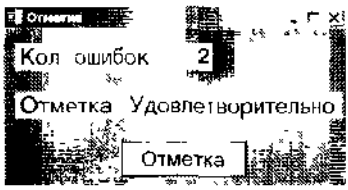


Рис. 4.7.1. Проект «Отметка» на языке Visual Basic

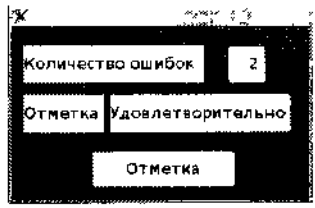


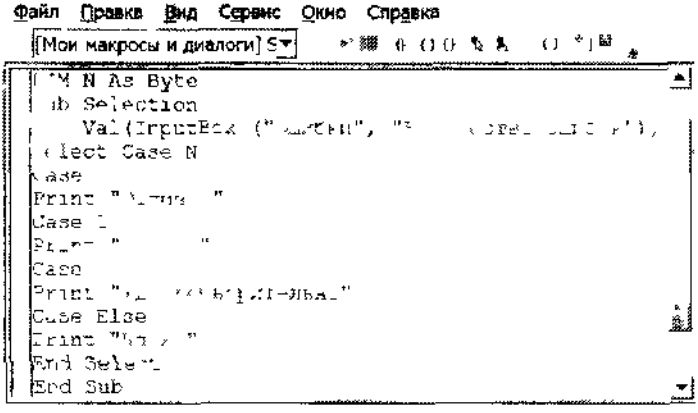
Рис. 4.7.2. Проект «Отметка» на языке Gambas

Проект «Отметка» на языке Visual Basic хранится в папке ..\IIKT9\VB2005\Отметка
Проект «Отметка» на языке Gambas хранится в папке ..\IIKT9\Gambas\Selection

Windows-CD

Программа «Отметка» на языке алгоритмического программирования OpenOffice.org Basic

1. В операционной системе Windows или Linux запустить один из компонентов интегрированного офисного приложения OpenOffice.org (например, OpenOffice.org Writer) командой [*Программы OpenOffice.org Writer*].
2. В меню приложения ввести команду [*Сервис-Макросы-Управление макросами-OpenOffice.org Бэйсик*]. Появится диалоговое окно *Макрос OpenOffice.org Basic*. Нажать кнопку *Создать*.
3. В появившемся диалоговом окне интерпретатора Basic ввести программный год на алгоритмическом языке Basic, между служебными словами **Sub** и **End Sub**.



4. Запустить на выполнение программу на языке Basic нажатием кнопки ▶ *Выполнить Basic*.
   В диалоговом окне ввести количество ошибок (например, 2)
5. В результате появится окно с отметкой.

Программа хранится в файле ..\IKT9\Basic\Selection.bas **Windows-CD**

Практическая работа 4.8
Проект «Коды символов»

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться применять оператор цикла со счетчиком в системах объектно-ориентированного и алгоритмического программирования.

Задание. Создать проект, который должен выводить в поле списка числовые коды символов и соответствующие им символы.

Проект «Коды символов» на языках объектно-ориентированного программирования Visual Basic 2005 или Gambas

1. В операционной системе Windows запустить систему объектно-ориентированного программирования Visual Basic 2005 командой [*Программы Visual Basic 2005 Express Edition*].
   Или:
   в операционной системе Linux запустить систему объектно-ориентированного программирования Gambas командой [*Программы Средства разработки-Gambas*].

Создадим графический интерфейс проекта.

2. Разместить на форме (рис. 4.8.1 и 4.8.2):
   - поле списка `ListBox1` для вывода числовых кодов символов;
   - поле списка `ListBox2` для вывода соответствующих им символов;
   - кнопку `Button1` для запуска обработчика событий.

Создадим обработчик события, в котором в качестве счетчика цикла используем целочисленную переменную `i`. В кодировке *Windows* первые 33 кода (десятичные коды с 0 по 32) соответствуют не знакам, а клавишам клавиатуры (*клавиши управления курсором*, *Пробел*, *Ввод* и др.). Поэтому воспользуемся циклом со счетчиком с шагом –1, для того чтобы выводить на форму символы, начиная с наибольшего числового кода 255.

Для преобразования числового кода в символ используем функцию `Chr()`, аргументом которой является число (от 33 до 255), а значением — символ. В теле цикла числовые коды символов и соответствующие им символы будут выводиться в поля списков с помощью метода `Iters.Add()`.

3.
```
Dim N As Integer
Sub Button1_Click(…)
For N = 255 To 33 Step -1
ListBox1.Items.Add(N)
ListBox2.Items.Add(Chr(N))
Next N
End Sub
```

Для вывода символов в кодировке *ASCII* воспользуемся циклом со счетчиком с шагом –1, для того чтобы печатать на форме символы, начиная с наибольшего числового кода 126.

Для преобразования числового кода в символ на языке Gambas используем функцию `Chr()`, аргументом которой является число (от 33 до 126), а значением — символ. В теле цикла числовые коды символов и соответствующие им символы будут выводиться в поля списков с помощью метода `Add()`.

4.
```
Public Sub Button1_Click()
Dim N As Integer
For N = 126 To 33 Step -1
ListBox1.Add(N)
ListBox2.Add(Chr(N))
Next
End
```

5. После запуска проекта на выполнение необходимо щелкнуть по кнопке *Пуск*.

В полях списка будут напечатаны последовательности числовых кодов символов и соответствующих им символов. С помощью полос прокрутки можно ознакомиться со всеми кодами и их символами.

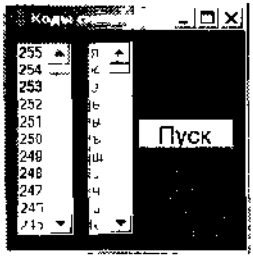


Рис. 4.8.1. Результат выполнения проекта «Коды символов» на языке Visual Basic

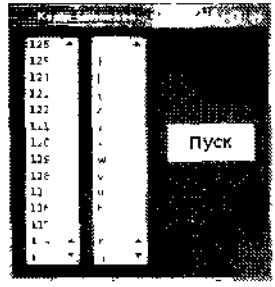
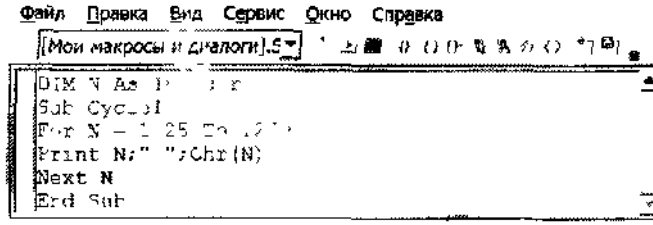


Рис. 4.8.2. Результат выполнения проекта «Коды символов» на языке Gambas

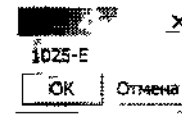
Проект «Коды символов» на языке Visual Basic хранится в папке ..\IIKT9\VB2005\Коды символов
Проект «Коды символов» на языке Gambas хранится в папке ..\IIKT9\Gambas\Cycle1
Windows-CD

Программа «Коды символов» на языке алгоритмического программирования OpenOffice.org Basic

1. В операционной системе Windows или Linux запустить один из компонентов интегрированного офисного приложения OpenOffice.org (например, OpenOffice.org Writer) командой [*Программы-OpenOffice.org Writer*].
2. В меню приложения ввести команду [*Сервис Макросы-Управление макросами OpenOffice.org Бэйсик*]. Появится диалоговое окно *Макрос OpenOffice.org Basic*. Нажать кнопку *Создать*.
3. В появившемся диалоговом окне интерпретатора Basic ввести программный код на алгоритмическом языке Basic, между служебными словами **Sub** и **End Sub**.
   В кодировке *Unicode* знаки кириллицы начинаются с десятичного кода 1025, а заканчиваются десятичным кодом 1279 (в кириллицу входят не только буквы русского алфавита). Учтем это при присваивании начального и конечного значений счетчика цикла.



4. Запустить на выполнение программу на языке Basic нажатием кнопки *Выполнить Basic*.
5. В диалоговых окнах последовательно появятся десятичные коды символов и соответствующие им символы в кодировке *Unicode*.



Программа хранится в файле ..\IIKT9\Basic\Cycle1.bas
Windows-CD

Практическая работа 4.9

Проект «Слово-перевертыш»

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться применять оператор цикла с предусловием в системах объектно-ориентированного и алгоритмического программирования.

