Н. Д. Угринович

ПРОФИЛЬНЫЙ УРОВЕНЬ

ИНФОРМАТИКА и ИКТ

11

БИНОМ

ПРОФИЛЬНЫЙ УРОВЕНЬ

Н. Д. Угринович

ИНФОРМАТИКА и ИКТ

Учебник для 11 класса

2-е издание,
исправленное и дополненное

Москва
БИНОМ. Лаборатория знаний
2009

УДК 004.9
ББК 32.97
У27

Угринович Н. Д.

У27 Информатика и ИКТ. Профильный уровень : учебник для 11 класса / Н. Д. Угринович. — 2-е изд., испр. и доп. — М. : БИНОМ. Лаборатория знаний, 2009. — 308 с. : ил.

ISBN 978-5-9963-0047-1

Учебник по курсу «Информатика и ИКТ. Профильный уровень» для 11 класса ориентирован на преподавание в общеобразовательных учреждениях курса «Информатика и ИКТ» на профильном уровне. Учебник соответствует образовательному стандарту, утвержденному Министерством образования и науки РФ. В учебнике рассматриваются технологии создания и обработки информации, системный подход к моделированию, формализация и ее визуализация с использованием интерактивных компьютерных моделей, базы данных и СУБД, коммуникационные технологии, проблемы информационного общества. Учебник мультисистемный, так как практические работы могут выполняться в операционных системах Windows и Linux. В учебник помещены тесты для подготовки к ЕГЭ по курсу «Информатика и ИКТ».

УДК 004.9
ББК 32.97

По вопросам приобретения обращаться:

(499) 157-5272, e-mail: binom@Lbz.ru
http://www.Lbz.ru

ISBN 978-5-9963-0047-1

Оглавление

Рекомендации по использованию учебника

1. Учебник «Информатика и ИКТ-11. Профильный уровень» обеспечивает изучение курса «Информатика и ИКТ» в 11 классе на профильном уровне.
   Учебник входит в состав учебно-программного комплекса, включающего:
   - учебники для основной школы: «Информатика и ИКТ-8» и «Информатика и ИКТ-9»;
   - учебники для старшей школы на базовом уровне: «Информатика и ИКТ-10. Базовый уровень» и «Информатика и ИКТ-11. Базовый уровень»;
   - учебники для старшей школы на профильном уровне: «Информатика и ИКТ-10. Профильный уровень» и **«Информатика и ИКТ-11. Профильный уровень»**;
   - учебное пособие и CD-ROM по элективному курсу для старшей школы «Исследование информационных моделей»;
   - методическое пособие для учителей «Преподавание курса «Информатика и ИКТ» в основной и старшей школе», к которому прилагаются:
     - Windows-CD, содержащий свободно распространяемую программную поддержку курса, готовые компьютерные проекты, рассмотренные в учебниках, тесты и методические материалы для учителей;
     - Visual Studio-CD (выпускается по лицензии корпорации Microsoft), содержащий дистрибутивы систем объектно-ориентированного программирования языков Visual Basic .NET, Visual C# и Visual J#;
     - Linux-DVD (выпускается по лицензии компании AltLinux), содержащий операционную систему Linux и программную поддержку курса;
     - TurboDelphi-CD (выпускается по лицензии компании Borland), содержащий систему объектно-ориентированного программирования Turbo Delphi.
2. В практических работах указана операционная система и необходимое для их выполнения программное обеспечение, которые обозначаются значками приложений. В случае выделения количества часов на предмет «Информатика и ИКТ» не большего, чем указано в Федеральном базисном учебном плане, рекомендуется выполнять практические задания компьютерного практи-

кума в одной операционной системе (Windows или Linux).

3. Возможно выполнение практических занятий во внеурочное время в компьютерном школьном классе или дома.
4. Проекты на языке программирования Turbo Delphi совместимы по формату с проектами на языках Delphi 6, Delphi 2006 и Delphi 2007.
5. В учебнике используются ссылки на внешние источники информации, в которых можно более подробно изучить данную тему (учебники по информатике и другим предметам, CD-диски и Интернет).
6. В тексте пособия приняты следующие обозначения и шрифтовые выделения:
   - шрифтом Arial выделены имена программ, файлов и Интернет-адреса;
   - *курсивом* выделены названия диалоговых окон, пунктов меню и элементов управления (текстовых полей, кнопок и т. д.) графического интерфейса операционных систем и приложений;
   - **полужирным шрифтом** выделены важные термины и понятия.
7. Важная информация выделена в тексте восклицательным знаком, а формулы — цифровым обозначением.
8. Абзацы, содержащие дополнительную интересную информацию, выделены значком  .
9. Дополнительные материалы и интерактивные тесты *для проверки усвоения материала находятся в Интернете* по адресу: http://iit.metodist.ru

Повторение

Окружающий мир как иерархическая система

5.1. Окружающий мир как иерархическая система

Информатика и ИКТ-9

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 атомов; атомы, 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**.

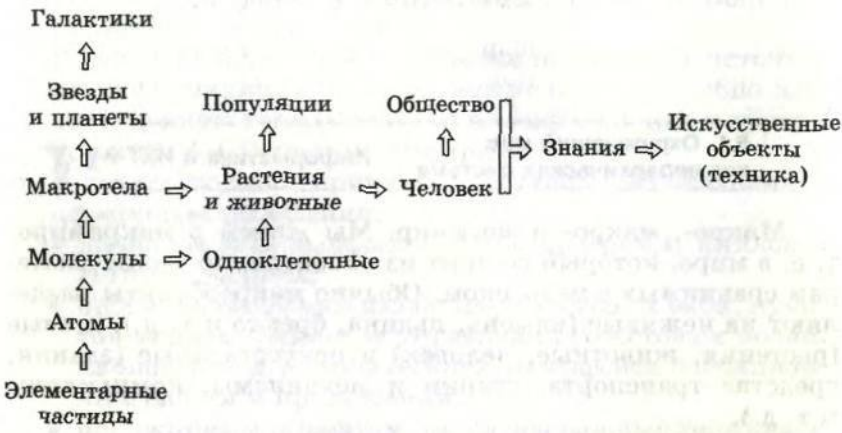
Все объекты от галактик до атомов состоят из **вещества**, при этом все материальные объекты взаимодействуют друг с другом и поэтому обладают **энергией**. Поднятое над поверхностью Земли тело обладает механической энергией, нагретый чайник — тепловой, заряженный проводник — электрической, а ядра атомов — атомной.

Окружающий мир можно представить в виде иерархического ряда объектов: элементарных частиц, атомов, химических веществ, макрообъектов, звезд и галактик. При этом на уровнях химических веществ и макрообъектов в этом иерархическом ряду образуется ответвление — другой ряд, связанный с живой природой.

В живой природе также существует иерархия: молекулы — клетки — организмы — группы организмов (популяции).

Вершиной эволюции жизни на Земле является человек, который, в свою очередь, 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.



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. Считаем мы 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.

Целостность системы. Необходимым условием существования системы является ее **целостное функционирование**. Система является не набором отдельных объектов, а совокупностью взаимосвязанных элементов.

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 систем. 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.

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.

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 толщиной с человеческий волос, сделанная из нанотрубок, способна удерживать груз в сотни килограммов. Электрические свойства нанотрубок могут меняться, что делает их одним из основных материалов наноэлектроники.

Глава 1

Построение и исследование информационных моделей

При изучении данной главы рекомендуется установить следующее программное обеспечение для операционных систем Windows и Linux:

	• OpenOffice.org Calc;	Windows-CD	
	• Windows XP;	Первая помощь ПО 1.0.	CD-2
	• Windows Vista;		CD-4
	• Microsoft Office 2003 (Excel 2003);		CD-6:7
	• *Microsoft Office 2007* (Excel 2007);		CD-5
	• Visual Basic 2005 Express Edition;	VisualStudio-CD	
	• Turbo Delphi 2006;	TurboDelphi-CD	
	• OpenOffice.org Calc.	Linux-DVD	

1.1. 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 значения тех или иных параметров. 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 на естественном языке. Такая модель выделяет существенные, с точки зрения целей проводимого исследования, свойства (параметры) объекта, а несущественными свойствами пренебрегает.

Формальная модель. На втором этапе создается формальная модель, т. е. описательная информационная модель записывается с помощью какого-либо формального языка. В такой модели с помощью формул, уравнений, неравенств и т. д. фиксируются формальные соотношения между исходными и искомыми величинами, а также накладываются ограничения на допустимые значения этих величин.

Однако далеко не всегда удается найти формулы, явно выражающие искомые величины через исходные. 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 два пути решения этой задачи:

- создание проекта на одном из языков программирования;
- построение компьютерной модели с использованием некоторого приложения, например электронных таблиц.

В процессе создания компьютерной модели полезно разработать удобный графический интерфейс, который позволит визуализировать формальную модель, а также реализо-

вать 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 программы на одном из языков программирования, ее нужно запустить на выполнение и получить результаты.

Если компьютерная модель исследуется в приложении, например в электронных таблицах, можно провести сортировку или поиск данных, построить диаграмму или график и т. д.

Анализ полученных результатов и корректировка исследуемой модели. Пятый этап представляет собой анализ полученных результатов и корректировку исследуемой модели. В случае отличия результатов, полученных при исследовании информационной модели, от измеренных параметров 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 таких случаях необходимо провести корректировку модели, причем уточнение модели может проводиться многократно, пока анализ результатов не покажет их соответствие изучаемому объекту.

Визуализация формальных моделей. В процессе исследования формальных моделей часто производится их визуализация. Для визуализации алгоритмов используются блок-схемы, пространственных соотношений параметров объектов — 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.

Вопросы для размышления

1. В каких случаях могут быть опущены отдельные этапы построения и исследования модели? Приведите известные вам примеры создания моделей в процессе изучения физики, химии, биологии, математики, географии и других предметов.

1.2. Построение и исследование физических моделей

1.2.1. Построение формальной модели движения тела, брошенного под углом к горизонту

Рассмотрим процесс построения и исследования модели на конкретном примере движения тела, брошенного под углом к горизонту.

Физика-9

Содержательная постановка задачи «Бросание мячика в стенку». В процессе тренировок теннисистов используются автоматы по бросанию мячика. Необходимо задать автомату необходимую скорость и угол бросания мячика для попадания в стенку определенной высоты, находящуюся на известном расстоянии.

Описательная модель. Сначала построим описательную модель процесса движения тела с использованием объектов, понятий и законов физики, т. е. в данном случае идеализированную модель движения объекта. Из условия задачи можно сформулировать следующие основные допущения:

- 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^2, следовательно, движение по оси Y можно считать равноускоренным;
- скорость бросания мячика мала, поэтому сопротивлением воздуха можно пренебречь, следовательно, движение по оси X можно считать равномерным.

Формальная модель. Для формализации модели обозначим величины:

- начальную скорость мячика — v_0;
- угол бросания мячика — α;
- высоту стенки — h;
- расстояние до стенки — s.

Изобразим график движения мячика (рис. 1.1).

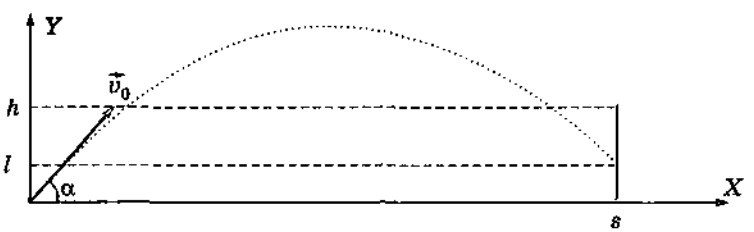


Рис. 1.1. Бросание мячика в стенку

Используем известные из курса физики формулы равномерного и равноускоренного движения для определения координат мячика. Дальность x и высоту y при заданной начальной скорости v_0 и угле бросания α для любого момента времени t можно вычислить по формулам:

$$\begin{aligned} x &= v_0 \cdot \cos\alpha \cdot t, \\ y &= v_0 \cdot \sin\alpha \cdot t - g \cdot t^2/2. \end{aligned} \qquad (1.1)$$

Чтобы определить, попадет ли мячик в стенку, необходимо вычислить его координату y в момент времени, когда он будет находиться на расстоянии s. Из первой формулы выражаем время, которое понадобится мячику, чтобы преодолеть расстояние s:

$$t = s/v_0 \cdot \cos\alpha.$$

Подставляем это значение времени t в формулу для вычисления координаты y. Получаем l — высоту мячика над землей на расстоянии s (см. рис. 1.1):

$$l = s \cdot \operatorname{tg}\alpha - g \cdot s^2/2 \cdot v_0^2 \cdot \cos^2\alpha. \qquad (1.2)$$

Формализуем теперь условие попадание мячика в мишень. Попадание произойдет, если значение высоты мячика l будет удовлетворять условию в форме неравенства:

$$0 \le l \le h.$$

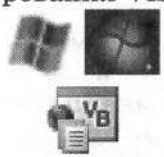
Если $l < 0$, то это означает «недолет», а если $l > h$, то это означает «перелет».

Вопросы для размышления

1. Какие условия входят в построение качественной модели бросания мячика в стенку?

1.2.2. Компьютерная модель движения тела на языке Visual Basic

На основе формальной модели, описывающей движение тела, брошенного под углом к горизонту, создадим компьютерную модель с использованием системы программирования Visual Basic.



Проект «Бросание мячика в стенку» на языке Visual Basic

Создадим сначала графический интерфейс проекта «Бросание мячика в стенку» (рис. 1.2).

1. Разместить на форме:
   - четыре текстовых поля для ввода значений: `TextBox1` — начальной скорости, `TextBox2` — угла бросания мячика, `TextBox3` — расстояния до стенки и `TextBox4` — высоты стенки;
   - надпись `Label1` для вывода высоты мячика на заданном расстоянии;
   - надпись `Label2` для вывода текстового сообщения о результатах броска;
   - десять надписей для вывода имен переменных и единиц измерения;
   - кнопку `Button1` для запуска событийной процедуры вычисления результатов бросания мячика;
   - кнопку `Button2` для демонстрации траектории движения мячика.

Обработчик события — вычисление результатов бросания мячика

2. Создать программный код обработчика события, который определят попадание мячика в стенку. В этом коде:
   - объявить вещественные константы одинарной точности `G` (ускорение свободного падения g) и `Pi` (число π);

- объявить вещественные переменные одинарной точности V0 (начальная скорость v_0), A (угол бросания α), S (расстояние до стенки s), H (высота стенки h) и L (высота мячика l);
- присвоить переменным V0, A, S, H значения, введенные в текстовые поля, с использованием функции преобразования строки в вещественное число Val();
- вычислить высоту мячика L на заданном расстоянии по формуле (1.2);
- вывести высоту мячика L на надпись Label1;
- вывести текстовое сообщение о результатах броска на надпись Label2 с использованием оператора **Select Case**, в котором в качестве условия проверяется значение переменной L.