Задание. Создать проект преобразования введенного слова в слово-перевертыш, т. е. в слово с обратной последовательностью следования символов.

Проект «Слово-перевертыш» на языках объектно-ориентированного программирования Visual Basic 2005 или Gambas

1. В операционной системе Windows запустить систему объектно-ориентированного программирования Visual Basic 2005 командой [*Программы-Visual Basic 2005 Express Edition*].
   Или:
   в операционной системе Linux запустить систему объектно-ориентированного программирования Gambas командой [*Программы-Средства разработки-Gambas*].

   Разработаем графичсский интерфейс проекта.
2. Поместить на форму (см. рис. 4.9.1 и 4.9.2).:
   - текстовое поле `TextBox1` для ввода исходного слова;
   - надпись `Label1` для вывода слова-перевертыша;
   - кнопку `Button1` для создания обработчика событий.

В обработчике события цикл с предусловием будет выполняться, пока справедливо условие `N <= Len(TextBox1.Text)`, т. е. пока значение переменной `N` меньше или равно количеству символов в слове. (Количество символов во введенном слове является значением строковой функции `Len()`.)

В цикле символы последовательно вырезаются из введенного слова в прямой последовательности (слева направо) с использованием функции вырезания подстроки из строки `Mid(TextBox1.Text, N, 1)` и присваиваются строковой переменной `S`. Затем вырезанные символы (значения переменной `S`) в обратной последовательности (справа налево) присваиваются свойству `Label1.Text`, значением которого после завершения цикла будет слово-перевертыш.

3.
```
Dim S As String, N As Byte
Private Sub Button1_Click( )
Label1.Text = ""
N = 1
Do While N <= Ler(TextBox1.Text)
S = Mid(TextBox1.Text, N, 1)
Label1.Text = S & Label1.Text
N = N + 1
Loop
End Sub
```

4. После запуска проекта на выполнение необходимо ввести *в текстовое поле исходное слово и щелкнуть по* кнопке *Пуск*. На метку будет выведено слово-перевертыш.

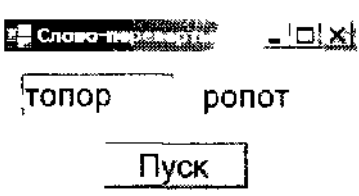


Рис. 4.9.1. Проект «Слово-перевертыш» на языке Visual Basic

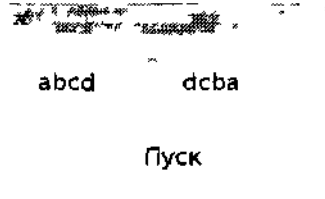


Рис. 4.9.2. Проект «Слово-перевертыш» на языке Gambas

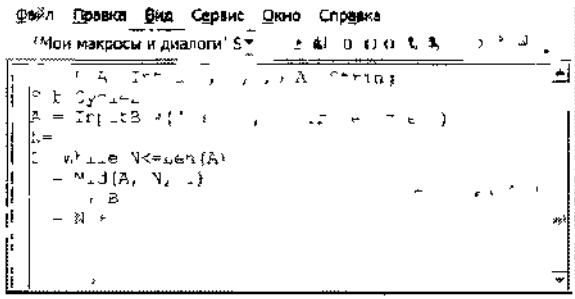
Проект «Слово-перевертыш» на языке Visual Basic хранится в папке ..\IIKT9\VB2005\Слово-перевертыш
Проект «Слово-перевертыш» на языке *Gambas* хранится в папке ..\IIKT9\Gambas\Cycle2
Windows-CD

Программа «Слово-перевертыш» на языке алгоритмического программирования OpenOffice.org Basic

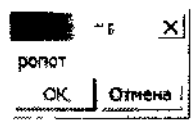
1. В операционной системе Windows или Linux запустить один из компонентов интегрированного офисного приложения OpenOffice.org (например, OpenOffice.org Writer) командой [*Программы-OpenOffice.org Writer*].
2. В меню приложения ввести команду [*Сервис-Макросы Управление макросами-OpenOffice.org Бэйсик*].

Появится диалоговое окно *Макрос OpenOffice.org Basic*. Нажать кнопку *Создать*.

3. В появившемся диалоговом окне интерпретатора Basic ввести программный год на алгоритмическом языке Basic, между служебными словами **Sub** и **End Sub**.



4. Запустить на выполнение программу на языке Basic нажатием кнопки ▶| *Выполнить Basic*.
5. В диалоговом окне появится слово-перевертыш.



Программа хранится в файле ..\IIKT9\Basic\Cycle2.bas **Windows-CD**

*Практическая работа 4.10

Проект «Графический редактор»

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows.

Цель работы. Научиться применять графические методы в объектно-ориентированном программировании.

Задание. Создать проект, который позволит рисовать линии, прямоугольники и окружности, заданным цветом.

Проект «Графический редактор» на языке объектно-ориентированного программирования Visual Basic 2005

1. В операционной системе Windows запустить систему объектно-ориентированного программирования Visual Basic 2005 командой [*Программы-Visual Basic 2005 Express Edition*].

Создадим графический интерфейс проекта.

2. Поместить на форму (рис. 4.10.1):
   - графическое поле `PictureBox1`, которое будет использоваться в качестве области рисования;
   - четыре текстовых поля `TextBox1`, `TextBox2`, `TextBox3` и `TextBox4` для ввода значений переменных `X1`, `Y1` и `X2`, `Y2`, содержащих координаты графических фигур;
   - три текстовых поля `TextBox5`, `TextBox6` и `TextBox7` для ввода значений переменных `Red`, `Green`, `Blue`, содержащих числовые коды цветов;
   - семь меток для вывода поясняющих текстов;
   - элемент управления `MenuStrip1` для создания меню проекта;
   - элемент управления `ColorDialog1` для выбора цвета в диалоговом окне.

Для большей понятности программного кода будем вводить в него комментарии, которые начинаются с символа апостроф «'».

Объявим объекты `Graphics` (*Область рисования*) и `Pen` (*Перо*), а также переменные, которые будут использоваться в проекте:

3.
```
Dim Graph1 As Graphics
Dim Pen1 As New Pen(Color.Red, 3)
Dim X1, X2, Y1, Y2, Red, Green, Blue As Integer
```

Создадим меню графического редактора. Пусть меню состоит из четырех заголовков верхнего уровня: *Нарисовать*, *Ввод координат*, *Ввод цвета* и *Стереть*. В пункт *Нарисовать* входят команды *Линия*, *Прямоугольник* и *Окружность*.

Нарисовать	*Ввод координат*	*Ввод цвета*	*Стереть*
Линия			
Прямоугольник			
Окружность			

Для создания меню используется управляющий элемент *MenuStrip1* (*Редактор меню*).

4. В редакторе меню, появившемся в левом верхнем углу формы, создать заголовки первого уровня (в поле *Type Here* внести пункты меню). Для создания подпункта меню перейти на следующую строку в редакторе меню.

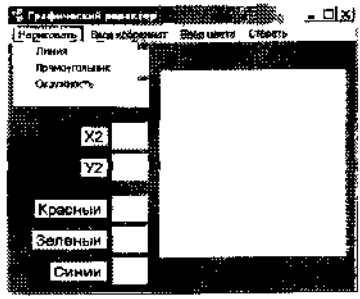


Рис. 4.10.1. Создание меню графического редактора

Создадим программный код обработчика события, который реализует ввод координат графических фигур. Для этого необходимо щелкнуть по пункту меню *Ввод координат* и в появившуюся заготовку ввести программный код:

5.
```
'Ввод координат
Private Sub ВводКоорди-
натToolStripMenuItem_Click( )
Graph1 = Me.PictureBox1.CreateGraphics()
X1 = TextBox1.Text
Y1 = TextBox2.Text
X2 = TextBox3.Text
Y2 = TextBox4.Text
End Sub
```

Создадим программный код обработчика события для ввода интенсивностей базовых цветов в системе RGB. Для этого надо щелкнуть по пункту меню *Ввод цвета* и в появившуюся заготовку ввести программный код. Для преобразования строкового значения в число используем функцию `Val()`:

6.
```
'Цвет
Private Sub ВводЦветаToolStripMenuItem_Click( )
Red = Val(TextBox5.Text)
Creen = Val(TextBox6.Text)
Blue = Val(TextBox7.Text)
End Sub
```

Создадим программный код обработчика события, который реализует рисование линии. Для этого необходимо щелкнуть по пункту меню *Линия* и в появившуюся заготовку ввести программный код. Цвет линии зададим с помощью цветовой константы (в данном случае `Red`):

7.
```
'Линия
Private Sub ЛинияToolStripMenuItem_Click(…)
Pen1.Color = Color.Red
Graph1.DrawLine(Pen1, X1, Y1, X2, Y2)
End Sub
```

Создадим программный код обработчика события, реализующего рисование прямоугольника. Для этого необходимо щелкнуть по пункту меню *Прямоугольник* и в появившуюся заготовку ввести программный код. Цвет контура зададим с помощью функции `FromArgb(Red,Green,Blue)`, аргументами которой являются интенсивности базовых цветов (красного, зеленого и синего).