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 значения углов из градусов в радианы с использованием константы Pi.

```
Const G As Single = 9.81
Const Pi As Single = 3.14
Dim V0, A, S, L, H As Single
Private Sub Button1_Click(…)
'Ввод начальных значений
V0=Val(TextBox1.Text)
A=Val(TextBox2.Text)
S=Val(TextBox3.Text)
H=Val(TextBox4.Text)
'Попадание в стенку
L=S*Math.Tan(A*Pi/180)-(G*S^2)/(2*V0^2*
Math.Cos(A*Pi/180)^2)
Label1.Text=L
Select Case L
Case Is <0
Label2.Text="Недолет"
Case Is >H
Label2.Text="Перелет"
Case Else
Label2.Text="Попадание"
End Select
End Sub
```

Обработчик события — демонстрация траектории движения мячика

Для визуализации компьютерной модели построим в графическом поле траекторию движения мячика.

3. Поместить дополнительно на форму графическое поле PictureBox1. 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.

Проект «Система координат» Информатика и ИКТ-9

4. 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 осуществить 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.

```
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)
Dim X, Y, T As Single
Private Sub Button2_Click(…)
Graph1=Me.PictureBox1.CreateGraphics()
Graph1.Clear(Color.White)
'Печать шкал математической системы координат
'в компьютерной системе координат
For X=0 To 400 Step 50
Graph1.DrawString(X/10, drawFont, drawBrush, X,
200)
Next X
```

```
For Y=20 To 220 Step 50
Graph1.DrawString((Y-20)/10, drawFont,
drawBrush, 0, 220-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
Graph1.DrawLine(Pen1, 0, -20, 0, 200) 'Ось Y
'Стенка
Graph1.DrawLine(Pen1, S*10, 0, S*10, H*10)
'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, попадания в стенку 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 стенку, затем угол, при котором мячик не долетит до стенки. Вычислим среднее значение углов, составляющих «вилку», и проверим, попадет ли при этом значении мячик в стенку. Если он попадет в стенку, то задача выполнена, если не попадет, то рассмотрим новую «вилку» и т. д.

5. Запустить проект и ввести значения начальной скорости, угла бросания, расстояния до стенки и ее высоты. Щелкнуть по кнопкам *Вычислить* и *Показать*. На надписи будут выведены результаты, а в графическом поле появится траектория движения тела (см. рис. 1.2). Подобрать значения начальной скорости и угла бросания мячика, обеспечивающие его попадание в стенку.

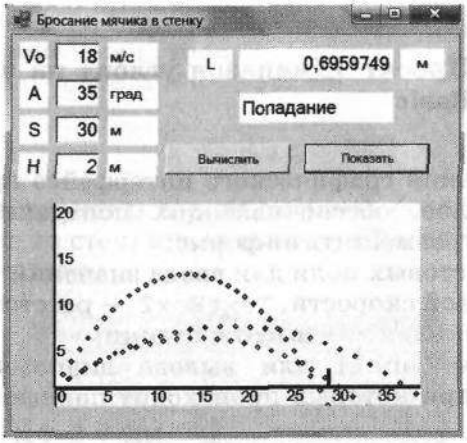


Рис. 1.2. Компьютерный эксперимент по бросанию мячика в стенку

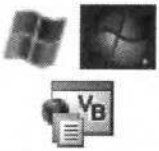
Например, при скорости бросания мячика $v_0 = 18$ м/с и угле бросания $\alpha = 35°$ мячик попадет в стенку высотой $h = 2$ м и находящуюся на расстоянии $s = 30$ м на высоте $l = 0{,}6959749$ м.

Проект «Бросание мячика в стенку» хранится в папке ..\IIKT11Prof\VB2005\Бросание мячика в стенку **Windows-CD**

Анализ результатов. Полученная точность высоты попадания мячика в стенку $l = 0{,}6959749$ м не имеет физического смысла и определяется типом переменной. Так как L является переменной одинарной точности, то ее значение вычисляется с точностью семи значащих цифр. Исходные данные заданы с точностью две значащие цифры, поэтому необходимо и результат округлить до двух значащих цифр: $l = 0{,}70$ м.

Корректировка модели. Рассмотренный выше проект «Бросание мячика в стенку» позволяет задавать различные значения скорости и угла бросания мячика. Предположим, что броски мячика производятся с одинаковой скоростью, и попробуем определить диапазон углов, при которых происходит попадание мячика в стенку.

Для этого необходимо в цикле со счетчиком по углу бросания вычислить положение мячика на расстоянии стенки и определить те углы, которые соответствуют высоте стенки.



Проект «Диапазон углов» на языке Visual Basic

1. Для создания графического интерфейса проекта «Диапазон углов, обеспечивающих попадание в стенку» (рис. 1.3) разместить на форме:
   - три текстовых поля для ввода значений: `TextBox1` — начальной скорости, `TextBox2` — расстояния до стенки и `TextBox3` — высоты стенки;
   - надпись `Label1` для вывода диапазона значений углов, при которых происходит попадание мячика в стенку;
   - восемь надписей для обозначения переменных и единиц измерения;
   - кнопку `Button1` для запуска обработчика события.

2. Для каждого значения скорости бросания мячика получить диапазон значений углов, обеспечивающих попадание мячика в стенку. Для этого в цикле со счетчиком, которым является переменная `A` (угол бросания), вычислять высоту мячика в момент его нахождения на расстоянии стенки. Для каждой высоты мячика с помощью оператора условного перехода в сокращенной форме **`If-Then-End If`** проверять, справедливо ли условие `0<=L` **`And`** `L<=H` (попадет ли мячик в стенку).
   Если условие справедливо, то значение переменной `A` (угол бросания) выводить на надпись `Label1` с использованием функции преобразования числа в строку `Str(A)`.

   Программный код обработчика события:

```
Const G As Single = 9.81
Const Pi As Single = 3.14
Dim V0, S, H, L As Single, A As Integer
Private Sub Button1_Click(…)
'Ввод начальных значений
V0=Val(TextBox1.Text)
S=Val(TextBox2.Text)
H=Val(TextBox3.Text)
Label1.Text=""
For A=0 To 90
'Попадание в стенку
L=S*Math.Tan(A*Pi/180)-(G*S^2)/(2*V0^2*
Math.Cos(A*Pi/180)^2)
```

```
'Вывод значений диапазона углов
If 0<=L And L<=H Then
Label1.Text=Label1.Text+Str(A)
End If
Next A
End Sub
```

3. Запустить проект и ввести скорость бросания мячика, расстояние до стенки и ее высоту. Щелкнуть по кнопке *Диапазон углов*.

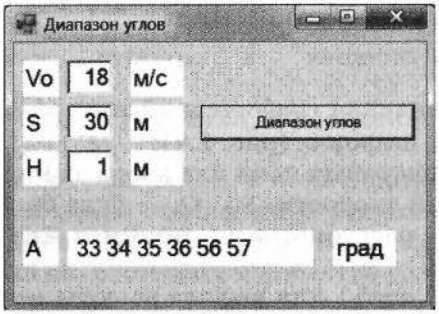


Рис. 1.3. Диапазон углов, обеспечивающих попадание мячика в стенку

Получим важный результат — оказывается, существуют два диапазона углов: от 33° до 36° и от 56° до 57°, которые обеспечивают попадание мячика при скорости бросания $v_0 = 18$ м/с в стенку высотой $h = 1$ м, находящуюся на расстоянии $S = 30$ м (см. рис. 1.3).

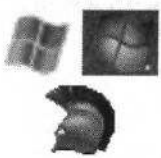
Проект «Диапазон углов» хранится в папке ..\IIKT11Prof\VB2005\Диапазон углов **Windows-CD**

Вопросы для размышления

1. От чего зависит точность вычисления значений переменных в языке программирования Visual Basic?
2. Имеет ли физический смысл вычисление значения координаты мячика с точностью 13 знаков после запятой? До какой точности целесообразно округлить полученное значение?

1.2.3. Компьютерная модель движения тела на языке Turbo Delphi

На основе формальной модели, описывающей движение тела, брошенного под углом к горизонту, создадим компьютерную модель с использованием системы программирования Turbo Delphi.



Проект «Бросание мячика в стенку» на языке Turbo Delphi

Создадим сначала графический интерфейс проекта.

1. Разместить на форме (рис. 1.4):
   - четыре текстовых поля для ввода значений: `EditV0` — начальной скорости, `EditA` — угла бросания мячика, `EditS` — расстояния до стенки и `EditL` — высоты стенки;
   - надпись `Label1` для вывода высоты мячика на заданном расстоянии;
   - надпись `Label2` для вывода текстового сообщения о результатах броска;
   - десять надписей для вывода имен переменных и единиц измерения;
   - кнопку `Button1` для запуска событийной процедуры вычисления результатов бросания мячика;
   - кнопку `Button2` для демонстрации траектории движения мячика.

2. Ввести в программный код в оператор **uses** модуль `Math`, который обеспечивает подключение математических функций (`Cos()`, `Tan()` и т. д.).

```
uses
Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms,  Dialogs, StdCtrls,
ExtCtrls, Math;
```

3. Объявить переменные (вещественные одинарной точности) и константы для всего проекта:

```
var      //начало раздела объявления переменных
V0: real; //начальная скорость
A:  real; //угол бросания
S:  real; //расстояние до стенки
H:  real; //высота стенки
```

```
const       //начало раздела объявления констант
G = 9.81;
Pi = 3.14;
```

4. Создать программный код событийной процедуры TForm1.Button1Click, определяющей попадание мячика в стенку. В этом коде:
   - объявить вещественную переменную одинарной точности L для данной событийной процедуры (высота мячика l в момент попадания в стенку);
   - присвоить переменным V0, A, S, H значения, введенные в текстовые поля, с использованием функции преобразования строки в вещественное число StrToFloat();
   - вычислить высоту мячика L на заданном расстоянии по формуле (1.2);
   - вывести высоту мячика L в поле надписи Label1;
   - вывести текстовое сообщение о результатах броска в поле надписи Label2 с использованием оператора **If-Then-Else**, в котором в качестве условия проверяется значение переменной L.

```
procedure TForm1.Button1Click(Sender: TObject);
var      //начало раздела объявления переменных
L: real; //высота мячика в момент попадания
          //в стенку
begin
V0:=StrToFloat(EditV0.Text);
A:=StrToFloat(EditA.Text);
S:=StrToFloat(EditS.Text);
H:=StrToFloat(EditL.Text);
              //Попадание в стенку
L:=S*Tan(A*Pi/180)-G*Sqr(S)/(2*Sqr(V0*
Cos(A*Pi/180)));
Label1.Caption:=FloatToStr(L);
if L<0 then
              Label2.Caption:='Недолет'
         else if L>H then
              Label2.Caption:='Перелет'
         else
Label2.Caption:='Попадание';
end;
```

Для визуализации формальной модели построим траекторию движения тела (график зависимости высоты мячика над поверхностью земли от дальности полета). Снабдим график осями координат со шкалами и выведем положение стенки.

5. Поместить на форму графическое поле Image1, в котором будет осуществляться построение графика.
   С помощью диалогового окна *Инспектор объектов* установить размеры графического поля, например свойству Height присвоить значение 200, а Width — 400.

6. Создать событийную процедуру TForm1.Button2Click, в которой:
   - объявить вещественную переменную T (время);
   - объявить целочисленные переменные X, Y и N (координаты мячика и счетчик цикла);
   - присвоить переменным V0, A, S, H значения, введенные в текстовые поля, с использованием функции преобразования строки в вещественное число StrToFloat();
   - построить траекторию движения мячика на объекте Image1.Canvas;
   - построить оси *X* и *Y* со шкалами и стенку.

Функция Round() необходима, чтобы преобразовывать вещественные переменные одинарной точности в целочисленные переменные (значения координат).

```
procedure TForm1.Button2Click(Sender: TObject);
var      //начало раздела объявления переменных
X: integer;   //координата X
Y: integer;   //координата Y
T: real;      //время
N: integer;   //счетчик
begin
//Ввод начальных значений
V0:=StrToFloat(EditV0.Text);
A:=StrToFloat(EditA.Text);
S:=StrToFloat(EditS.Text);
L:=StrToFloat(EditL.Text);
                    //рисование траектории
with Image1.Canvas do
begin
while T<5 Do
    begin
       T:=T+0.005;
       Y:=180-Round(10*(V0*Sin(A*Pi/180)*T-
            G*T*T/2));
       X:=5+Round(10*(V0*Cos(A*Pi/180)*T));
       Pixels[X,Y]:=clBlack;
    end;
```

```
   MoveTo(0,180); LineTo(400,180);      //ось X
   MoveTo(5,0); LineTo(5,400);          //ось Y
   MoveTo(Round(5+10*S),Round(180));    //стенка
   LineTo(Round(5+10*S),180-Round(10*H));

                        //шкала оси X
N:=0;
while N<400 do
   begin
     N:=N+50;
     MoveTo(5+N,180); LineTo(5+N,200);
     TextOut(7+N,180,IntToStr(Round(N/10)));
   end;
                        //шкала оси Y
N:=0;
while N<200 do
   begin
     N:=N+50;
     MoveTo(0,180-N); LineTo(10,180-N);
     TextOut(0,180-N,IntToStr(Round(N/10)));
   end;
end;
end;
end.
```

Проект хранится в папке ..\ИКТ11Prof\Delphi\Phys1 **Windows-CD**

Компьютерный эксперимент (см. рис. 1.4)

7. Запустить проект и ввести значения начальной скорости, угла бросания, расстояния до стенки и ее высоты. Щелкнуть по кнопке *Бросок*. В поля меток будут выведены значение координаты мячика и результат броска. Щелкнуть по кнопке *Траектория*. В графическом поле появится траектория движения тела.
Подобрать значения начальной скорости и угла бросания мячика, обеспечивающие его попадание в стенку.

Например, при скорости бросания мячика $v_0 = 18$ м/с и угле бросания $\alpha = 34°$ мячик попадет в стенку высотой $h = 1$ м, находящуюся на расстоянии $S = 30$ м на высоте $l = 0{,}406314279729352$ м.

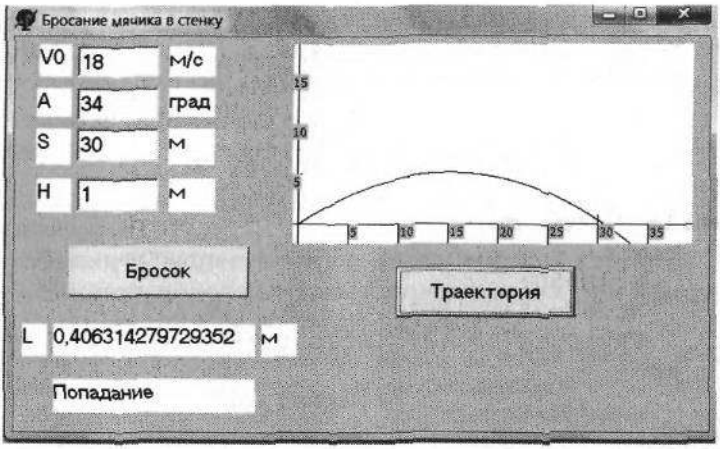
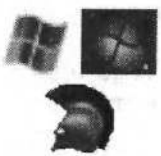


Рис. 1.4. Проект «Бросание мячика в стенку» на языке Turbo Delphi

Анализ результатов. Полученная точность высоты попадания мячика в стенку $l = 0{,}406314279729352$ м не имеет физического смысла и определяется типом переменной. Значение переменной типа `Real` вычисляется с точностью 15 значащих цифр, однако исходные данные заданы с точностью две значащие цифры, поэтому целесообразно результат округлить до трех значащих цифр: $l = 0{,}406$ м.

Корректировка модели. Модернизируем проект так, чтобы для каждого значения скорости бросания получить диапазон значений углов, обеспечивающий попадание мячика в стенку.

Для этого необходимо в цикле со счетчиком по углу бросания вычислить положение мячика на расстоянии стенки и определить те углы, которые соответствуют высоте стенки.



Проект «Диапазон углов» на языке Turbo Delphi

1. Поместить на форму (рис. 1.5):
   - три текстовых поля для ввода значений: `EditV0` — начальной скорости, `EditS` — расстояния до стенки и `EditH` — высоты стенки;

- надпись Label1 для вывода диапазона значений углов, при которых происходит попадание мячика в стенку;
- восемь надписей для обозначения переменных и единиц измерения;
- кнопку Button1 для запуска событийной процедуры.

2. Ввести в начало программного кода в оператор **uses** модуль Math, который обеспечивает подключение математических функций (Cos(), Tan() и т. д.).

```
uses
Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms,  Dialogs, StdCtrls,
ExtCtrls, Math;
```

3. Объявить переменные:

```
const    //начало раздела объявления констант
G=9.81;
Pi=3.14;
var      //начало раздела объявления переменных
V0: real;      //начальная скорость
A: integer;    //угол бросания
S: real;       //расстояние до стенки
H: real;       //высота стенки
L: real;//высота мячика на заданном расстоянии
```

4. Ввести программный код событийной процедуры:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
          //Ввод начальных значений
V0:=StrToFloat(EditV0.Text);
S:=StrToFloat(EditS.Text);
H:=StrToFloat(EditH.Text);
          //Попадание в стенку
for A:=0 to 90 do
begin
  L:=S*Tan(A*Pi/180)-G*Sqr(S)/(2*Sqr(V0*
     Cos(A*Pi/180)));
  if (0<L) And (L<H)
  then
      Label1.Caption := Label1.Caption+''+
                        IntToStr(A);
end;
end;
end.
```

5. Запустить проект и ввести скорость бросания мячика, расстояние до стенки и ее высоту.
Щелкнуть по кнопке *Диапазон углов*.

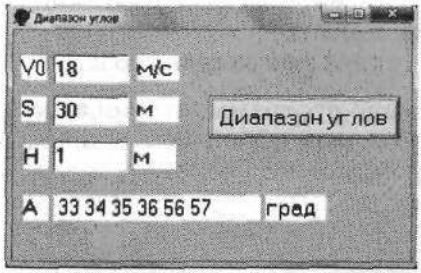


Рис. 1.5. Проект «Диапазон углов» на языке Turbo Delphi

Будет получен неочевидный результат, оказывается, существуют два диапазона углов: от 33° до 36° и от 56° до 57°, которые обеспечивают попадание мячика при скорости бросания $v_0 = 18$ м/с в стенку высотой $h = 1$ м, находящуюся на расстоянии $s = 30$ м (см. рис. 1.5).

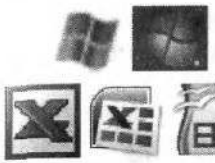
Проект хранится в папке ..\IIKT11Prof\Delphi\Phys2 **Windows-CD**

Вопросы для размышления

1. От чего зависит точность вычисления значений переменных в языке программирования Turbo Delphi?
2. Имеет ли физический смысл вычисление значения координаты мячика с точностью 13 знаков после запятой? До какой точности целесообразно округлить полученное значение?

1.2.4. Компьютерная модель движения тела в электронных таблицах

На основе формальной модели «Бросание мячика в стенку» создадим компьютерную модель с использованием электронных таблиц Microsoft Excel или OpenOffice.org Calc.



Проект «Бросание мячика в стенку» в электронных таблицах

Построение траектории движения мячика

Для ввода начальной скорости бросания мячика v_0 будем использовать ячейку B1, а для ввода угла бросания — ячейку B2 (рис. 1.6).

Введем в ячейки A5:A18 значения времени t с интервалом в 0,2 с и вычислим по формулам (1.1) значения координат тела x и y для заданных значений времени.

В электронных таблицах аргументы функций COS() и SIN() задаются в радианах, поэтому необходимо преобразовать значений углов из градусов в радианы с помощью функции РАДИАНЫ().

1. 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
2. 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 мячика по оси *Y*, вычисленные для определенных моментов времени (см. рис. 1.6).

	A	B	C
1	V_0 =	18,0	м/с
2	α =	35,0	град
3			
4	t	$x = v_0 \cdot \cos\alpha \cdot t$	$y = v_0 \cdot \sin\alpha \cdot t - g \cdot t^2/2$
5	0,0	0,0	0,0
6	0,2	2,9	1,9
7	0,4	5,9	3,3
8	0,6	8,8	4,4
9	0,8	11,8	5,1
10	1,0	14,7	5,4
11	1,2	17,7	5,3
12	1,4	20,6	4,8
13	1,6	23,6	4,0
14	1,8	26,5	2,7
15	2,0	29,5	1,0
16	2,2	32,4	-1,0
17	2,4	35,4	-3,5
18	2,6	38,3	-6,3

Рис. 1.6. 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. При построении графика в качестве категорий использовать диапазон ячеек B5:B18, а в качестве значений — диапазон ячеек C5:C18 (рис. 1.7).

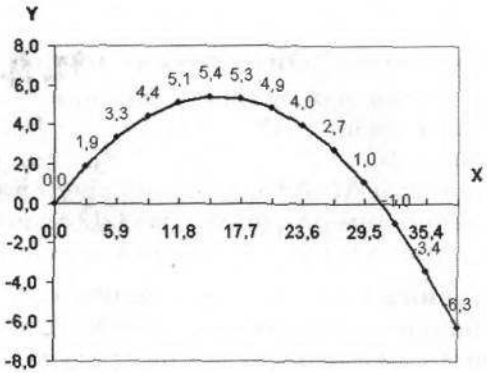


Рис. 1.7. Траектория движения мячика

3.3. Построение диаграмм и графиков

Информатика и ИКТ-9

Компьютерный эксперимент. Исследуем модель и определим с заданной точностью (например, 0,1°) диапазон углов бросания, которые обеспечивают попадание мячика в стенку.

В качестве начальных условий бросания мячика выберем, например, следующие: скорость бросания $v_0 = 18$ м/с, высота стенки $h = 1$ м, расстояние до стенки $s = 30$ м.

4. Ввести (рис. 1.8):
   - в ячейку B21 — значение *расстояния до стенки*;
   - в ячейку B22 — значение *начальной скорости*;
   - в ячейку B23 — значение угла;
   - в ячейку B25 — формулу для вычисления высоты мячика в момент попадания в стенку для заданных начальных условий:

21	S =	30,0	м
22	Vo =	18,0	м/с
23	α =	35,0	град
24			
25	L =	0,7	м

Рис. 1.8. Исходные данные для метода *Подбор параметра*

=B21*TAN(РАДИАНЫ(B23))–(9,81*B21^2)/(2*B22^2*COS(РАДИАНЫ(B23))^2)

Для заданных начальных условий (скорости бросания и расстояния до стенки) проведем поиск углов, которые дают попадание в стенку на высотах 0 и 1 м. Используем для этого метод *Подбор параметра*.

Надстройка *Подбор параметра* в электронных таблицах Microsoft Excel и OpenOffice.org Calc установлена по умолчанию.

Методом *Подбор параметра* будем сначала искать значение угла бросания, которое обеспечит попадание мячика в стенку на минимальной высоте 0 метров. В данном случае значение функции (высота мячика при попадании в стенку) хранится в ячейке B25 (см. рис. 1.8), а значение аргумента (угла бросания) — в ячейке B23. Значит, необходимо установить в ячейке B25 значение 0 и методом *Подбор параметра* найти соответствующее значение аргумента в ячейке B23.

5. Выделить ячейку B25, содержащую значение высоты мячика, и ввести команду [*Сервис-Подбор параметра...*].

6. В появившемся диалоговом окне (рис. 1.9) ввести в поле *Значение:* наименьшую высоту попадания в стенку (т. е. 0). В поле *Изменяя значение ячейки:* ввести адрес ячейки $B$23, содержащей значение угла бросания.
Щелкнуть по кнопке *ОК*.

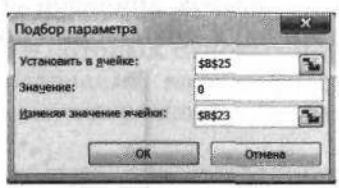


Рис. 1.9. Ввод данных для метода *Подбор параметра*

В ячейке B23 появится значение 32,6, т. е. минимальное значение угла бросания мячика, которое обеспечивает попадание в стенку при заданных начальных условиях.

21	S =	30,0	м
22	Vo =	18,0	м/с
23	α =	32,6	град
24			
25	L =	0,0	м

Рис. 1.10. Определение диапазона углов методом *Подбор параметра*

Методом *Подбор параметра* найдем теперь угол бросания, который обеспечит попадание мячика в стенку на максимальной высоте 1 метр.

7. Выделить ячейку B25, содержащую значение высоты мячика, и ввести команду [*Сервис-Подбор параметра...*].

8. В появившемся диалоговом окне ввести в поле *Значение:* наибольшую высоту попадания в стенку (т. е. 1).
9. В поле *Изменяя значение ячейки:* ввести адрес ячейки $B$23, содержащей значение угла бросания.
   Щелкнуть по кнопке ОК.
   В ячейке B23 появится значение 36,1, т. е. максимальное значение угла бросания мячика, которое обеспечивает попадание в стенку при заданных начальных условиях.

Анализ результатов. Таким образом, исследование компьютерной модели в электронных таблицах показало, что существует диапазон значений угла бросания мячика от 32,6° до 36,1°, при котором обеспечивается попадание в стенку высотой 1 м, находящуюся на расстоянии 30 м, мячиком, брошенным со скоростью 18 м/с.

Если повторить процедуру определения диапазона углов при начальном значении угла в ячейке B23, равном 55°, то получим значения предельных углов 55,8° и 57,4°, т. е. второй диапазон углов. Траектории движения мячика для двух диапазонов показаны на рис. 1.11.

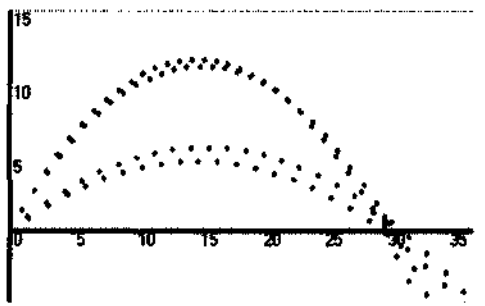


Рис. 1.11. Траектории мячика для двух диапазонов углов бросания

С учетом точности вычислений в электронных таблицах оба диапазона углов, обеспечивающие попадание в стенку при заданных начальных условиях, совпадают с результатами, полученными при исследовании этой компьютерной модели на языках программирования Visual Basic и Turbo Delphi.

Модель хранится в папке ..\IIKT11Prof в файле Model.xls на листе *Бросание мячика в стенку* **Windows-CD**

Вопросы для размышления

1. От чего зависит точность вычислений в электронных таблицах?
2. До какой точности целесообразно округлить полученные значения?

1.3. Приближенное решение уравнений

Алгебра-9

1.3.1. Графические и численные методы решения уравнений

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 $x^3 - \cos x = 0$ путем равносильных алгебраических преобразований. Однако такие уравнения можно решать приближенно графическими и численными методами.

Графические методы решения уравнений. 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 являются 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 или Turbo Delphi;

- в электронных таблицах Microsoft Excel или OpenOffice.org Calc путем построения диаграммы типа *График*.

Численные методы решения уравнений. Для решения уравнений с заданной точностью можно применить разработанные в вычислительной математике численные методы решения уравнений путем последовательных приближений. Самый простой из них — метод половинного деления. Если мы определим числовой отрезок аргумента x, на котором существует корень, и функция на краях этого отрезка принимает значения разных знаков, то можно использовать метод половинного деления.

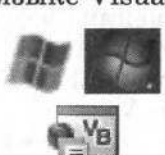
Вопросы для размышления

1. Для каких уравнений используются приближенные методы решения?
2. Почему целесообразно сначала использовать графический метод приближенного решения уравнения?

1.3.2. Приближенное решение уравнений на языке Visual Basic

Задача. Найти корень уравнения $x^3 - \cos x = 0$ приближенными методами (графическим и численным методом деления пополам числового отрезка аргумента).

Формальная модель задана уравнением, для нахождения корня уравнения разработаем компьютерную модель на языке Visual Basic.



Проект «Приближенное решение уравнения» на языке Visual Basic

Графический метод (рис. 1.12)

```
1. 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
'Графическое решение уравнения
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, 50)
Next X
For Y=0 To 200 Step 50
Graph1.DrawString((Y-150)/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, -50)
                              'Сдвиг по осям X и Y
'Рисование осей математической системы
'координат
Graph1.DrawLine(Pen1, -150, 0, 300, 0) 'Ось X
Graph1.DrawLine(Pen1, 0, -150, 0, 50)   '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.Cos(X)
Graph1.DrawEllipse(Pen1, X*100, Y*100, 1, 1)
Next X
End Sub
```

2. **График функции пересекает ось X один раз, следовательно, уравнение имеет один корень. По графику грубо приближенно можно определить, что $x \approx 0{,}8$ (см. рис. 1.12).**

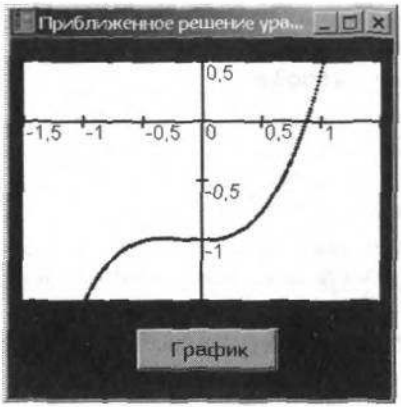


Рис. 1.12. Графическое решение уравнения

Численный метод половинного деления

Идея метода состоит в выборе точности решения и сведении первоначального числового отрезка $[A; B]$, на котором существует корень уравнения, к отрезку заданной точности. Процесс заключается в последовательном делении отрезков пополам точкой $C = (A + B)/2$ и отбрасыванию той половины отрезка ($[A; C]$ или $[C; B]$), на котором корня нет.

Выбор нужной половины отрезка основывается на проверке знаков значений функции на его краях. Выбирается та половина, на которой произведение значений функции на краях отрицательно, т. е. когда функция имеет разные знаки — пересекает ось абсцисс.