8.
```
'Прямоугольник, где аргумент Width=X2-X1, а
'аргумент Height=Y2-Y1
Private Sub ПрямоугольникToolStripMenuItem_Click(…)
Pen1.Color = Color.FromArgb(Red, Green, Blue)
Graph1.DrawRectangle(Pen1, X1, Y1, X2-X1,
Y2-Y1)
End Sub
```

Создадим программный код обработчика события, реализующего рисование окружности. Для этого необходимо щелкнуть по пункту меню *Окружность* и в появившуюся заготовку ввести программный код.

Цвет контура зададим с помощью диалога `ColorDialog1`, который предоставляет для выбора цвета диалоговое окно *Цвет* (рис. 4.10.2). Оно будет выведено с помощью метода `ShowDialog()`:

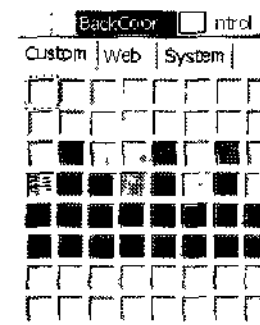


Рис. 4.10.2. Диалоговое окно *Цвет*

9.
```
'Окружность
Private Sub ОкружностьToolStripMenuItem_Click(…)
ColorDialog1.ShowDialog()
Pen1.Color = ColorDialog1.Color
Graph1.DrawEllipse(Pen1, X1, Y1, X2, Y2)
End Sub
```

Предусмотрим стирание неудачно нарисованного. Создадим программный код обработчика события, реализующего стирание. Для этого необходимо щелкнуть по пункту меню *Стереть* и в появившуюся заготовку ввести программный код:

10.
```
'Стирание
Private Sub СтеретьToolStripMenuItem_Click(…)
Graph1.Clear(Color.White)
End Sub
```

11. Запустить проект. Ввести в текстовые поля значения координат графических фигур и щелкнуть по пункту меню *Ввод координат*.

Ввести в текстовые поля интенсивности базовых цветов и щелкнуть по кнопке *Ввод цвета*.

Нарисовать графические фигуры, последовательно щелкнув по пунктам меню *Линия*, *Прямоугольник*, *Окружность*.

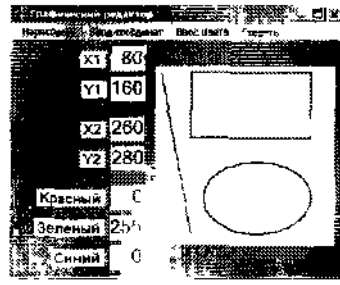


Рис. 4.10.3. Проект «Графический редактор»

При необходимости очистить поле рисования, щелкнув по пункту меню *Стереть*.

Результат выполнения проекта — на рис. 4.10.3.

Проект «Графический редактор» на языке Visual Basic хранится в папке ..\IIKT9\VB2005\Графический редактор **Windows-CD**

*Практическая работа 4.11

Проект «Системы координат»

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows.

Цель работы. Научиться создавать различные системы координат в системах объектно-ориентированного программирования.

Задание. Создать проект, который обеспечит рисование *осей и печать шкалы в компьютерной системе координат* (см. рис. 4.10) и математической системе координат (см. *рис.* 4.11).

Проект «Системы координат» на языке объектно-ориентированного программирования Visual Basic 2005

1. В операционной системе Windows запустить систему объектно-ориентированного программирования Visual Basic 2005 командой [*Программы-Visual Basic 2005 Express Edition*].

Создадим графический интерфейс проекта и поместим на форму:

2. • графическое поле PictureBox1, которое будет использоваться в качестве области рисования;
   • кнопки Button1 и Button2 для запуска обработчиков событий.

Установим размеры графического поля PictureBox1.

3. Присвоить свойству Size значение 300;200.

Создадим обработчик события, реализующий рисование осей и печать шкал в компьютерной системе координат:

4.
```
'Объявление переменных
Dim Graph1 As Graphics
Dim Pen1 As New Pen(Color.Red, 3)
Dim drawBrush As New SolidBrush(Color.Black)
Dim drawFont As New Font("Arial", 10)
Dim X, Y As Integer
'Событийная процедура
Private Sub Button1_Click(…)
Graph1 = Me.PictureBox1.CreateGraphics()
Graph1.Clear(Color.White)
'Рисование шкал компьютерной системы координат
Graph1.DrawLine(Pen1, 0, 0, 300, 0) 'Ось X
Graph1.DrawLine(Pen1, 0, 0, 0, 200) 'Ось Y
For X = 0 To 300 Step 50 'Засечки на оси X
Graph1.DrawLine(Pen1, X, 0, X, 10)
Next X
For Y = 0 To 200 Step 50 'Засечки на оси Y
Graph1.DrawLine(Pen1, 0, Y, 10, Y)
Next Y
'Печать шкалы оси X
For X = 0 To 300 Step 50
Graph1.DrawString(X, drawFont, drawBrush, X, 10)
Next X
```

```
'Печать шкалы оси Y
For Y = 0 To 200 Step 50
Graph1.DrawString(Y, drawFont, drawBrush, 10, Y)
Next Y
End Sub
```

Создадим обработчик события, реализующий рисование осей и печать шкал в математической системе координат:

5.
```
Private Sub Button1_Click(…)
Graph1 = Me.PictureBox1.CreateGraphics()
Graph1.Clear(Color.White)
'Печать шкал математической системы координат
'в компьютерной системе координат
For X = -150 To 150 Step 50
Graph1.DrawString(X, drawFont, drawBrush, X +
150, 80)
Next X
For Y = 0 To 200 Step 50
Graph1.DrawString(Y - 100, drawFont, drawBrush,
150, 200 - Y)
Next 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 с засечками в математической
'системе координат
Graph1.DrawLine(Pen1, -150, 0, 150, 0) 'Ось X
Graph1.DrawLine(Pen1, 0, -100, 0, 100) 'Ось Y
For X = -150 To 150 Step 50 'Засечки на оси X
Graph1.DrawLine(Pen1, X, -5, X, 5)
Next X
For Y = -100 To 100 Step 50 'Засечки на оси Y
Graph1.DrawLine(Pen1, -5, Y, 5, Y)
Next Y
End Sub
```

Проект «Система координат» на языке Visual Basic хранится в папке ..\IIKT9\VB2005\Системы координат\ Windows-CD

*Практическая работа 4.12

Проект «Анимация»

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows.

Цель работы. Научиться создавать анимацию в системах объектно-ориентированного программирования.

Задание. Разработать проект, в котором реализуется «полет бабочки». Для создания иллюзии взмаха крыльями два изображения бабочки («с развернутыми крыльями» и «со свернутыми крыльями») с определенной частотой выводятся в поле рисования. Для создания иллюзии движения при каждом взмахе координаты изображения изменяются на определенную величину.

Проект «Системы координат» на языке объектно-ориентированного программирования Visual Basic 2005

1. Запустить систему объектно-ориентированного программирования Visual Basic 2005 командой [*Программы-Visual Basic 2005 Express Edition*].

Создадим графический интерфейс проекта.

2. Поместить на форму (рис. 4.12.1):
   - графическое поле `PictureBox1`, которое будет использоваться в качестве области рисования;
   - объект `Timer1`, для периодического вывода изображения на графическое поле, который вызывает событие `Tick` через определенные пользователем интервалы времени.

Периодичность события `Tick` может быть задана в свойстве `Interval`, измеряемом в миллисекундах (может изменяться от 0 до 65535).