Процесс продолжается до тех пор, пока длина числового отрезка не станет меньше заданной удвоенной точности. Деление этого отрезка пополам дает значение корня с заданной точностью $x \approx (A + B)/2$.

3. Поместить на форму (рис. 1.13):
   - текстовые поля `TextBox1` и `TextBox2` для ввода числовых значений концов отрезка A и B;
   - текстовое поле `TextBox3` для ввода точности вычислений `P`;
   - надпись `Label1` для вывода значений корня;
   - четыре надписи для вывода обозначений;
   - кнопку `Button2` для запуска обработчика события.

4. Ввести программный код, позволяющий вычислить корень уравнения методом половинного деления с использованием цикла с постусловием, который будет выполняться, пока выполняется условие $(B - A)/2 > P$:

```
'Численное решение уравнения
Dim A, B, C, P As Single
Private Sub Button2_Click(…)
A=Val(TextBox1.Text)
B=Val(TextBox2.Text)
P=Val(TextBox3.Text)
Do
C=(A+B)/2
If (A^3-Math.Cos(A))*(C^3-Math.Cos(C))<0
Then
B=C
Else
A=C
End If
Loop While (B-A)/2>P
Label1.Text=(A+B)/2
End Sub
```

Из графика функции видно, что корень находится на отрезке [0,5; 1]. Введем в текстовые поля значения концов числового отрезка, а также точность вычислений (например, 0,0001).

На надпись будет выведено значения корня: $x \approx 0{,}8654175$ (см. рис. 1.13).

Рис. 1.13. Проект «Приближенное решение уравнения» на языке Visual Basic

Точность вычисления корня зависит не только от параметров используемого численного метода, но и от типа переменной. В нашем случае имеет смысл говорить о математической точности результата, которая не может превышать точность числового метода, т. е. $x \approx 0{,}8654$.

Проект хранится в папке ..\ИКТ11Prof\VB2005\Приближенное решение уравнения **Windows-CD**

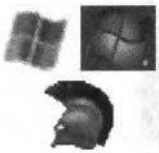
Вопросы для размышления

1. От чего зависит точность вычисления значений переменных в языке программирования Visual Basic?
2. До какой точности целесообразно округлить полученное значение корня уравнения?

1.3.3. Приближенное решение уравнений на языке Turbo Delphi

Задача. Найти корень уравнения $x^3 - \cos x = 0$ приближенными методами (графическим и численным методом деления пополам числового отрезка аргумента).

Формальная модель задана уравнением, для нахождения корня уравнения разработаем компьютерную модель на языке Turbo Delphi.



Проект «Приближенное решение уравнения» на языке Turbo Delphi

Графический метод (рис. 1.14)

1. Ввести в начало программного кода в оператор **uses** модуль `Math`, который обеспечивает подключение математических функций (`Cos()`, `Tan()` и т. д.).

```
uses
Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms,  Dialogs, StdCtrls,
ExtCtrls, Math;
```

2. **Ввести событийную процедуру** `TForm1.Button1Click (Sender: TObject)` **графического решения уравнения.**

```
var
X: real;
Y: real;
N: integer;
procedure TForm1.Button1Click(Sender: TObject);
begin
with Image1.Canvas do
begin
    //график функции
    X:=-3;
    while X<3 Do
    begin
    X:=X+0.001;
    Y:=X*X*X-Cos(X);
    Pixels[Round(100*X)+200,200-Round(20*Y)]:=
          clBlack;
    end;
    MoveTo(0,200); LineTo(500,200);   //Ось X
    MoveTo(250,0); LineTo(250,500);   //Ось Y
          //Шкала оси X
    N:=0;
    while N<500 do
    begin
    N:=N+100;
    MoveTo(N,190); LineTo(N,210);
    TextOut(N,200,FloatToStr(Round(N-250)/50));
    end;
          //Шкала оси Y
    N:=0;
    while N<400 do
    begin
    N:=N+100;
    MoveTo(245,400-N); LineTo(255,400-N);
    TextOut(245,400-N,FloatToStr
          (Round((N-200)/10)));
    end;
end;
end;
end.
```

3. График функции пересекает ось X один раз, следовательно, уравнение имеет один корень. По графику грубо приближенно можно определить, что $x \approx 0{,}8$ (см. рис. 1.14).

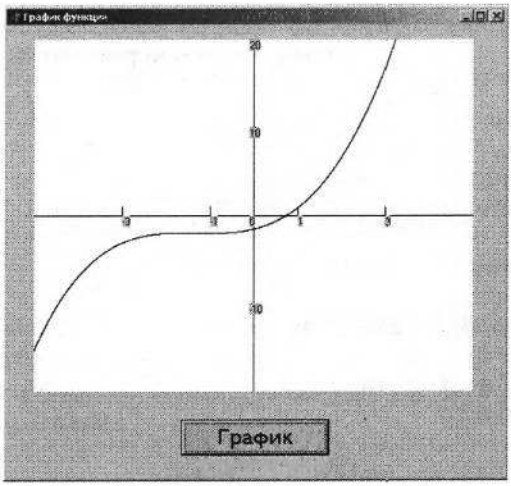


Рис. 1.14. Графическое решение уравнения

Численный метод половинного деления

4. Поместить на форму (рис. 1.15):
   - два текстовых поля `Edit1` и `Edit2` для ввода числовых значений концов числового отрезка A и B;
   - текстовое поле `Edit3` для ввода точности вычислений;
   - надпись `Label1` для вывода значений корня;
   - четыре надписи для вывода обозначений.

5. Поместить на форму кнопку `Button2` и создать событийную процедуру `TForm1.Button2Click()`. Ввести программный код, позволяющий вычислить корень уравнения методом половинного деления с использованием цикла с постусловием, который будет выполняться, пока не станет истинным условие $(B - A)/2 < E$:

```
var
A: real;
B: real;
C: real;
E: real;
```

```
procedure TForm1.Button2Click(Sender: TObject);
begin
A:=StrToFloat(Edit1.Text);
B:=StrToFloat(Edit2.Text);
E:=StrToFloat(Edit3.Text);
Repeat
C:=(A+B)/2;
If (A*A*A-Cos(A))*(C*C*C-Cos(C))<0
   Then B:=C
   Else A:=C
Until (B-A)/2<E;
Label1.Caption:=FloatToStr((A+B)/2);
end;
```

6. Из графика функции видно, что корень находится на отрезке [0; 1]. Введем в текстовые поля значения концов числового отрезка, а также точность вычислений (например, 0,0001).
На надпись будет выведено значения корня: $x \approx 0{,}86541$ (см. рис. 1.15).
Точность вычисления корня зависит не только от параметров используемого численного метода, но и от типа переменной. В нашем случае имеет смысл говорить о математической точности результата, которая не может превышать точность числового метода, т. е. $x \approx 0{,}8654$.

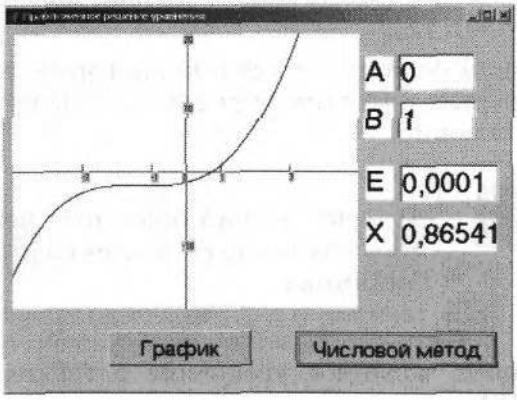


Рис. 1.15. Проект «Приближенное решение уравнения» на языке Turbo Delphi

Проект хранится в папке ..\IIKT11Prof\Delphi\Math1 **Windows-CD**

Вопросы для размышления

1. От чего зависит точность вычисления значений переменных в языке программирования Turbo Delphi?
2. До какой точности целесообразно округлить полученное значение корня уравнения?

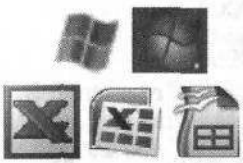
1.3.4. Приближенное решение уравнений в электронных таблицах

Возможности электронных таблиц не ограничиваются вычислениями по формулам и построением диаграмм и графиков.

С помощью метода *Подбор параметра* можно приближенно с заданной точностью решать уравнения.

Задача. Найти корень уравнения $x^3 - \cos x = 0$ приближенными методами (графическим и с помощью метода *Подбор параметра*).

Представим функцию в табличной форме, построим ее график, который позволит определить корень уравнения грубо приближенно.



Проект «Приближенное решение уравнения» в электронных таблицах

1. Представить заданное уравнение в табличной форме (*рис. 1.16*).

	A	B	C	D	E	F	G	H	I
1	x	0,000	0,200	0,400	0,600	0,800	1,000	1,200	1,400
2	$y = x^3 - \cos x$	-1,000	-0,972	-0,857	-0,609	-0,185	0,460	1,366	2,574

Рис. 1.16. Табличное представление уравнения

2. Для грубо приближенного определения корня построить диаграмму типа *График*.
   По графику грубо приближенно можно определить, что $x \approx 0{,}8$ (рис. 1.17).

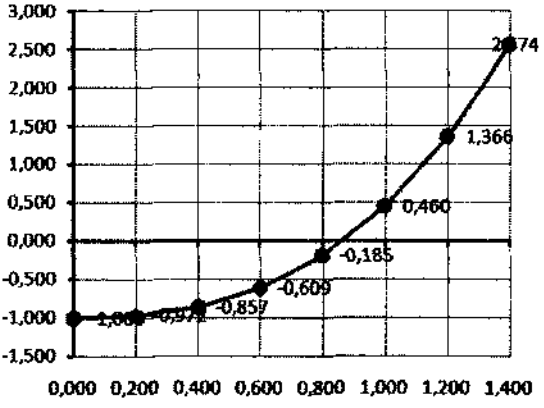


Рис. 1.17. График функции

Для поиска решения с заданной точностью используем метод *Подбор параметра*. Точность подбора зависит от заданной точности представления чисел в ячейках таблицы (например, до трех знаков после запятой).

Методом подбора параметра необходимо определить значение аргумента x (ячейка F1), при котором значение функции y (ячейка F2) будет равно нулю.

3. Выделить ячейку со значением функции F2 и ввести команду [*Сервис-Подбор параметра*...] в Microsoft Excel 2003 и OpenOffice.org Calc или команду [*Анализ "что-если"-Подбор параметра*...] в Microsoft Excel 2007.

4. В диалоговом окне *Подбор параметра* (рис. 1.18):
   - в поле *Установить в ячейке:* ввести адрес ячейки $F$2;
   - в поле *Значение:* ввести требуемое значение функции (в данном случае 0);
   - в поле *Изменяя значение ячейки:* ввести адрес ячейки $F$1, в которой будет производиться подбор значения аргумента.

   Щелкнуть по кнопке *ОК*.

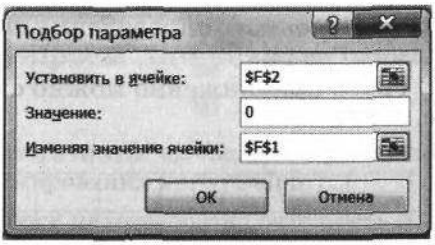


Рис. 1.18. Диалоговое окно надстройки *Подбор параметра*

5. На панели *Результат подбора параметра* будет выведена информация о величине значения в ячейке F2 (рис. 1.19).

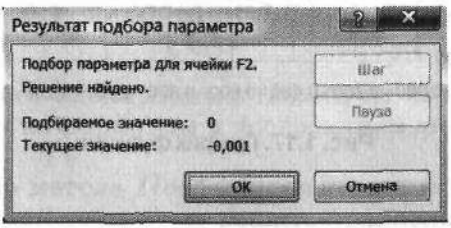


Рис. 1.19. Информационная панель надстройки *Подбор параметра*

6. В ячейке аргумента F1 появится подобранное значение 0,855. Таким образом, корень уравнения $x \approx 0{,}855$ найден с заданной точностью.

Модель хранится в папке ..\IIKT11Prof
в файле Model.xls **Windows-CD**
на листе Решение уравнения

Вопросы для размышления

1. От чего зависит точность вычислений в электронных таблицах?
2. До какой точности целесообразно округлить полученное значение корня уравнения?

1.4. Вероятностные модели

1.4.1. Построение информационной модели с использованием метода Монте-Карло

Вероятностные модели базируются на использовании больших серий испытаний со случайными параметрами, причем точность полученных результатов зависит от количества проведенных опытов.

Построим вероятностную модель, позволяющую приближенно вычислять площади геометрических фигур. Эта модель будет основана на методе Монте-Карло.

Описательная модель вычисления площадей геометрических фигур с использованием метода Монте-Карло. Сначала построим описательную вероятностную модель данного метода:

- поместим геометрическую фигуру полностью внутрь квадрата;
- будем случайным образом «бросать» точки в этот квадрат, т. е. с помощью генератора случайных чисел задавать координаты точек внутри квадрата;
- будем считать, что отношение числа точек, попавших внутрь фигуры, к общему числу точек, попавших в квадрат, приблизительно равно отношению площади фигуры к площади квадрата, причем это отношение тем точнее, чем больше количество точек.

Формальная модель «Определение площади круга методом Монте-Карло». Построим формальную модель для вычисления площади круга радиуса r, центр которого совпадает с началом координат.

Круг вписан в квадрат со стороной $2 \cdot r$, тогда площадь квадрата можно вычислить по формуле:

$$S_1 = 4 \cdot r^2.$$

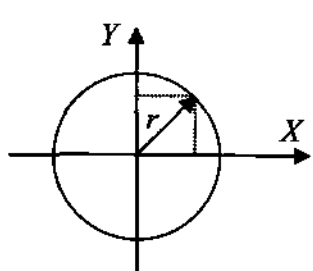


Рис. 1.20. Круг, вписанный в квадрат

Пусть N — количество точек, которые случайным образом генерируются внутри квадрата. Случайный выбор координат точек, которые попадают внутрь квадрата (N то-

чек), должен производиться так, чтобы координаты точек x и y удовлетворяли условиям:

$$-r \le x \le r \quad \text{и} \quad -r \le y \le r.$$

Пусть M — количество точек, попавших внутрь круга, их координаты удовлетворяют условию:

$$x^2 + y^2 \le r^2.$$

Предположим, что отношение площадей круга S_2 и квадрата S_1 равно отношению количества точек M, попавших внутрь круга, к количеству точек N, попавших внутрь квадрата, тогда получим формулу:

$$\frac{S_2}{S_1} = \frac{M}{N}.$$

Тогда площадь круга можно вычислить в единицах измерения радиуса по формуле:

$$S_2 = S_1 \cdot M/N = 4r^2 \cdot M/N.$$

Более того, таким способом можно вычислить значение числа π. Подставим в формулу значение площади круга $S_2 = \pi r^2$ и получим формулу для вычисления числа π:

$$\pi \cdot r^2 = 4r^2 \cdot M/N;$$
$$\pi = 4 \cdot M/N.$$

Вопросы для размышления

1. Почему с возрастанием количества случайных точек возрастает точность вычисления площади фигуры?
2. Можно ли с помощью метода Монте-Карло определить площадь треугольника? Площадь произвольной фигуры?

1.4.2. Компьютерные модели, построенные с использованием метода Монте-Карло, на языке Visual Basic

Разработаем на языке Visual Basic компьютерную модель, позволяющую методом Монте-Карло определять площадь круга и число π.

Проект «Метод Монте-Карло» на языке Visual Basic

1. Поместить на форму (рис. 1.21):
   - графическое поле PictureBox1, в котором будет отображаться процесс случайной генерации точек;
   - два текстовых поля: TextBox1 — для ввода количества генерируемых точек и TextBox2 — для ввода радиуса окружности;
   - две надписи для вывода значения площади круга и числа π;
   - кнопку Button1 для запуска обработчика события.
2. Установить размер графического поля: с помощью окна *Свойства* присвоить свойству Size значение 200;200.
3. Создать обработчик события, который обеспечивает:
   - стирание результатов предыдущего опыта;
   - ввод количества генерируемых точек и присваивание его переменной N;
   - ввод значения радиуса окружности и присваивание его переменной R;
   - генерацию случайных координат точек;
   - рисование в графическом поле квадрата со стороной 2*R и окружности радиуса R;
   - подсчет в переменной M количества точек, попавших внутрь круга;
   - вычисление и вывод значения площади круга и числа π на надписи.

```
Dim Graph1 As Graphics
Dim Pen1 As New Pen(Color.Black, 1)
Dim I, N, M, X, Y, R As Long
Private Sub Button1_Click(…)
Graph1=Me.PictureBox1.CreateGraphics()
'Стирание
Graph1.Clear(Color.White)
'Ввод данных
N=Val(TextBox1.Text)
R=Val(TextBox2.Text)
'Сдвиг по осям X и Y
Graph1.TranslateTransform(100, 100)
'Рисование круга и квадрата
Graph1.DrawEllipse(Pen2, -R, -R, 2*R, 2*R)
Graph1.DrawRectangle(Pen2, -R, -R, 2*R, 2*R)
```

```
For I=1 To N
'Генерация случайных координат точек и
рисование точек в графическом поле
X=Int(Rnd()*2*R)-R
Y=Int(Rnd()*2*R)-R
Graph1.DrawEllipse(Pen1, X, Y, 1, 1)
'Подсчет точек, попавших внутрь круга
If X^2+Y^2<=R^2 Then M=M+1
Next I
'Вывод площади круга и числа Pi
Label1.Text=4*(M/N)*(R^2)
Label2.Text=4*(M/N)
End Sub
```

4. **Ввести количество генерируемых точек и радиус окружности (от 1 до 100).**
   Щелкнуть по кнопке *Выполнить*, в графическом поле будет отображен процесс генерации случайных точек, а на надписи будут выведены значение площади круга и число π (см. рис. 1.21).

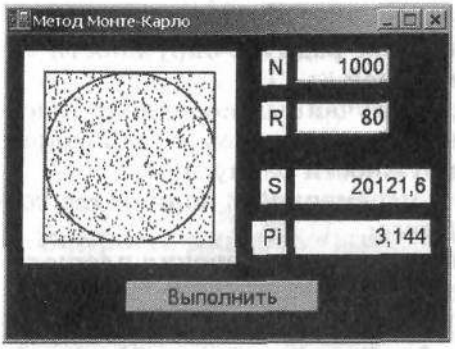
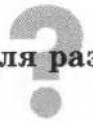


Рис. 1.21. Проект «Метод Монте-Карло» на языке Visual Basic

Компьютерный эксперимент. С помощью метода Монте-Карло можно определить с необходимой точностью значение числа π (при увеличении количества генерируемых точек будет увеличиваться точность вычисления числа π).

Проект хранится в папке ..\IIKT11Prof\Метод Монте-Карло **Windows-CD**

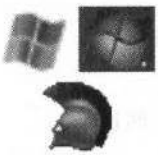


Вопросы для размышления

1. До какой точности целесообразно округлить полученное значение числа π?

1.4.3. Компьютерные модели, построенные с использованием метода Монте-Карло, на языке Turbo Delphi

Разработаем на языке Turbo Delphi компьютерную модель, позволяющую методом Монте-Карло определять площадь круга и число π.



Проект «Метод Монте-Карло» на языке Turbo Delphi

1. Поместить на форму (рис. 1.22):
   - графическое поле `Image1`, в котором будет отображаться процесс случайной генерации точек;
   - два текстовых поля: `Edit1` — для ввода радиуса окружности и `Edit2` — для ввода количества генерируемых точек;
   - надпись `Label1` для вывода значения площади круга;
   - надпись `Label2` для вывода числа π.
2. Установить размер графического поля: с помощью окна *Object Inspector* присвоить свойству `Size` значение `200;200`.
3. Ввести в начало программного кода в оператор **uses** модуль `Math`, который обеспечивает подключение математических функций (`Cos()`, `Tan()` и т. д.).

```
uses
Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms, Dialogs, StdCtrls,
ExtCtrls, Math;
```

4. Поместить на форму кнопку `Button1` и создать для нее событийную процедуру `TForm1.Button1Click()`, которая обеспечивает:

- стирание результатов предыдущего опыта;
- ввод значения радиуса окружности в переменную R;
- ввод количества генерируемых точек и присваивание его переменной N;
- генерацию случайных координат точек;
- рисование в графическом поле квадрата со стороной 2*R и окружности радиуса R;
- подсчет в переменной M количества точек, попавших внутрь круга;
- вычисление и вывод значений площади круга и числа π на надписи.

```
var
R: integer; //радиус
N: integer; //количество "брошенных" точек
X: real;    //координата точек X
Y: real;    //координата точек Y
M: integer; //количество точек, попавших
            //в круг
I: integer; //счетчик цикла
procedure TForm1.Button1Click(Sender: TObject);
begin
//Стирание
Image1.Canvas.Brush.Color := clWhite;
Image1.Canvas.Rectangle(100-R,100-R,100+R,100+R);
Image1.Canvas.FillRect(Rect(100-R,100-R,100+R,
                            100+R));
M:=0;
//Ввод значений
R:=StrToInt(Edit1.Text);
N:=StrToInt(Edit2.Text);
//Рисование квадрата и круга
Image1.Canvas.Rectangle(100-R,100-R,100+R,100+R);
Image1.Canvas.Ellipse(100-R,100-R,100+R,100+R);
//Генерация точек
Randomize;
For I:=1 To N do
begin
X:=Random(2*R)-R;
Y:=Random(2*R)-R;
Image1.Canvas.Pixels[Round(X)+100,
                     Round(Y)+100]:=clBlack;
If Sqr(X)+Sqr(Y)<=Sqr(R)
     Then M:=M+1
end;
```

```
//Площадь
Label1.Caption:=FloatToStr(4*Sqr(R)*M/N);
//Число PI
Label2.Caption:=FloatToStr(4*M/N);
end;
end.
```

5. Ввести радиус окружности и количество генерируемых точек. После щелчка по кнопке *Пуск* в графическом поле будет отображен процесс генерации случайных точек, а на надписи будут выведены значение площади круга и число π (см. рис.1.22).

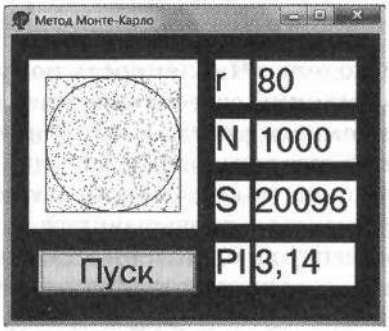


Рис. 1.22. Проект «Метод Монте-Карло» на языке Turbo Delphi

Компьютерный эксперимент. С помощью метода Монте-Карло можно определить с необходимой точностью значение числа π (при увеличении количества генерируемых точек будет увеличиваться точность вычисления числа π).

Проект хранится в папке ..\IIKT11Prof\Delphi\Monte-Karlo **Windows-CD**

Вопросы для размышления

1. До какой точности целесообразно округлить полученное значение числа π?

1.5. Биологические модели развития популяций

Общая биология 10–11 ☞

1.5.1. Информационные модели развития популяций

В биологии при исследовании развития биосистем строятся динамические модели изменения численности популяций различных живых существ (бактерий, рыб, зверей и т. д.) с учетом различных факторов. Взаимовлияние популяций рассматривается в моделях типа «жертва–хищник».

Формальная модель «Численность популяций». Изучение динамики популяций естественно начать с простейшей модели **неограниченного роста**, в которой численность популяции ежегодно увеличивается на определенный процент. Математическую модель можно записать с помощью рекуррентной формулы, связывающей численность популяции следующего года с численностью популяции текущего года, с использованием коэффициента роста a:

$$x_{n+1} = a \cdot x_n.$$

Например, если ежегодный прирост численности популяции составляет 5%, то $a = 1{,}05$.

В модели **ограниченного роста** учитывается эффект перенаселенности, связанный с нехваткой питания, болезнями и т. д., который замедляет рост популяции с увеличением ее численности. Введем коэффициент перенаселенности b, значение которого обычно существенно меньше a ($b \ll a$). Тогда коэффициент ежегодного увеличения численности равен $(a - b \cdot x_n)$, и формула принимает вид:

$$x_{n+1} = (a - b \cdot x_n) \cdot x_n .$$

В модели **ограниченного роста с отловом** учитывается, что на численность популяций промысловых животных оказывает влияние величина ежегодного отлова. Если величина ежегодного отлова равна c, то формула принимает вид:

$$x_{n+1} = (a - b \cdot x_n) \cdot x_n - c .$$

Популяции обычно существуют не изолированно, а во взаимодействии с другими популяциями. Наиболее важным типом является взаимодействие между жертвами и

хищниками (например, караси–щуки, зайцы–волки и т. д.). В модели **«жертва–хищник»** количество жертв x_n и количество хищников y_n связаны между собой. Количество встреч жертв с хищниками можно считать пропорциональным произведению собственно количеств жертв и хищников, а коэффициент f характеризует возможность гибели жертвы при встрече с хищниками. В этом случае численность популяции жертв уменьшается на величину $f \cdot x_n \cdot y_n$, и формула для расчета численности жертв принимает вид:

$$x_{n+1} = (a - b \cdot x_n) \cdot x_n - c - f \cdot x_n \cdot y_n .$$

Численность популяции хищников в отсутствие жертв (в связи с отсутствием пищи) уменьшается, что можно описать рекуррентной формулой

$$y_{n+1} = d \cdot y_n ,$$

где значение коэффициента $d < 1$ характеризует скорость уменьшения численности популяции хищников.

Увеличение популяции хищников можно считать пропорциональной произведению собственно количеств жертв и хищников, а коэффициент g характеризует величину роста численности хищников за счет жертв. Тогда для численности хищников можно использовать формулу:

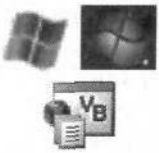
$$y_{n+1} = d \cdot y_n + g \cdot x_n \cdot y_n .$$

Вопросы для размышления

1. Приведите примеры изменения численности популяций, соответствующих моделям неограниченного роста, ограниченного роста, ограниченного роста с отловом и «жертва–хищник» в природе.

1.5.2. Компьютерные модели развития популяций на языке Visual Basic

Построим на языке Visual Basic компьютерную модель, позволяющую исследовать изменение со временем численности популяций с использованием различных моделей: неограниченного роста, ограниченного роста, ограниченного роста с отловом и «жертва–хищник».



Проект «Численность популяций» на языке Visual Basic

1. Поместить на форму (рис. 1.23) текстовые поля для ввода:
   - значений коэффициентов a, b, c и f, влияющих на изменение численности жертв: `TextBoxA`, `TextBoxB`, `TextBoxC` и `TextBoxF`;
   - значений коэффициентов d и g, влияющих на изменение численности хищников: `TextBoxD` и `TextBoxG`;
   - начальной численности популяций жертв и хищников: `TextBoxX` и `TextBoxY`;
   - количества рассматриваемых жизненных циклов (лет) `TextBoxN`.

2. Поместить на форму надписи для вывода численности популяций через заданное количество лет:
   - при неограниченном росте `Label1`;
   - при ограниченном росте `Label2`;
   - при ограниченном росте с отловом `Label3`;
   - в модели «жертва–хищник» `Label4` и `Label5`.

3. Поместить на форму графическое поле `PictureBox1` для рисования графиков зависимости численности популяций жертв и хищников от количества жизненных циклов (лет).
   С помощью диалогового окна *Свойства* присвоить свойству `Width` значение `320`, а свойству `Height` — значение `220`.

4. Поместить на форму надписи для вывода обозначений и поясняющих текстов.

5. Объявить переменные и графические инструменты:

```
Dim Graph1 As Graphics
Dim Pen1 As New Pen(Color.Black, 2)
Dim Pen2 As New Pen(Color.Green, 2)
Dim Pen3 As New Pen(Color.Blue, 2)
Dim Pen4 As New Pen(Color.Brown, 2)
Dim Pen5 As New Pen(Color.Red, 2)
Dim drawBrush As New SolidBrush(Color.Black)
Dim drawFont As New Font("Arial", 10)
Dim X, Y As Integer, N, I, A, B, C, D, G,
F, X1, X2, X3, X4, Y1 As Single
```

Для различных моделей динамики изменения численности популяций создадим событийные процедуры, где в цикле вычисляется численность популяций и результат (численность популяций) выводится на надписи.