3. Выделить объект `Timer1` и с помощью диалогового окна *Свойства* присвоить свойству `Interval` значение 100, а свойству `Enabled` — значение **True**.

Создадим заготовку обработчика события щелчком по объекту `Timer1`:

4.
```
Private Sub Timer1_Tick(…)

End Sub
```

Выберем в качестве области рисования графическое поле PictureBox1.

Используем в качестве изображения бабочки «с развернутыми крыльями» графический файл bfly1.bmp, а изображения «со свернутыми крыльями» — графический файл bfly2.bmp.

Объявим графические объекты Image1 и Image2 и создадим их из файлов с помощью функции Image.FromFile(). Аргументом функции является путь к графическому файлу, а также его имя.

Для того чтобы при каждом событии Tick выводить попеременно то одно, то другое изображение бабочки, используем логическую переменную flg1. При каждом событии с помощью оператора **If-Then-Else** в однострочной форме будем менять ее значение с **True** на **False** или, наоборот, с **False** на **True**. Затем с использованием оператора **If-Then-Else-End If** в многострочной форме будем выводить в зависимости от значения логической переменной flg1 то или иное изображение бабочки.

Вывод изображения на область рисования будем осуществлять с помощью метода DrawImage(Image1, X, Y), аргументами которого являются изображение и координаты его левого верхнего угла на области рисования.

Для реализации перемещения из нижнего левого угла графического поля в правый верхний угол сместим ось координат Y вниз на 200 точек и при каждом событии будем увеличивать координату X и уменьшать координату Y.

В результате обработчик события примет следующий вид:

5.
```
Dim Graph1 As Graphics
Dim Image1, Image2 As Image
Dim X, Y As Single
Dim flg1 As Boolean
Private Sub Timer1_Tick(…)
Graph1 = Me.PictureBox1.CreateGraphics()
Image1 =
Image.FromFile("\IIKT9\VB2005\Анимация\bfly1.bmp")
Image2 =
Image.FromFile("\IIKT9\VB2005\Анимация\bfly2.bmp")
If flg1 Then flg1 = False Else flg1 = True
Graph1.TranslateTransform(0, 200)
If flg1 Then
    Graph1.DrawImage(Image1, X, Y)
```

```
Else
    Graph1.DrawImage(Image2, X, Y)
End If
X = X + 5
Y = Y - 5
End Sub
```

6. Запустить проект. В графическом поле начнется «порхание» бабочки из нижнего левого угла графического поля в правый верхний угол. Для изменения частоты «порхания» необходимо изменить у объекта `Timer1` значение свойства `Interval`. Для изменения скорости полета необходимо изменить шаг изменения координат.

Рис. 4.12.1. Проект «Анимация»

Проект «Анимация» на языке Visual Basic хранится в папке ..\IIKT9\VB2005\Анимация **Windows-CD**

Практические работы к главе 5 «Моделирование и формализация»

	Установить: • систему объектно-ориентированного программирования Visual Basic 2005; • электронные таблицы OpenOffice.org Calc; • электронные таблицы Microsoft Excel	VisualStudio-CD Windows-CD Дистрибутив Microsoft Office
	Установить: • электронные таблицы OpenOffice.org Calc	Linux-DVD

*Практическая работа 5.1

Проект «Бросание мячика в площадку»

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться создавать компьютерные модели движения на языке объектно-ориентированного программирования Visual Basic или в электронных таблицах.

Задание. Разработать проект, в котором визуализируется траектория движения тела, брошенного под углом к горизонту, и выясняется, попадет ли оно в площадку определенной длины, находящуюся на заданном расстоянии.

Проект «Бросание мячика в площадку» на языке объектно-ориентированного программирования Visual Basic 2005

1. В операционной системе Windows запустить систему объектно-ориентированного программирования Visual Basic 2005 командой [*Программы-Visual Basic 2005 Express Edition*].

Создадим сначала графический интерфейс проекта.

2. Разместить на форме (рис. 5.1.1):
   - четыре текстовых поля `TextBox` для ввода значений начальной скорости и угла бросания мячика, расстояния до площадки и ее длины;
   - два метки `Label` для вывода координаты `X` мячика в момент падения на поверхность и текстового сообщения о результатах броска;
   - десять надписей `Label` для обозначения назначения текстовых полей (имен переменных и единиц измерения).

3. Создать программный код обработчика события, который определяет попадание мячика в площадку:
   - объявить вещественные константы одинарной точности `G` (ускорение свободного падения g) и `Pi` (число π);
   - объявить вещественные переменные одинарной точности `V0` (начальная скорость v), `A` (угол бросания α), `S` (расстояние до площадки s), `L` (длина площадки l), `X` и `Y` (координаты мячика), `T` (время);
   - присвоить переменным `V0`, `A`, `S`, `L` значения, введенные в текстовые поля, с использованием функции преобразования строки в вещественное число `Val()`;

- вычислить координату мячика X в момент падения на поверхность;
- вывести координату X мячика на метку Label1;
- вывести текстовое сообщение о результатах броска в поле метки Label2 с использованием оператора Select Case, в котором в качестве условия проверяется значение переменной X.

В языке программирования Visual Basic аргументы тригонометрических функций Sin(), Cos() и Tan() задаются в радианах, а угол бросания мячика мы будем вводить в градусах. Поэтому необходимо преобразовать значение угла из градусов в радианы с использованием константы Pi.

4. Поместить на форму кнопку Button1 и создать для нее событийную процедуру Button1_Click():

```
Const G As Single = 9.81
Const Pi As Single = 3.14
Dim V0, A, S, L, X As Single
Private Sub Button1_Click(…)
'Ввод начальных значений
V0 = Val(TextBox1.Text)
A = Val(TextBox2.Text)
S = Val(TextBox3.Text)
L = Val(TextBox4.Text)
'Вычисление координаты мячика в момент падения
'на площадку
X = V0^2*Math.Sin(2*A*Pi/180)/G
Label1.Text = X
'Попадание в площадку
Select Case X
Case Is < S
Label2.Text = "Недолет"
Case Is > S + L
Label2.Text = "Перелет"
Case Else
Label2.Text = "Попадание"
End Select
End Sub
```

Для визуализации формальной модели построим траекторию движения тела (график зависимости высоты мячика над поверхностью земли от дальности полета).

5. Поместить дополнительно на форму графическое поле `PictureBox1` (см. рис. 5.1.1). С помощью диалогового окна *Свойства* установить с использованием свойства `Size` размер поля, например `400;220`.

В обработчике события осуществим преобразование компьютерной системы координат графического поля в математическую систему координат, удобную для построения траектории движения. Нарисуем оси координат и нанесем на них шкалы.

4.11. Практическая работа «Проект «Системы координат»

В математической системе координаты находятся в диапазонах $0 \le X \le 400$ и $-20 \le Y \le 200$. Траектория движения мячика, скорее всего, будет в диапазоне координат $0 \le X \le 40$ м и $0 \le Y \le 20$ м. Следовательно, необходимо увеличить масштаб графика в 10 раз:

- координаты точек графика необходимо умножить на 10;
- значения шкал осей разделить на 10.