6. Поместить на форму кнопку Button2 и создать событийную процедуру вычисления численности популяции в модели неограниченного роста:

```
'Неограниченный рост
Private Sub Button2_Click(…)
A=Val(TextBoxA.Text)
X1=Val(TextBoxX.Text)
N=Val(TextBoxN.Text)
For I=1 To N
X1=A*X1
Next I
Label1.Text=Int(X1)
End Sub
```

7. Поместить на форму кнопку Button3 и создать событийную процедуру вычисления численности популяции в модели ограниченного роста:

```
'Ограниченный рост
Private Sub Button3_Click(…)
A=Val(TextBoxA.Text)
B=Val(TextBoxB.Text)
X2=Val(TextBoxX.Text)
N=Val(TextBoxN.Text)
For I=1 To N
X2=(A-B*X2)*X2
Next I
Label2.Text=Int(X2)
End Sub
```

8. Поместить на форму кнопку Button4 и создать событийную процедуру вычисления численности популяции в модели ограниченного роста с отловом:

```
'Ограниченный рост с отловом
Private Sub Button4_Click(…)
A=Val(TextBoxA.Text)
B=Val(TextBoxB.Text)
C=Val(TextBoxC.Text)
X3=Val(TextBoxX.Text)
N=Val(TextBoxN.Text)
For I=1 To N
X3=(A-B*X3)*X3-C
Next I
```

```
Label3.Text=Int(X3)
End Sub
```

9. **Поместить на форму кнопку** `Button5` **и создать событийную процедуру вычисления численности популяции в модели «жертва–хищник»:**

```
'Жертва-хищник
Private Sub Button5_Click(…)
A=Val(TextBoxA.Text)
B=Val(TextBoxB.Text)
C=Val(TextBoxC.Text)
D=Val(TextBoxD.Text)
F=Val(TextBoxF.Text)
G=Val(TextBoxG.Text)
X4=Val(TextBoxX.Text)
Y1=Val(TextBoxY.Text)
N=Val(TextBoxN.Text)
For I=1 To N
X4=(A-B*X4)*X4-C-F*X4*Y1
Y1=D*Y1+G*X4*Y1
Next I
Label4.Text=Int(X4)
Label5.Text=Int(Y1)
End Sub
```

10. **Поместить на форму кнопку** `Button1` **и создать событийную процедуру построения графиков зависимости численности популяций от количества жизненных циклов (лет) для различных моделей:**

```
Private Sub Button1_Click(…)
Graph1=Me.PictureBox1.CreateGraphics()
Graph1.Clear(Color.White)
'Печать шкал системы координат
For X=20 To 320 Step 50
Graph1.DrawString((X-20), drawFont, drawBrush,
X-20, 200)
Next X
For Y=0 To 200 Step 40
Graph1.DrawString(Y, drawFont, drawBrush, 0,
200-Y)
Next Y
'Преобразование компьютерной системы координат
в используемую систему координат
Graph1.ScaleTransform(1, -1)     'Поворот оси Y
Graph1.TranslateTransform(20, -200)
                              'Сдвиг по осям X и Y
```

```
'Рисование осей с засечками
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, -5, X, 5)
Next X
For Y=0 To 200 Step 40   'Засечки на оси Y
Graph1.DrawLine(Pen1, -5, Y, 5, Y)
Next Y
'График неограниченного роста
X1=Val(TextBoxX.Text)
For I=1 To 100
Graph1.DrawEllipse(Pen1, I, X1, 1, 1)
X1=A*X1
Next I
'График ограниченного роста
X2=Val(TextBoxX.Text)
For I=1 To N
Graph1.DrawEllipse(Pen2, I, X2, 1, 1)
X2=(A-B*X2)*X2
Next I
'График ограниченного роста с отловом
X3=Val(TextBoxX.Text)
For I = 1 To N
Graph1.DrawEllipse(Pen3, I, X3, 1, 1)
X3=(A-B*X3)*X3-C
Next I
'Графики в модели жертва-хищник
X4=Val(TextBoxX.Text)
Y1=Val(TextBoxY.Text)
For I=1 To N
X4=(A-B*X4)*X4-C-F*X4*Y1
Y1=D*Y1+G*X4*Y1
Graph1.DrawEllipse(Pen4, I, X4, 1, 1)
Graph1.DrawEllipse(Pen5, I, Y1, 1, 1)
Next I
End Sub
```

11. **Запустить проект и ввести значения коэффициентов, начальное количество жертв и хищников и количество жизненных циклов (лет). Последовательно щелкнуть по кнопкам, на надписи будут выведены значения численности популяций, а в графическом поле будут построены графики, показывающие динамику развития популяций (см. рис. 1.23).**

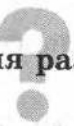
Рис. 1.23. Проект «Численность популяций» на языке Visual Basic

Анализ полученных результатов. Из графиков видно, что **модель неограниченного роста** приводит к катастрофическому возрастанию численности популяции. В природе существуют отдельные периоды, когда создаются наиболее благоприятные условия для жизни какого-либо вида, и тогда мы являемся свидетелями появления несметного количества комаров, саранчи и т. д.

В моделях **ограниченного роста** и **ограниченного роста с отловом** динамика изменения численности популяции практически одинакова при определенных значениях коэффициента отлова. Это означает, что могут существовать научно обоснованные нормы лова рыбы или охоты, которые практически не влияют на численность популяции с течением времени.

Исследование динамики численности популяций в модели **«жертва–хищник»** показывает, что численности популяций жертв и хищников тесно связаны. Так первоначальный рост численности жертв стимулирует резкий рост численности хищников, что с течением времени приводит к резкому уменьшению численности жертв и спустя некоторое время к уменьшению численности хищников и т. д.

Проект хранится в папке ..\IIKT11Prof\VB2005\Численность популяций **Windows-CD**

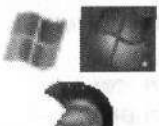


Вопросы для размышления

1. Почему численности популяций в моделях ограниченного роста и ограниченного роста с отловом при некоторых значениях параметров фактически совпадают?
2. Почему численности популяций в модели «жертва–хищник» фактически находятся в противофазе?

1.5.3. Компьютерные модели развития популяций на языке Turbo Delphi

Построим на языке Turbo Delphi компьютерную модель, позволяющую исследовать изменение со временем численности популяций с использованием различных моделей: неограниченного роста, ограниченного роста, ограниченного роста с отловом и «жертва–хищник».



Проект «Численность популяций» на языке Turbo Delphi

1. Поместить на форму (рис. 1.24) текстовые поля для ввода:
   - значений коэффициентов a, b, c и f, влияющих на изменение численности жертв: `EditA`, `EditB`, `EditC` и `EditF`;
   - значений коэффициентов d и g, влияющих на изменение численности хищников: `EditD` и `EditG`;
   - начальной численности популяций жертв и хищников: `EditX` и `EditY`;
   - количества рассматриваемых жизненных циклов (лет) `EditN`.
2. Поместить на форму надписи для вывода численности популяций через заданное количество лет:
   - при неограниченном росте `LabelNR`;
   - при ограниченном росте `LabelOR`;
   - при ограниченном росте с отловом `LabelORO`;
   - в модели «жертва–хищник» `LabelX_Y` и `LabelY_X`.

3. **Поместить на форму графическое поле** `Image1` **(например,** `300, 500`**), в котором будут строиться графики зависимости численности популяций от количества жизненных циклов (лет).**
4. **Поместить на форму надписи для вывода обозначений и поясняющих текстов.**
5. **Прежде всего, необходимо объявить переменные:**

```
var
A: real;    //коэффициент роста популяции
B: real;    //коэффициент уменьшения популяции
C: real;    //коэффициент отлова
D: real;    //коэффициент уменьшения численности
            //хищников в отсутствие жертв
G: real;    //коэффициент увеличения численности
            //хищников при наличии жертв
F: real;    //коэффициент уменьшения численности
            //жертв при наличии хищников
X: real;    //первоначальное количество жертв
Y: real;    //первоначальное количество хищников
N: integer; //количество циклов (лет)
I: integer; //счетчик цикла
```

6. **Поместить на форму кнопку** `Button1` **и начать создание событийной процедуры** `TForm1.Button1Click()`**. Присвоить переменным значения, вводимые в текстовые поля, с использованием функций преобразования типов данных** `StrToFloat()` **и** `StrToInt()`**:**

```
procedure TForm1.Button1Click(Sender: TObject);
begin
//Ввод данных
A:=StrToFloat(EditA.Text);
B:=StrToFloat(EditB.Text);
C:=StrToFloat(EditC.Text);
D:=StrToFloat(EditD.Text);
G:=StrToFloat(EditG.Text);
F:=StrToFloat(EditF.Text);
X:=StrToFloat(EditX.Text);
Y:=StrToFloat(EditY.Text);
N:=StrToInt(EditN.Text);
```

7. **В событийной процедуре установить ширину линий рисования на холсте, равную, например, 3 пикселям:**

```
//Установка ширины линии рисования
Image1.Canvas.Pen.Width:=3;
```

8. Ввести программный код модели **неограниченного роста**, где:
   - задается начальная точка графика с использованием метода `MoveTo()`;
   - задается цвет графика путем задания значения свойства `Color`;
   - в цикле вычисляется численность популяции и строится график с использованием метода `LineTo()`;
   - конечная численность населения выводится на надпись `Label1NR` с использованием функции преобразования типов данных `FloatToStr(X)`:

```
//Неограниченный рост
X:=StrToFloat(EditX.Text);
Image1.Canvas.MoveTo(0,250);
Image1.Canvas.Pen.Color:=clDkGray;
For I:=1 to N  Do
begin
Image1.Canvas.LineTo(25*I-25,250-Round(25*X)+25);
X:=A*X;
end;
Label1NR.Caption:=FloatToStr(X);
```

9. Ввести программный код модели ограниченного роста:

```
//Ограниченный рост
X:=StrToFloat(EditX.Text);
Image1.Canvas.MoveTo(0,250);
Image1.Canvas.Pen.Color:=clDkGray;
For I:=1 to N Do
begin
Image1.Canvas.LineTo(25*I-25,250-Round(25*X)+25);
X:=(A-B*X)*X;
end;
Label1OR.Caption:=FloatToStr(X);
```

10. Ввести программный код модели ограниченного роста с отловом:

```
//Ограниченный рост с отловом
X:=StrToFloat(EditX.Text);
Image1.Canvas.MoveTo(0,250);
Image1.Canvas.Pen.Color:=clBlue;
For I:=1 to N Do
begin
Image1.Canvas.LineTo(25*I-25,250-Round(25*X)+25);
```

```
X:=(A-B*X)*X-C;
Label1RO.Caption:=FloatToStr(X);
end;
Label1RO.Caption:=FloatToStr(X);
```

11. Ввести программный код модели «жертва–хищник» для вычисления численности жертв:

```
//Жертвы
X:=StrToFloat(EditX.Text);
Y:=StrToFloat(EditY.Text);
Image1.Canvas.MoveTo(0,250);
Image1.Canvas.Pen.Color:=clGreen;
For I:=1 to N Do
begin
Image1.Canvas.LineTo(25*I-25,250-Round(25*X)+25);
X:=(A-B*X)*X-C-F*X*Y;
Y:=D*Y+G*X*Y;
end;
Label1X_Y.Caption:=FloatToStr(X);
```

12. Ввести код модели «жертва–хищник» для вычисления численности хищников:

```
//Хищники
X:=StrToFloat(EditX.Text);
Y:=StrToFloat(EditY.Text);
Image1.Canvas.MoveTo(0,250);
Image1.Canvas.Pen.Color:=clRed;
For I:=1 to N Do
begin
Image1.Canvas.LineTo(25*I-25,250-Round(25*Y)+25);
X:=(A-B*X)*X-C-F*X*Y;
Y:=D*Y+G*X*Y;
end;
Label1Y_X.Caption:=FloatToStr(Y);
end;
end.
```

13. Запустить проект и ввести значения коэффициентов, начальное количество жертв и хищников и количество жизненных циклов (лет). (Для простоты примем начальные количества жертв и хищников за единицу.)
Щелкнуть по кнопке *Пуск*. Графики покажут динамику развития популяций, а на надписи будут выведены конечные значения численности популяций (см. рис. 1.24).

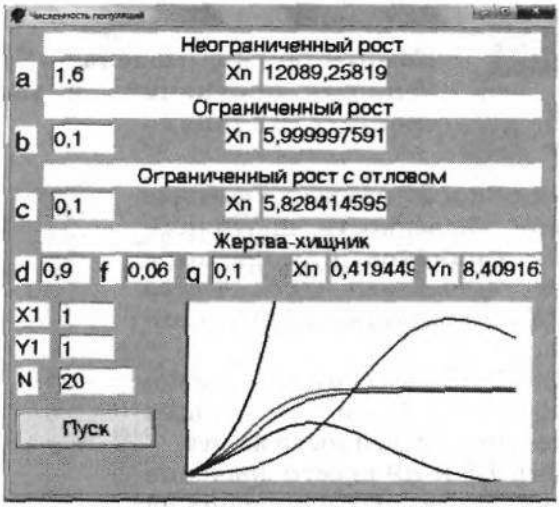


Рис. 1.24. Проект «Численность популяций» на языке Turbo Delphi

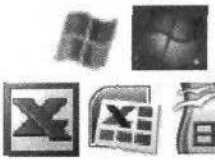
Проект хранится в папке ..\IIKT11Prof\Delphi\Biol **Windows-CD**

Вопросы для размышления

1. Почему численности популяций в моделях ограниченного роста и ограниченного роста с отловом при некоторых значениях параметров фактически совпадают?
2. Почему численности популяций в модели «жертва–хищник» фактически находятся в противофазе?

1.5.4. Компьютерные модели развития популяций в электронных таблицах

Построим в электронных таблицах компьютерную модель, позволяющую исследовать численность популяций с использованием различных моделей: неограниченного роста, ограниченного роста, ограниченного роста с отловом и «жертва–хищник».



Компьютерная модель «Численность популяций» в электронных таблицах

1. В ячейки B2 и B7 ввести начальные значения численности популяций жертв и хищников. Для простоты примем начальные количества жертв X1 и хищников Y1 за единицу (рис. 1.25).
   В ячейки B3:B6 ввести значения коэффициентов a, b, c и f, влияющих на изменение численности жертв.
   В ячейки B8 и B9 внести значения коэффициентов d и g, влияющих на изменение численности хищников.

	A	B
1		
2	X1=	1,00
3	a=	1,60
4	b=	0,05
5	c=	0,30
6	f=	0,06
7	Y1=	1,00
8	d=	0,90
9	g=	0,05

Рис. 1.25. Ввод начальных значений

В столбце D будем вычислять численность популяции в соответствии с моделью неограниченного роста, в столбце E — ограниченного роста, в столбце F — ограниченного роста с отловом, в столбцах G и H численность популяций жертв и хищников.

2. В ячейки D2, E2, F2, G2 и H2 внести значения начальной численности популяций.
   В ячейку D3 внести рекуррентную формулу неограниченного роста =$B$3*D2.
   В ячейку E3 внести рекуррентную формулу ограниченного роста =($B$3–$B$4*E2)*E2.
   В ячейку F3 внести рекуррентную формулу ограниченного роста с отловом =($B$3–$B$4*F2)*F2–$B$5.
   В ячейку G3 внести рекуррентную формулу роста жертв =($B$3–$B$4*G2)*G2–$B$5–$B$6*G2*H2.
   В ячейку H3 внести рекуррентную формулу роста хищников =$B$8*H2+$B$9*G2*H2.

3. Скопировать внесенные формулы в ячейки столбцов командами [*Правка-Заполнить-Вниз*].
   В ячейках столбцов ознакомиться с динамикой изменения численности популяций (рис. 1.26).

D	E	F	G	H
Неогр. рост	Огран. рост	С отловом	Жертвы	Хищники
1,0	1,0	1,0	1,0	1,0
1,6	1,6	1,3	1,2	1,0
2,6	2,4	1,6	1,5	0,9
4,1	3,5	2,2	1,9	0,9
6,6	5,0	2,9	2,4	0,9
10,5	6,7	4,0	3,1	0,9
16,8	8,5	5,2	4,0	0,9
26,8	10,0	6,7	5,1	1,0
42,9	11,0	8,2	6,3	1,2
68,7	11,5	9,5	7,3	1,5
110,0	11,8	10,4	8,1	1,9
175,9	11,9	10,9	8,5	2,4
281,5	12,0	11,2	8,4	3,2
450,4	12,0	11,3	8,0	4,2
720,6	12,0	11,4	7,3	5,5
1152,9	12,0	11,5	6,3	7,0
1844,7	12,0	11,5	5,2	8,4
2951,5	12,0	11,5	4,0	9,8
4722,4	12,0	11,5	3,0	10,8
7555,8	12,0	11,5	2,1	11,3
12089,3	12,0	11,5	1,4	11,3

Рис. 1.26. Динамика изменения численности популяций

Для визуализации компьютерной модели построим графики изменения популяций с течением времени.