Построение траектории осуществим в цикле со счетчиком (координата `X`) с использованием метода рисования точки `DrawEllipse(Pen1, X * 10, Y * 10, 1, 1)`, в котором координатами точки являются координаты мячика:

6.
```
'Объявление переменных
Dim X, Y, T As Single
Dim Graph1 As Graphics
Dim Pen1 As New Pen(Color.Black, 4)
Dim drawBrush As New SolidBrush(Color.Black)
Dim drawFont As New Font("Arial", 10)
'Обработчик события
Private Sub Button2_Click( )
Graph1 = Me.PictureBox1.CreateGraphics()
'Печать шкал математической системы координат
в компьютерной системе координат
For X = 0 To 400 Step 50
Graph1.DrawString(X / 10, drawFont, drawBrush,
X - 15, 200)
Next X
```

```
For Y = 0 To 200 Step 50
Graph1.DrawString(Y / 10, drawFont, drawBrush,
0, 200 - Y)
Next Y
'Преобразование компьютерной системы координат
'в математическую систему координат
Graph1.ScaleTransform(1, -1) 'Поворот оси Y
Graph1.TranslateTransform(0, -200)
'Сдвиг по оси Y
'Рисование осей математической системы
'координат
Graph1.DrawLine(Pen1, 0, 0, 400, 0) 'Ось X
For X = 0 To 400 Step 50 'Засечки на оси X
Graph1.DrawLine(Pen1, X, 0, X, 10)
Next X
Graph1.DrawLine(Pen1, 0, -20, 0, 200) 'Ось Y
For Y = 0 To 200 Step 50 'Засечки на оси Y
Graph1.DrawLine(Pen1, 0, Y, 10, Y)
Next Y
'Площадка
Graph1.DrawLine(Pen1, S * 10, 4, (S + L) *
10, 4) 'Построение траектории движения мячика
For T = 0 To 10 Step 0.1
Y = V0 * Math.Sin(A * Pi / 180) * T - G * T * T / 2
X = V0 * Math.Cos(A * Pi / 180) * T
Graph1.DrawEllipse(Pen1, X * 10, Y * 10, 1, 1)
Next T
End Sub
```

Компьютерный эксперимент. Введем произвольные значения начальной скорости и угла бросания мячика, скорее всего, его попадания в площадку не будет. Меняя один из параметров, например угол, произведем пристрелку, используя известный артиллерийский прием «взятие в вилку», в котором применяется эффективный метод «деление пополам». Сначала найдем угол, при котором мячик перелетит площадку, затем угол, при котором мячик не долетит до нее. Вычислим среднее значение углов, составляющих «вилку», и проверим, попадет ли мячик в площадку. Если он попадет в площадку, то задача выполнена, если не попадет, то рассматривается новая «вилка» и т. д.

7. Запустить проект и ввести значения начальной скорости, угла, расстояния до площадки и ее длины. Щелкнуть по кнопкам *Вычислить* и *Показать*. На надписи будут выведены результаты, а в графическом поле появится траектория движения тела.

8. Подобрать значения начальной скорости и угла бросания мячика, обеспечивающие его попадание в площадку. Например, при скорости бросания мячика $v_0 = 18$ *м/с* и угле бросания $\alpha = 34$ *град* мячик попадет в площадку длиной $L = 2$ *м*, находящуюся на расстоянии $S = 30$ *м*, на расстоянии $x = 30{,}61514$ *м*.

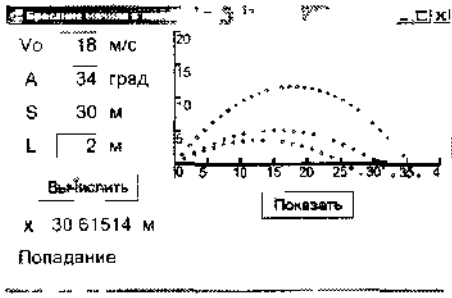


Рис. 5.1.1. Проект «Бросание мячика в площадку» на языке Visual Basic

Проект «Бросание мячика в площадку» на языке Visual Basic хранится в папке ..\IKT9\VB2005*Бросание мячика в площадку*\ **Windows-CD**

Анализ результатов. Полученная точность расстояния попадания мячика в площадку $x = 30{,}61514$ *м* не имеет физического смысла и определяется типом переменной. Так как X является переменной одинарной точности, то ее значение вычисляется с точностью семи значащих цифр. Исходные данные заданы с точностью двух значащих цифр, поэтому целесообразно результат округлить до трех значащих цифр $x = 30{,}6$ *м*.

Проект «Бросание мячика в площадку» в электронных таблицах Microsoft Excel и OpenOffice.org Calc

1. В операционной системе Windows запустить электронные таблицы Microsoft Excel командой [*Программы Microsoft Office Microsoft Office Excel*] или электрон-

ные таблицы OpenOffice org Calc командой [*Программы OpenOffice-OpenOffice Calc*].
Или:
в операционной системе Linux запустить электронные таблицы OpenOffice.org Calc командой [*Программы OpenOffice OpenOffice Calc*].

Построение траектории движения мячика. Для ввода начальной скорости бросания мячика v_0 будем использовать ячейку B1, а для ввода угла бросания — ячейку B2 (рис. 5.1.2).

Введем в ячейки A5:A18 значения времени t с интервалом 0,2 с и вычислим по формулам (5.1) значения координат тела x и y для заданных значений времени.

2. Ввести:
   - в ячейку B5 формулу
     =$B$1*COS(РАДИАНЫ($B$2))*A5;
   - в ячейку C5 формулу
     =$B$1*SIN(РАДИАНЫ($B$2))*A5–4,9*A5*A5

В электронных таблицах аргументы функций COS() и SIN() задаются в радианах, поэтому необходимо преобразовать значения углов из градусов в радианы с помощью функции РАДИАНЫ().

3. Скопировать введенные формулы в ячейки B6:B18 и C6:C18 соответственно.
Получим в столбце B значения координаты мячика по оси X, а в столбце C — координаты по оси Y, вычисленные для определенных моментов времени (см. рис. 5.1.2).

	A	B	C
1	v_0 =	18 м/с	
2	α =	34 град	
3			
4	t	x=v0 cosa t	y=v0 sina t–gt^2/2
5	0,0	0,0	0,0
6	0,2	3,0	1,8
7	0,4	6,0	3,2
8	0,6	9,0	4,3
9	0,8	11,9	4,9
10	1,0	14,9	5,2
11	1,2	17,9	5,0
12	1,4	20,9	4,5
13	1,6	23,9	3,5
14	1,8	26,9	2,2
15	2,0	29,8	0,5
16	2,2	32,8	–1,8
17	2,4	35,8	–4
18	2,6	38,8	–7,0

Рис. 5.1.2. Координаты мячика в заданные моменты времени

Визуализируем модель, построив график зависимости координаты y от координаты x (траекторию движения тела). Для построения траектории движения мячика используем диаграмму типа *График*.

3.3. Построение диаграмм и графиков

4. При построении графика в качестве категорий использовать диапазон ячеек B5:B18, а в качестве значений — диапазон ячеек C5:C18 (рис. 5.1.3).

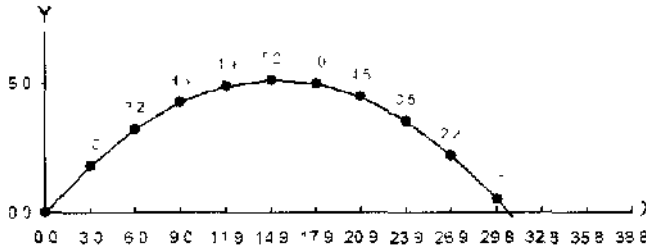


Рис. 5.1.3. Траектория движения мячика

5. По полученному графику (траектории движения мячика) можно качественно судить, попадет ли он в площадку при заданных начальных условиях (расстоянии до площадки и ее длины).

Проект «Бросание мячика в площадку» в электронных таблицах хранится в файле Model.xls в папке ..\IIKT9\Model **Windows-CD**

Практическая работа 5.2

Проект «Графическое решение уравнения»

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux.

Цель работы. Научиться создавать компьютерные модели решения уравнений на языке объектно-ориентированного программирования Visual Basic.

Задание. Разработать проект, в котором приближенно графически решается уравнение $x^3 - \sin x = 0$.

Проект «Графическое решение уравнения» на языке объектно-ориентированного программирования Visual Basic 2005

1. В операционной системе Windows запустить систему объектно-ориентированного программирования Visual Basic 2005 командой [*Программы-Visual Basic 2005 Express Edition*].

2. Разместить на форме (рис. 5.2.1):
   - графическое поле PictureBox1, в котором будет осуществляться построение графика функции $y = x^3 - \sin x$;
   - кнопку Button1 для запуска обработчика события, реализующего построение графика.

В обработчике события осуществим преобразование компьютерной системы координат графического поля в математическую систему координат, удобную для построения графика функции. Нарисуем оси координат и нанесем на них шкалу.

4.11. Практическая работа «Проект «Система координат»

В полученной математической системе координаты находятся в диапазонах $-150 \le X \le 150$ и $-100 \le Y \le 100$. Однако для поиска корней уравнения необходимо построить график функции в диапазоне аргумента $-1{,}5 \le X \le 1{,}5$, на котором функция принимает значения примерно в диапазоне $-1 \le Y \le 1$. Следовательно, необходимо увеличить масштаб графика в 100 раз:

- координаты точек графика необходимо умножить на 100;
- значения шкал осей разделить на 100.

Построение графика функции осуществим в цикле со счетчиком (аргумент X) с использованием метода рисования точки DrawEllipse(Pen1, X * 100, Y * 100, 1, 1), в котором координатами точки являются аргумент функции и значение функции.

3.
```
Dim Graph1 As Graphics
Dim Pen1 As New Pen(Color.Black, 2)
Dim drawBrush As New SolidBrush(Color.Black)
Dim drawFont As New Font("Arial", 10)
Dim X, Y As Single
Private Sub Button1_Click(.)
Graph1 = Me.PictureBox1.CreateGraphics()
Graph1.Clear(Color.White)
'Печать шкал математической системы координат
в компьютерной системе координат
For X = -150 To 150 Step 50
Graph1.DrawString(X / 100, drawFont, drawBrush,
X + 150, 80)
Next X
```

```
For Y = 0 To 200 Step 50
Graph1.DrawString((Y - 100) / 100, drawFont,
drawBrush, 150, 180 - Y)
Next 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 математической системы
'координат
Graph1.DrawLine(Pen1, -150, 0, 300, 0) 'Ось X
Graph1.DrawLine(Pen1, 0, -100, 0, 100) 'Ось Y
For X = -150 To 150 Step 50 'Засечки на оси X
Graph1.DrawLine(Pen1, X, -5, X, 5)
Next X
For Y = -100 To 100 Step 50 'Засечки на оси Y
Graph1.DrawLine(Pen1, -5, Y, 5, Y)
Next Y
'График функции
For X = -1.5 To 1.5 Step 0.01
Y = X ^ 3 - Math.Sin(X)
Graph1.DrawEllipse(Pen1, X * 100, Y * 100, 1, 1)
Next X
End Sub
```

4. Запустить проект на выполнение и щелкнуть по кнопке *График*.

График функции пересекает ось X три раза и, следовательно, уравнение имеет три корня. По графику грубо приближенно можно определить, что $x_1 \approx -0{,}9$, $x_2 \approx 0$ и $x_3 \approx 0{,}9$.

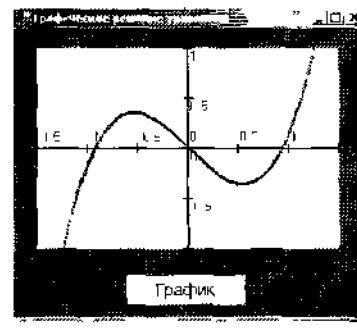


Рис. 5.2.1. Проект «Графическое решение уравнения»

Проект «Графическое решение уравнения» хранится в папке ..\IIKT9\VB2005\Графическое решение уравнения **Windows-CD**

Проект «Графическое решение уравнения» в электронных таблицах Microsoft Excel и OpenOffice.org Calc

1. В операционной системе Windows запустить электронные таблицы Microsoft Excel командой [*Программы-Microsoft Office-Microsoft Office Excel*] или электронные таблицы OpenOffice.org Calc командой [*Программы- OpenOffice-OpenOffice Calc*].
   Или:
   в операционной системе Linux запустить электронные таблицы OpenOffice.org Calc командой [*Программы-OpenOffice OpenOffice Calc*].

Для графического решения уравнения представим функцию $y = x^3 - \sin x$ в табличной форме.

2. В диапазон ячеек B1:P1 (рис. 5.2.2) ввести значения аргумента функции от –1,4 до 1,4 с шагом 0,2.
3. В ячейку B2 введем формулу вычисления значений функции =B1^3–SIN(B1) и скопируем ее в диапазон ячеек C2:P2.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	x	-1,4	-1,2	-1,0	-0,8	-0,6	-0,4	-0,2	0,0	0,2	0,4	0,6	0,8	1,0	1,2	1,4
2	y = x3 - sinx	-1,8	-0,8	-0,2	0,2	0,3	0,3	0,2	0,0	-0,2	-0,3	-0,3	-0,2	0,2	0,8	1,8

Рис. 5.2.2. Табличное представление функции $y = x^3 - \sin x$

4. *Для грубо приближенного определения корней уравнения* построить диаграмму типа *График*.

3.3. Построение диаграмм и графиков

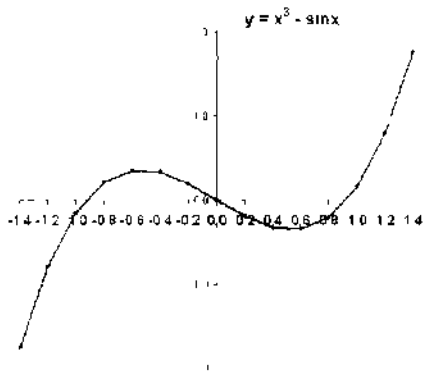


Рис. 5.2.3. Решение уравнения путем построения диаграммы типа *График*

5. График функции пересекает ось X три раза и, следовательно, уравнение имеет три корня.
   По графику приближенно можно определить, что $x_1 \approx 0{,}9$, $x_2 \approx 0$ и $x_3 \approx 0{,}9$.

Проект «Графическое решение уравнения» хранится в файле Model.xls в папке ..\IIKT9\Model **Windows-CD**

Практическая работа 5.3
Проект «Распознавание удобрений»

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows.

Цель работы. Научиться создавать компьютерные модели экспертных систем на языке объектно-ориентированного программирования Visual Basic.

Задание. Разработать проект, в котором необходимо создать экспертную систему распознавания удобрений. Вам даются удобрения, химические реактивы и справочная таблица по взаимодействию удобрений с некоторыми реактивами и предлагается распознать каждое из удобрений.

Проект «Распознавание удобрений» на языке объектно-ориентированного программирования Visual Basic 2005

1. В операционной системе Windows запустить систему объектно-ориентированного программирования Visual Basic 2005 командой [*Программы-Visual Basic 2005 Express Edition*].

Первый шаг в экспертной системе распознавания реализуем с помощью событийной процедуры. Для реализации других шагов создадим четыре общие процедуры.

Создадим графический интерфейс проекта.

2. Поместить на форму (см. рис. 5.3.1):
   - список `ListBox1`, в который будем помещать результаты распознавания (названия удобрений);
   - кнопку `Button1` для запуска обработчика события.

В событийной и общих процедурах задавать вопросы и реагировать на ответы пользователя будем с использовани-

ем функции MsgBox(). Диалоговое окно функции будет содержать вопрос, кнопки *Да* и *Нет* и информационный значок *о типе сообщения*.

4.5. Функции в языках объектно-ориентированного и алгоритмического программирования

Добавление названий удобрений в окно списка реализуем с помощью метода Item.Add().

Первую развилку экспертной системы (условие *При взаимодействии с H_2SO_4 выделяется бурый газ*) реализуем в форме событийной процедуры, которая содержит вызовы общих процедур Щелочь1() и Соль():

```
3. Dim A As Byte
   Private Sub Button1_Click(…)
   A = MsgBox("При взаимодействии с серной
   кислотой выделяется бурый газ?", 36,
   "Первый вопрос")
   If A = 6 Then Щелочь1() Else Соль()
   End Sub
```

Для распознавания удобрений первой группы (1-го и 2-го) создадим общую процедуру Щелочь1() (условие *При взаимодействии с раствором щелочи ощущается запах аммиака*):

```
4. Sub Щелочь1()
   A = MsgBox("При взаимодействии со щелочью
   ощущается запах аммиака?", 36,
   "Второй вопрос")
   If A = 6 Then ListBox1.Items.Add
                 ("1.Аммиачная селитра")
            Else ListBox1.Items.Add
                 ("2.Натриевая селитра")
   End Sub
```

Для распознавания удобрений второй группы сначала необходимо создать общую процедуру Соль() (условие *При взаимодействии с $BaCl_2$ выпадает белый осадок*), которая содержит вызовы общих процедур Щелочь2() и Внешний_вид():

```
5. Sub Соль()
   A = MsgBox("При взаимодействии с солью
   выпадает белый осадок?", 36, "Второй вопрос")
   If A = 6 Then Щелочь2() Else Внешний_вид()
   End Sub
```

Для распознавания 3-го и 4-го удобрений создать общую процедуру Щелочь2() (условие *При взаимодействии с раствором щелочи ощущается запах аммиака*):

```
6. Sub Щелочь2()
   A = MsgBox("При взаимодействии со щелочью
   ощущается запах аммиака?", 36, "Третий вопрос")
   If A = 6 Then
           ListBox1.Items.Add("3.Сульфат аммония")
              Else
           ListBox1.Items.Add("4.Суперфосфат")
   End Sub
```

Для распознавания 5-го и 6-го удобрений создать общую процедуру Внешний_вид() (условие *Внешний вид — розовые кристаллы*):

```
7. Sub Внешний_вид()
   A = MsgBox("Розовые кристаллы?", 36,
   "Третий вопрос")
   If A = 6 Then
   ListBox1.Items.Add("5.Сильвинит")
   Else ListBox1.Items.Add("6.Калийная соль")
   End Sub
```

Компьютерный эксперимент. Работа с экспертной системой позволит более эффективно спланировать и провести распознавание удобрений в процессе выполнения лабораторной работы по химии.