4. Выделить столбцы данных D, E, F, G, H и построить диаграмму типа *График*.

   Появятся графики изменения численности популяций в соответствии с моделями неограниченного роста, ограниченного роста, ограниченного роста с отловом, количества жертв и хищников в модели «жертва–хищник» (рис. 1.27).

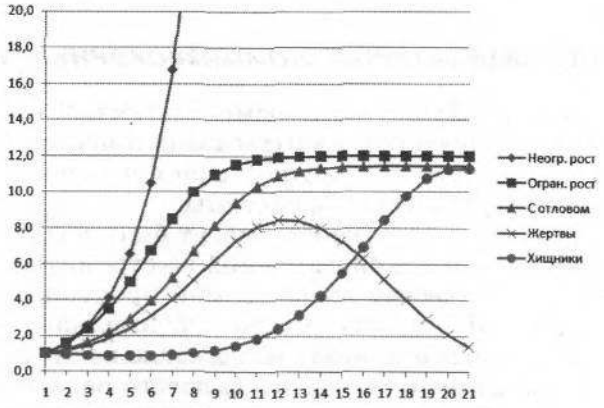
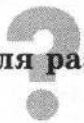


Рис. 1.27. Графики изменения численности популяций

Исследование модели. Изменяя значения начальной численности популяций, а также коэффициенты, можно получать различные варианты изменения численности популяций в зависимости от времени. При заданных значениях коэффициентов по графикам видно, что, начиная примерно с пятнадцатого жизненного цикла (года):

- отлов на численность популяции не влияет;
- количество хищников резко возрастает, а количество жертв уменьшается практически до нуля.

Модель хранится в папке ..\IIKT11Prof в файле Model.xls на листе Численность популяций **Windows-CD**



Вопросы для размышления

1. Почему численности популяций в моделях ограниченного роста и ограниченного роста с отловом при некоторых значениях параметров фактически совпадают?
2. Почему численности популяций в модели «жертва–хищник» фактически находятся в противофазе?

1.6. Оптимизационное моделирование в экономике

1.6.1. Информационные оптимизационные модели

В сфере управления сложными системами (например, в экономике) применяется оптимизационное моделирование, в процессе которого осуществляется поиск наиболее оптимального пути развития системы.

Критериями оптимальности могут быть различные параметры, например в экономике можно стремиться к максимальному количеству выпускаемой продукции, а можно — к ее низкой себестоимости. Оптимальное развитие соответствует экстремальному (максимальному или минимальному) значению выбранного целевого параметра.

Развитие сложных систем зависит от множества факторов (параметров), следовательно, целевой параметр зависит

от множества параметров. Выражением такой зависимости является целевая функция

$$K = F(X_1, X_2, ..., X_n),$$

где K — целевой параметр;
$X_1, X_2, ..., X_n$ — параметры, влияющие на развитие системы.

Цель исследования состоит в нахождении экстремума этой функции и определении значений параметров, при которых этот экстремум достигается. Если целевая функция нелинейная, то она имеет экстремумы, которые находятся определенными методами.

Однако часто целевая функция линейна и, соответственно, экстремумов не имеет. Задача поиска оптимального режима при линейной зависимости приобретает смысл только при наличии определенных ограничений на параметры. Если ограничения на параметры (система неравенств) также имеют линейный характер, то такие задачи являются задачами **линейного программирования**. (То есть термин «линейное программирование» в оптимизационном моделировании понимается как поиск экстремумов линейной функции, на которую наложены ограничения.)

Рассмотрим в качестве примера экономического моделирования поиск вариантов оптимального раскроя листов материала на заготовки определенного размера.

Содержательная постановка задачи. В ходе производственного процесса из листов материала получают заготовки деталей двух типов А и Б тремя различными способами, при этом количества получаемых заготовок при этих способах различны. В таблице 1.1 на пересечении строк и столбцов записаны количества заготовок типов А и Б при соответствующих способах раскроя.

Таблица 1.1. **Способы раскроя заготовок**

Тип заготовки	Способы раскроя		
	1-й	2-й	3-й
А	10	3	8
Б	3	6	4

Необходимо выбрать оптимальное сочетание способов раскроя, для того чтобы получить 500 заготовок типа А и 300 заготовок типа Б при расходовании наименьшего количества листов материала.

Формальная модель «Оптимизация раскроя». Параметрами, значения которых требуется определить, являются количества листов материала, которые будут раскроены различными способами:

X_1 — количество листов, раскроенное способом 1;

X_2 — количество листов, раскроенное способом 2;

X_3 — количество листов, раскроенное способом 3.

Целевая функция, выражающая количество листов материала, которое надо минимизировать, примет вид:

$$F = X_1 + X_2 + X_3 .$$

Ограничения накладываются требуемыми количествами заготовок типов А и Б, тогда с учетом количества заготовок, получаемых различными способами, должны выполняться два равенства:

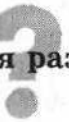
$$10 \cdot X_1 + 3 \cdot X_2 + 8 \cdot X_3 = 500;$$

$$3 \cdot X_1 + 6 \cdot X_2 + 4 \cdot X_3 = 300.$$

Кроме того, количества листов не могут быть отрицательными, поэтому должны выполняться неравенства:

$$X_1 \geq 0; \quad X_2 \geq 0; \quad X_3 \geq 0.$$

Таким образом, необходимо найти удовлетворяющие ограничениям значения параметров, при которых целевая функция принимает минимальное значение.



Вопросы для размышления

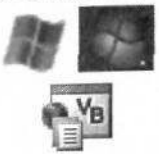
1. Почему для определения минимума или максимума целевой функции, линейно зависящей от параметров, необходимо наличие условий?

1.6.2. Построение и исследование оптимизационной модели на языке Visual Basic

Набор параметров X_1, X_2 и X_3 (количества листов материала, которые должны быть раскроены разными способами) должен удовлетворять одновременно двум условиям, что на языке Visual Basic запишется следующим образом:

```
10*X1+3*X2+8*X3=500 And 3*X1+6*X2+4*X3=300
```

Для того чтобы найти наборы значений параметров, удовлетворяющих этому условию, необходимо произвести перебор всех возможных вариантов с помощью трех вложенных циклов. С помощью оператора условного перехода надо вывести значения набора параметров и значение целевой функции на надписи.



Проект «Оптимизация раскроя» на языке Visual Basic

1. Поместить на форму (рис. 1.28):
   - три надписи Label1, Label2 и Label3 для вывода значений параметров;
   - надпись Label4 для вывода значения целевой функции;
   - надписи для вывода обозначений;
   - кнопку Button1 для запуска обработчика события.
2. Создать обработчик события:

```
Dim X1, X2, X3, F As Byte
Private Sub Button1_Click(…)
F=300
For X1=0 To 100
For X2=0 To 100
For X3=0 To 100
If 10*X1+3*X2+8*X3=500 And 3*X1+6*X2+4*X3=300
Then
If X1+X2+X3<F
Then F=X1+X2+X3
Label1.Text=X1
Label2.Text=X2
Label3.Text=X3
Label4.Text=F
End If
End If
Next X3
Next X2
Next X1
End Sub
```

3. Запустить проект и щелкнуть по кнопке *Оптимизировать*.
   На надписи будет выведен набор параметров (см. рис. 1.28):

X_1 (количество листов, раскроенное способом 1) — 20;
X_2 (количество листов, раскроенное способом 2) — 20;
X_3 (количество листов, раскроенное способом 3) — 30.
Значение целевой функции (количество листов материала) — 70.

Рис. 1.28. Проект «Оптимизация раскроя» на языке Visual Basic

Проект хранится в папке ..\IIKT11Prof\VB2005\Оптимизация раскроя **Windows-CD**

Вопросы для размышления

1. Почему процесс оптимизации требует перебора всех вариантов значений параметров?

1.6.3. Построение и исследование оптимизационной модели на языке Turbo Delphi

Набор параметров X_1, X_2 и X_3 (количества листов материала, которые должны быть раскроены различными способами) должен удовлетворять одновременно двум условиям, что на языке Turbo Delphi запишется следующим образом:

```
(10*X1+3*X2+8*X3=500) And (3*X1+6*X2+4*X3=300)
```

Для того чтобы найти наборы значений параметров, удовлетворяющих этому условию, необходимо произвести перебор всех возможных вариантов с помощью трех вложенных циклов. С помощью оператора условного перехода надо вывести значения набора параметров и значение целевой функции на форму.

Проект «Оптимизация раскроя» на языке Turbo Delphi

1. Поместить на форму (рис. 1.29):
   - три надписи Label1, Label2 и Label3 для вывода значений параметров;
   - надпись Label4 для вывода значения целевой функции;
   - надписи для вывода обозначений;
   - кнопку Button1 для запуска событийной процедуры.
2. Создать событийную процедуру TForm1.Button1Click():

```
var
X1: integer;
X2: integer;
X3: integer;
F:  integer;
procedure TForm1.Button1Click(Sender: TObject);
begin
F=300;
For X1:=0 To 100 Do
begin
For X2:=0 To 100 Do
begin
For X3:=0 To 100 Do
begin
If (10*X1+3*X2+8*X3=500) And (3*X1+6*X2+4*X3=300)
   Then begin
     If X1+X2+X3<F
          Then begin
            F:=X1+X2+X3;
            Label1.Caption:=IntToStr(X1);
            Label2.Caption:=IntToStr(X2);
            Label3.Caption:=IntToStr(X3);
            Label4.Caption:=IntToStr(F);
          end;
   end;
end;
end;
end;
end;
```

3. Запустить проект и щелкнуть по кнопке *Пуск*.
   На надписи будет выведен набор параметров (см. рис. 1.29):

X_1 (количество листов, раскроенное способом 1) — 20;
X_2 (количество листов, раскроенное способом 2) — 20;
X_3 (количество листов, раскроенное способом 3) — 30.
Значение целевой функции (количество листов материала) — 70.

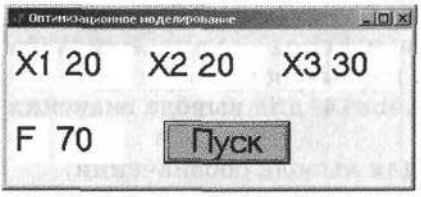


Рис. 1.29. Проект «Оптимизация раскроя» на языке Turbo Delphi

Проект «Оптимизация раскроя» хранится в папке ..\IIKT11Prof\Delphi\Optim **Windows-CD**

Вопросы для размышления

1. Почему процесс оптимизации требует перебора всех вариантов значений параметров?

1.6.4. Построение и исследование оптимизационной модели в электронных таблицах

Надстройка *Поиск решения*. Возможности электронных таблиц Microsoft Excel не ограничиваются вычислениями по формулам и построением диаграмм и графиков. Задачи оптимизационного моделирования можно решать с помощью надстройки электронных таблиц *Поиск решения*, которая требует дополнительной установки.

Процедура поиска решения позволяет найти оптимальное значение формулы, содержащейся в ячейке, которая называется **целевой**. Эта процедура работает с группой ячеек, прямо или косвенно связанных с формулой в целевой ячейке. Чтобы получить по формуле, содержащейся в целевой ячейке, заданный результат, процедура изменяет значения во влияющих ячейках. Чтобы сузить множество значений, используемых в модели, применяются ограничения. Эти ограничения могут ссылаться на другие влияющие ячейки.

Компьютерная модель «Оптимизация раскроя» в электронных таблицах

Установим надстройку *Поиск решения* в Microsoft Excel 2003.

1. Ввести команду [*Сервис- Надстройки*].
   В диалоговом окне *Надстройки* в списке *Доступные надстройки:* выбрать настройку *Поиск решения* путем установки флажка (рис. 1.30).
   Щелкнуть по кнопке *ОК*.

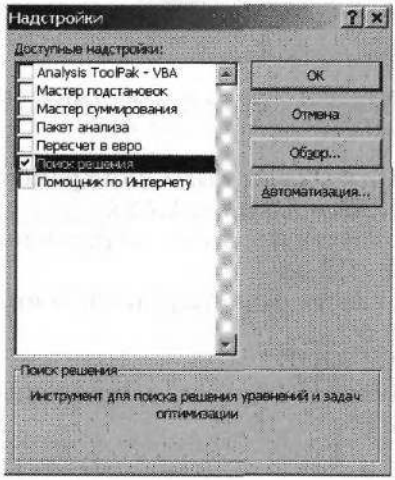


Рис. 1.30. Диалоговое окно *Надстройки* в Microsoft Excel 2003

Установим надстройку *Поиск решения* в Microsoft Excel 2007.

1. Щелкнуть по значку *Кнопка Microsoft Office*  .
   В появившемся диалоговом окне щелкнуть по кнопке *Параметры Excel*.
   В появившемся диалоговом окне выбрать пункт *Надстройки*.
   В списке *Управление* выбрать пункт *Надстройки Excel* и нажать кнопку *Перейти*.
   В диалоговом окне *Надстройки* в списке *Доступные надстройки:* выбрать настройку *Поиск решения* путем установки флажка (рис. 1.31).
   Щелкнуть по кнопке *ОК*.

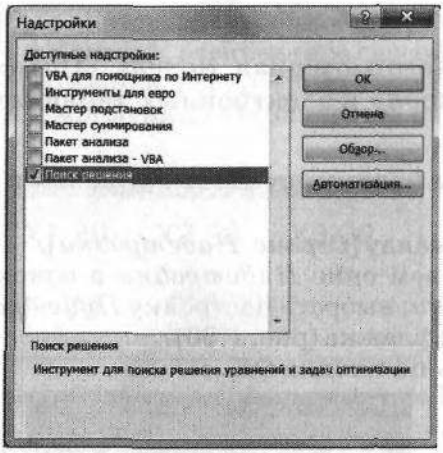


Рис. 1.31. Диалоговое окно *Надстройки* в Microsoft Excel 2007

2. Ячейки B2, C2 и D2 выделить для хранения значений параметров X1, X2 и X3 (рис. 1.32).
В ячейку B4 ввести формулу вычисления целевой функции: =B2+C2+D2.
В ячейку B7 ввести формулу вычисления количества заготовок типа А:
=10*B2+ 3*C2 + 8*D2.
В ячейку B8 ввести формулу вычисления количества заготовок типа Б:
=3*B2+ 6*C2 + 4*D2.