8. Запустить проект щелчком по кнопке *Распознать*.
   Экспертная система начнет задавать вопросы, на которые надо отвечать на основе проводимых химических опытов. Проделать процедуру распознавания для каждого вещества.
   В результате в окно списка будут выведены названия всех удобрений.

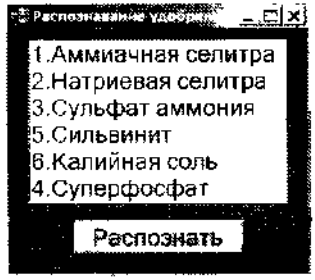


Рис. 5.3.1. Результаты распознавания удобрений

Проект «Распознавание удобрений» на языке Visual Basic хранится в папке ..\IKT9\VB2005\Распознавание удобрений **Windows-CD**

Практическая работа 5.4

Проект «Модели систем управления»

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows.

Цель работы. Научиться создавать компьютерные модели систем управления без обратной связи, с обратной связью и автоматической обратной связью на языке объектно-ориентированного программирования Visual Basic.

Задание. Разработать проект, в котором управляемым объектом будет точка, которую управляющий объект (пользователь) должен переместить в центр мишени (окружности). Прямое управление положением точки будем производить путем нажатия на кнопки, которые перемещают объект влево и вправо, вверх и вниз. Рассмотрим три варианта:

1) обратная связь отсутствует, так как текущие положения точки в процессе управления невидимы;
2) обратная связь присутствует, так как текущие положения точки в процессе управления видимы;
3) автоматическая обратная связь присутствует.

Проект «Модели систем управления» на языке объектно-ориентированного программирования Visual Basic 2005

1. В операционной системе Windows запустить систему объектно-ориентированного программирования Visual Basic 2005 командой [*Программы-Visual Basic 2005 Express Edition*].

Вариант 1 «Система управления без обратной связи». Для создания графического интерфейса проекта поместим на форму (рис. 5.4.1):

2. • графическое поле `PictureBox1`, по которому будет перемещаться точка;
   • кнопку `Button1` для запуска обработчика события, реализующего вывод первоначального положения точки и мишени (окружности);
   • четыре кнопки `Button2`, `Button3`, `Button4` и `Button5` для управления движением точки;
   • кнопку `Button6` для запуска обработчика события, реализующего вывод конечного положения точки.

3. Выделить объект PictureBox1 и с помощью диалогового окна *Свойства* установить для свойства Size значение, например, 200;200.

Создадим обработчик события, реализующего рисование в графическом поле точки (управляемого объекта) и окружности (мишени).

При каждом запуске обработчика события будем рисовать точку с различными координатами. Случайные значения координат получим с помощью оператора Randomize() и функции Rnd(), которая генерирует случайные числа в интервале $0 \le X < 1$. Начальные координаты точки должны быть заданы целыми числами и соответствовать размерам графического поля, т. е. $0 \le X < 200$ и $0 \le Y < 200$. Поэтому значения функции Rnd() необходимо увеличить в 200 раз и выделить целую часть числа с использованием функции Int().

Для рисования в графическом поле точки и окружности используем графические методы.

4.
```
Dim Graph1 As Graphics
Dim Pen1 As New Pen(Color.Black, 3)
Dim Brush1 As New SolidBrush(Color.Black)
Dim X, Y As Integer
Private Sub Button1_Click(..)
Graph1 = Me.PictureBox1.CreateGraphics()
Graph1.Clear(Color.White)
Randomize()
X = Int(Rnd() * 200)
Y = Int(Rnd() * 200)
Graph1.DrawEllipse(Pen1, X, Y, 2, 2)
Graph1.FillEllipse(Brush1, X, Y, 2, 2)
Graph1.DrawEllipse(Pen1, 90, 90, 20, 20)
End Sub
```

Четыре обработчика событий перемещения точки влево и вправо, вверх и вниз должны обеспечивать соответствующие изменения координат точки. В компьютерной системе координат для перемещения влево координата точки x должна уменьшаться, а для перемещения вправо увеличиваться. Для перемещения вниз координата точки y должна увеличиваться, а для перемещения вверх — уменьшаться.

Обратная связь должна отсутствовать, поэтому текущие положения точки не будут отображаться в графическом поле.

Например, обработчик события перемещения точки влево будет следующим:

```
5. Private Sub Button4_Click(…)
   X = X - 1
   End Sub
```

Возможность увидеть результаты управления, т. е. попала точка в центр мишени или нет, обеспечивает обработчик события вывода конечного положения точки:

```
6. Private Sub Button6_Click(…)
   Graph1.DrawEllipse(Pen1, X, Y, 2, 2)
   Graph1.FillEllipse(Brush1, X, Y, 2, 2)
   End Sub
```

7. Запустить проект и для вывода первоначального положения точки и мишени щелкнуть по кнопке *Мишень и точка*.
Щелчками по кнопкам управления направления перемещения точки (кнопки со стрелками) постараться переместить точку в центр окружности.
Щелкнуть по кнопке *Результат*. Управление перемещением точки производится без обратной связи, поэтому попасть в центр окружности довольно трудно.

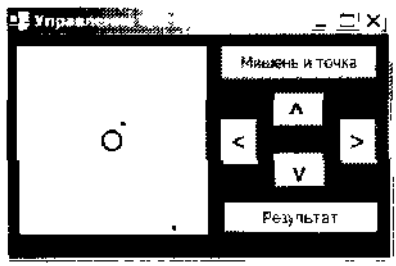


Рис. 5.4.1. Модель системы управления без обратной связи

Проект «Управления без обратной связи» на языке Visual Basic хранится в папке ..\IIKT9\VB2005\Управление без обратной связи **Windows-CD**

Вариант 2 «Система управления с обратной связью». За основу возьмем проект «Управление без обратной связи». Для осуществления обратной связи будем рисовать текущее положение точки в графическом поле, а также выводить текущие координаты точки на надписи (рис. 5.4.2).

8. Поместить на форму две метки Label1 и Label2 для вывода текущих координат точки.

В программные коды обработчиков событий перемещения точки добавим строки рисования точки и вывода на метки ее текущих координат. Обработчик события перемещения точки влево примет вид:

9.
```
Private Sub Button3_Click(…)
X = X - 1
Graph1.DrawEllipse(Pen1, X, Y, 2, 2)
Graph1.FillEllipse(Brush1, X, Y, 2, 2)
Label1.Text = X
Label2.Text = Y
End Sub
```

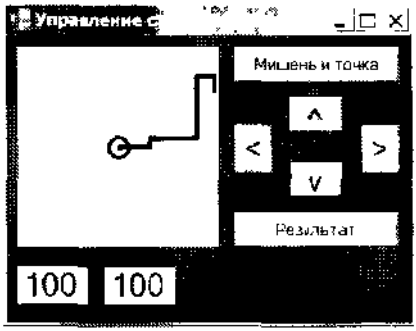


Рис. 5.4.2. *Модель системы управления с обратной связью*

10. Запустить проект и осуществить попадание точки в мишень, имеющую координаты центра окружности (100,100).
Легко убедиться, что использование обратной связи обеспечивает уверенное попадание точки в центр мишени.

Проект «Управление с обратной связью» на языке Visual Basic хранится в папке ..\IIKT9\VB2005\Управление с обратной связью\ Windows-CD

Вариант 3 «Система управления с автоматической обратной связью». За основу возьмем проект «Управление с обратной связью».

Уберем из графического интерфейса:

11. • четыре кнопки Button2, Button3, Button4 и Button5 для управления движением точки и кнопку Button6, выводящую результат.

Поместим на графический интерфейс:

12. • кнопку Button2 для создания автоматической обратной связи.

Для осуществления автоматической обратной связи используем корректировку координат с использованием инструкции выбора **Select Case**.

Обработчик события, реализующий корректировку положения точки, примет вид:

13.
```
Private Sub Button2_Click_1(...)
X2 = 100
Y2 = 100
'Автоматическая корректировка координаты X
Select Case X2 - X1
    Case Is > 0
    X1 = X1 + 1
    Case Is < 0
    X1 = X1 - 1
    Case Is = 0
    X1 = X1
End Select
'Автоматическая корректировка координаты Y
Select Case Y2 - Y1
    Case Is > 0
    Y1 = Y1 + 1
    Case Is < 0
    Y1 = Y1 - 1
    Case Is = 0
    Y1 = Y1
End Select
Label1.Text = X1
Label2.Text = Y1
Graph1.DrawEllipse(Pen1, X1, Y1, 2, 2)
Graph1.FillEllipse(Brush1, X1, Y1, 2, 2)
End Sub
```

14. Запустить проект и нажатием на кнопку Шаг осуществить попадание точки в мишень, имеющую координаты центра окружности (100,100) (рис. 5.4.3).

Рис. 5.4.3. Модель системы управления с автоматической обратной связью

Проект «Управление с автоматической обратной связью» на языке Visual Basic хранится в папке ..\IIKT9\VB2005\Управление с автоматической обратной связью

Windows-CD

Ответы и решения к заданиям для самостоятельного выполнения

Глава 1. Кодирование и обработка графической и мультимедийной информации

1.1. Ответ № 2.

1.2. 100 битов.

1.3. 100 байтов.

1.4. Разрешающая способность сканера 1200 dpi (dot per inch — точек на дюйм) означает, что на отрезке длиной 1 дюйм сканер способен различить 1200 точек.
Переведем разрешающую способность сканера из точек на дюйм (1 дюйм = 2,54 см) в точки на сантиметр:
1200 dpi : 2,54 ≈ 472 точек/см.
Следовательно, размер изображения в точках составит 4720 × 4720 точек.
Общее количество точек изображения равно:
4720 · 4720 = 22 278 400.
Информационный объем файла равен:
24 бита · 22 278 400 = 534 681 600 битов ≈ 64 Мбайт.

1.5. Выразим размер диагонали в сантиметрах:
2,54 см/дюйм · 17 дюймов = 43,18 см.
У стандартных мониторов соотношение между высотой и шириной экрана равно 0,75.
Определим ширину экрана. Пусть ширина экрана равна L, тогда высота равна $0{,}75L$. По теореме Пифагора имеем:

$L + (0{,}75L) = 43{,}18$,
$1{,}5625L \approx 1864{,}5$,

$L \approx 1193$,
$L \approx 34{,}5$ см.
Количество точек по ширине экрана равно:
345 мм : 0,28 мм = 1232.
Максимально возможным разрешением экрана монитора является 1152 × 864.

1.6.

Цвет	Интенсивность базовых цветов		
	Красный	Зеленый	Синий
Черный	00000000	00000000	00000000
Красный	11111111	00000000	00000000
Зеленый	00000000	11111111	00000000
Синий	00000000	00000000	11111111
Голубой	00000000	11111111	11111111
Пурпурный	11111111	00000000	11111111
Желтый	11111111	11111111	00000000
Белый	11111111	11111111	1111111

1.7.

Цвет	Формирование цвета
Белый	$C = 0, M = 0, Y = 0$
Красный	$Y + M = W - B - G$
Зеленый	$Y + C = W - B - R$
Синий	$M + C = W - G - R$
Голубой	$W - R = G + B$
Пурпурный	$W - G = R + B$
Желтый	$W - B = R + G$

1.8. Ответ № 4.

1.9. Ответ № 3.

1.10. а) 8 битов · 8000 с · 10 с = 640 000 битов ≈ 78 Кбайт;
б) 16 битов · 48 000 с · 10 с · 2 = 15 360 000 битов ≈ ≈ 1,8 Мбайт.

1.11. Информационный объем дискеты 512 байтов · 2847 = = 1 457 664 байтов = 1423,5 Кбайт.

а) Информационный объем 1 секунды звукового файла низкого качества: 8 битов · 8 000 с = 64 000 бит/с = = 8 000 байт/с ≈ 7,8 Кбайт/с.

Длительность, умещающаяся на дискете:
1423,5 Кбайт : 7,8 Кбайт/с = 182, 5 с ≈ 3 минуты.

б) Информационный объем 1 секунды звукового файла высокого качества: 16 битов · 2 · 48 000 с = 1 536 000 бит/с = = 192 000 байт/с ≈ 187,5 Кбайт/с.

Длительность, умещающаяся на дискете:
1423,5 Кбайт : 187,5 Кбайт/с ≈ 7,6 секунды.

Глава 2. Кодирование и обработка текстовой информации

2.1. 4000 байтов.

2.2. $N = 2 \Rightarrow 256 = 2 \Rightarrow I = 8$ битов; $I = 8$ битов $\cdot 100 =$ $= 800$ битов $= 100$ байтов.

$N = 2 \Rightarrow 65\,256 = 2 \Rightarrow I = 16$ битов; $I = 16$ битов $\cdot 100 =$ $= 1600$ битов $=$ 200 байтов, где I — информационная емкость сообщения, набранного пользователем.

2.8. Полужирный курсив, подчеркнутый.

2.9. Ответ № 1.

2.10. Ответ № 4.

Глава 3. Кодирование и обработка числовой информации

3.1. $3{,}14 = 3 \cdot 10 + 1 \cdot 10 + 4 \cdot 10$.
$10{,}1 = 1 \cdot 2 + 0 \cdot 2 + 1 \cdot 2$.

3.2. В 10 раз, в 2 раза.

3.3. 2.

3.4. 2, 10.

3.6. $1010 + 10 = 1100$.

$1010 - 10 = 1000$.

$1010 \cdot 10 = 10100$.

$1010 : 10 = 101$.

3.7. Переведем десятичное число в двоичную систему счисления:

$10 = 1010$.

Представим его в компьютерном формате целого неотрицательного числа:

0	0	0	0	1	0	1	0

Представим его в компьютерном формате целого числа со знаком:

Знак	Число														
0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0

3.8. B2, A1:A2, D1:E2, A4:C4.

3.9. =A1+B1, =A3–B5, =C1*C2, =A10/B10.

3.10. 6, 6, 21. Лист *Задание 3.10* в файле ..\IIKT9\Calc.xls на Windows-CD.

3.11. 16. Лист *Задание 3.11* в файле ..\IIKT9\Calc.xls на Windows-CD.

3.12. 2. Лист *Задание 3.12* в файле ..\IIKT9\Calc.xls на Windows-CD.

Глава 4. Основы алгоритмизации и объектно-ориентированного программирования и глава 5. Моделирование и формализация

Проекты, рассмотренные в главах, хранятся на Windows-CD в следующих папках:

- в формате OpenOffice.org Basic в папке ..\IIKT9\Basic\;
- в формате Gambas в папке ..\IIKT9\Gambas\;
- в формате Visual Basic 2005 Express Edition в папке ..\IIKT9\VB2005\.

В процессе выполнения проектов на языках объектно-ориентированного программирования Visual Basic 2005 Express Edition и Gambas должна производиться запись информации на диск. Поэтому необходимо перед запуском проектов скопировать папки проектов с Windows-CD на жесткий диск.

Учебное издание

Угринович Николай Дмитриевич

ИНФОРМАТИКА И ИКТ
Учебник для 9 класса

Ведущий редактор *О. Полежаева*
Художественный редактор *С. Инфанте*
Технический редактор *О. Лапко*
Корректор *Е. Клитина*
Компьютерная верстка *В. Носенко*

Подписано в печать 19.03.09. Формат 60×90 $^{1}/_{16}$.
Бумага офсетная. Гарнитура Школьная. Печать офсетная.
Усл. печ. л. 18,5. Тираж 25000 экз. Заказ №4990053

Издательство «БИНОМ. Лаборатория знаний»
Адрес для переписки: 125167, Москва, проезд Аэропорта, 3
Телефон: (499) 157-52 72, e-mail: binom@Lbz.ru
http://www.Lbz.ru

Отпечатано на ОАО «Нижполиграф»,
603006 Нижний Новгород, ул Варварская, 32