	A	B	C	D
1		X1	X2	X3
2	Параметры:	0	0	0
3				
4	Целевая функция:	0		
5				
6	Ограничения			
7	Кол-во заготовок А:	0		
8	Кол-во заготовок Б:	0		

Рис. 1.32. Ввод формул оптимизационной модели

Для поиска оптимального набора значений параметров, который соответствует минимальному значению целевой функции, воспользуемся надстройкой электронных таблиц *Поиск решения*.

3. В электронных таблицах Microsoft Excel 2003 ввести команду [*Сервис-Поиск решения...*].
В электронных таблицах Microsoft Excel 2007 ввести команду [*Данные-Поиск решения...*].
В появившемся диалоговом окне *Поиск решения* (рис. 1.33) установить:
   - адрес целевой ячейки;
   - вариант оптимизации значения целевой ячейки (в нашем случае минимизацию);
   - адреса ячеек, значения которых изменяются в процессе поиска решения (в которых хранятся значения параметров);
   - ограничения (типа *равно* для ячеек, хранящих количество заготовок, и типа *больше равно* для параметров).

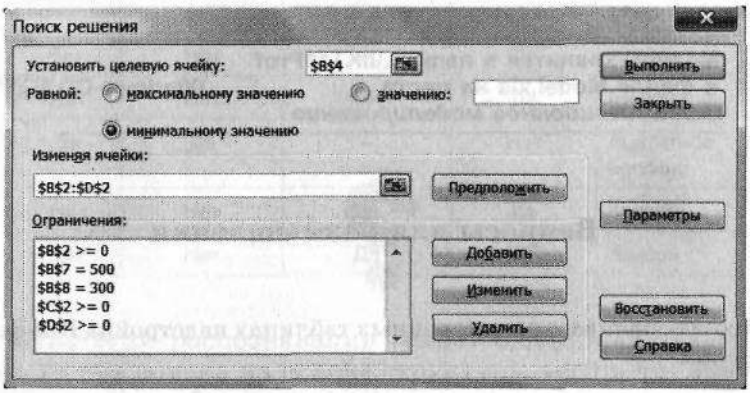


Рис. 1.33. Диалоговое окно *Поиск решения*

4. Щелкнуть по кнопке *Выполнить*.
В диалоговом окне *Текущее состояние поиска решения* (рис. 1.34) нажимать кнопку *Продолжить* до тех пор, пока в ячейке целевой функции не появится минимальное значение (рис. 1.35).

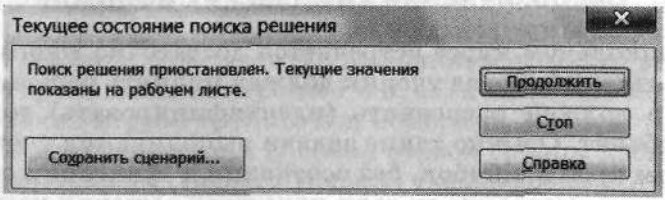


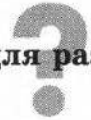
Рис. 1.34. Процесс поиска решения

5. Таким образом, для изготовления 500 деталей А и 300 деталей Б требуется 70 листов материала, при этом 20 листов необходимо раскроить по первому, 20 листов — по второму и 30 листов — по третьему варианту (см. рис. 1.35).

	A	B	C	D
1		X1	X2	X3
2	Параметры:	20	20	30
3				
4	Целевая функция:	70		
5				
6	Ограничения			
7	Кол-во заготовок А:	500		
8	Кол-во заготовок Б:	300		

Рис. 1.35. Результат оптимизации

Модель хранится в папке ..\IIKT11Prof в файле Model.xls на листе *Оптимизационное моделирование* **Windows-CD**



Вопросы для размышления

1. Что обеспечивает в электронных таблицах надстройка *Поиск решения*?

1.7. Модели распознавания химических волокон

1.7.1. Построение информационной модели распознавания химических волокон

В школьном курсе встречается достаточно много учебных ситуаций, когда ученик должен принять решение, например должен распознать (идентифицировать) тот или иной объект. Обычно такие задачи выполняются учеником методом проб и ошибок, без осознания и фиксации стратегии поиска. Создание модели принятия решения как раз и является осознанием и фиксацией последовательности рас-

суждений (действий), которая приводит к распознаванию того или иного объекта среди некоторой совокупности.

В качестве примера такой модели рассмотрим лабораторную работу по химии «Распознавание волокон». Учащемуся даются образцы волокон, газовая горелка и справочная таблица по свойствам горения каждого образца волокон (табл. 1.2). Предлагается распознать каждое из волокон.

Органическая химия-10–11

Таблица 1.2. **Свойства волокон**

№	Горит быстро	Растворяется в ацетоне	Из расплава можно вытянуть нити	При горении ощущается запах	Волокно (результат распознавания)
1	Да	Нет	–	Да	Хлопок
2	Нет	Нет	Нет	Да	Шерсть
3	Да	Да	–	Нет	Ацетатное волокно
4	Нет	Нет	Да	Да	Капрон
5	Нет	Нет	Да	Нет	Лавсан

Стратегия распознавания может быть представлена в виде дерева поиска на основе структуры «если–то–иначе», причем может быть множество различных деревьев с различным количеством шагов. Наша цель — нахождение оптимальной стратегии распознавания (достижения цели за минимальное число шагов). Такая стратегия будет реализована, если каждый шаг будет максимально уменьшать неопределенность (нести максимальное количество информации).

Формальная модель «Распознавание волокон». На первом шаге разделим пять волокон на две группы по условию *Горит быстро*, если условие выполняется, то это волокна первой группы под номерами 1 и 3, если не выполняется, то это волокна второй группы под номерами 2, 4 и 5.

Для идентификации волокон первой группы достаточно проверить справедливость условия *Растворяется в ацетоне*. Если условие выполняется, то это волокно 3 — ацетатное волокно, если не выполняется, то это волокно 1 — хлопок.

Для идентификации волокон второй группы сначала необходимо проверить справедливость условия *Из расплава можно вытянуть нити*. Если условие выполняется, то это волокна 4 и 5, если не выполняется, то это волокно 2 — шерсть.

Для идентификации волокон 4 и 5 достаточно проверить справедливость условия *При горении ощущается запах*. Если условие выполняется, то это волокно 4 — капрон, если не выполняется, то это волокно 5 — лавсан.

Целесообразно представить иерархическую модель распознавания (принятия решения) в виде блок-схемы (рис. 1.36).

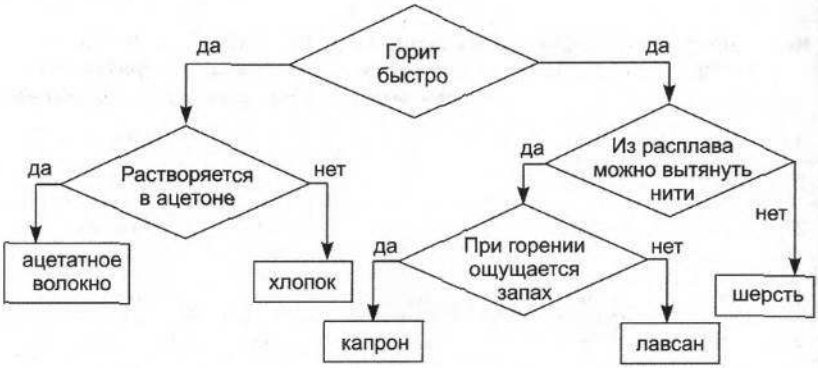


Рис. 1.36. Блок-схема модели «Распознавание волокон»

Вопросы для размышления

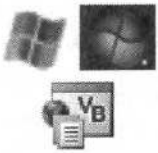
1. Могут ли существовать несколько моделей распознавания для одной и той же ситуации?
2. Какая стратегия построения модели распознавания быстрее всего приведет к цели?

1.7.2. Модель распознавания химических волокон на языке Visual Basic

Реализуем модель распознавания волокон с использованием языка Visual Basic. В проекте пользователю задается серия вопросов, анализируются ответы и сравниваются с

имеющимися фактами. При этом производится логический вывод и формируется ответ на интересующий пользователя вопрос, т. е. определяется название волокна.

Результаты распознавания (названия волокон) будем помещать в управляющий элемент ListBox1 (*поле списка*), который удобен для ввода элементов списка с помощью метода ListBox1.Items.Add().



Проект «Распознавание волокон» на языке Visual Basic

1. Поместить на форму (рис. 1.37) кнопку Button1 и список ListBox1.

Первую развилку (условие *Горит быстро*) реализуем в форме обработчика события, а остальные — в форме общих процедур:

- для идентификации волокон первой группы (1-го и 3-го) создадим процедуру Раствор() (условие *Растворяется в ацетоне*);
- для идентификации волокон второй группы сначала необходимо создать процедуру Нити() (условие *Из расплава можно вытянуть нити*);
- для идентификации 4-го и 5-го волокон надо создать процедуру Запах() (условие *При горении ощущается запах*).

2. Создать обработчик события Button1_Click(), который содержит вызовы общих процедур Раствор() и Нити():

```
Dim A As Byte
Private Sub Button1_Click(…)
A=MsgBox("Горит быстро?", 36, "Первый вопрос")
If A=6 Then Раствор() Else Нити()
End Sub
```

3. Создать общую процедуру Раствор(), которая позволяет распознать 1-е и 3-е волокна:

```
Sub Раствор()
A=MsgBox("Растворяется в ацетоне?", 36, "Второй
вопрос")
If A=6 Then ListBox1.Items.Add("3.Ацетатное
волокно") Else ListBox1.Items.Add("1.Хлопок")
End Sub
```

4. Создать общую процедуру Нити(), которая позволяет распознать 2-е волокно и содержит вызов общей процедуры Запах():

```
Sub Нити()
A=MsgBox("Из расплава можно вытянуть нити?",
36, "Второй вопрос")
If A=6 Then Запах() Else
ListBox1.Items.Add("2.Шерсть")
End Sub
```

5. Создать общую процедуру Запах(), которая позволяет распознать 4-е и 5-е волокна:

```
Sub Запах()
A=MsgBox("При горении ощущается запах?", 36,
"Третий вопрос")
If A=6 Then ListBox1.Items.Add("4.Капрон") Else
ListBox1.Items.Add("5.Лавсан")
End Sub
```

Компьютерный эксперимент (см. рис. 1.37). Работа с моделью позволит более эффективно спланировать и провести распознавание волокон в процессе выполнения лабораторной работы по химии.

6. Запустить проект и проводить химические опыты в соответствии с задаваемыми вопросами.
Выполнить процедуру распознавания для каждого волокна.

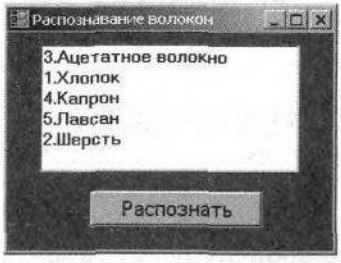


Рис. 1.37. Проект «Распознавание волокон» на языке Visual Basic

Проект хранится в папке ..\IIKT11Prof\VB2005\Распознавание волокон **Windows-CD**

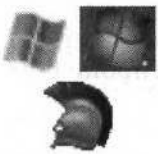
Вопросы для размышления

1. Может ли процесс распознавания химических волокон проводиться в произвольном порядке?

1.7.3. Модель распознавания химических волокон на языке Turbo Delphi

Реализуем модель распознавания волокон с использованием языка Turbo Delphi. В проекте пользователю задается серия вопросов, анализируются ответы и сравниваются с имеющимися фактами. При этом производится логический вывод и формируется ответ на интересующий пользователя вопрос, т. е. определяется название волокна.

Результаты распознавания (названия волокон) будем помещать в управляющий элемент `ListBox` (*список*), который удобен для ввода элементов списка с помощью метода `Items.Add()`.



Проект «Распознавание волокон» на языке Turbo Delphi

1. Поместить на форму (рис. 1.38) кнопку `Button1` и список `ListBox1`.

Первую развилку (условие *Горит быстро*) реализуем в форме событийной процедуры, а остальные — в форме общих процедур:

- для идентификации волокон первой группы (1-го и 3-го) создадим процедуру `Rastvor` (условие *Растворяется в ацетоне*);
- для идентификации волокон второй группы сначала необходимо создать процедуру `Niti` (условие *Из расплава можно вытянуть нити*);
- для идентификации 4-го и 5-го волокон надо создать процедуру `Zapax` (условие *При горении ощущается запах*).

2. Определить переменные и процедуры. Создать событийную процедуру `TForm1.Button1Click()`, которая содержит вызовы общих процедур `Rastvor` и `Niti`:

```
var
A:   integer;
Form1: TForm1;
procedure Rastvor;
procedure Niti;
procedure Zapax;
procedure TForm1.Button1Click(Sender: TObject);
begin
A:=MessageDlg('Горит быстро?', MtConfirmation,
[mbYes,mbNo],0);
if A=idYes then Rastvor else Niti;
end;
```

3. Создать общую процедуру Rastvor, которая позволяет распознать 1-е и 3-е волокна:

```
procedure Rastvor;
begin
A:=MessageDlg('Растворяется в ацетоне?',
MtConfirmation, [mbYes,mbNo],0);
If A=idYes Then
Form1.ListBox1.Items.Add('3.Ацетатное волокно')
Else Form1.ListBox1.Items.Add('1.Хлопок');
end;
```

4. Создать общую процедуру Niti, которая позволяет распознать 2-е волокно и содержит вызов общей процедуры Zapax:

```
procedure Niti;
begin
A:=MessageDlg('Из расплава можно вытянуть
нити?', MtConfirmation, [mbYes,mbNo],0);
If A=idYes Then Zapax Else
Form1.ListBox1.Items.Add('2.Шерсть');
end;
```

5. Создать общую процедуру Zapax, которая позволяет распознать 4-е и 5-е волокна:

```
procedure Zapax;
begin
A:=MessageDlg('При горении ощущается запах?',
MtConfirmation, [mbYes,mbNo],0);
If A=idYes Then
Form1.ListBox1.Items.Add('4.Капрон') Else
Form1.ListBox1.Items.Add('5.Лавсан');
end;
end.
```

Компьютерный эксперимент (см. рис. 1.38). Работа с моделью позволит более эффективно спланировать и провести распознавание волокон в процессе выполнения лабораторной работы по химии.

6. Запустить проект и проводить химические опыты в соответствии с задаваемыми вопросами.
   Проделать процедуру распознавания для каждого волокна.

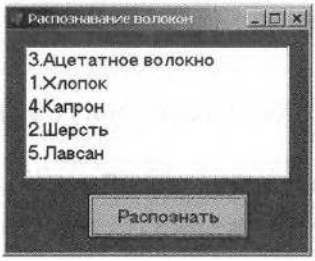


Рис. 1.38. Проект «Распознавание волокон» на языке Turbo Delphi

Проект хранится в папке ..\IIKT11Prof\Delphi **Windows-CD**

Вопросы для размышления

1. Может ли процесс распознавания химических волокон проводиться в произвольном порядке?

1.8. Модели логических устройств

1.8.1. Логические схемы полусумматора и триггера

При изучении базовых логических устройств компьютера (сумматор, триггер) целесообразно использовать компьютерные модели. Такие модели позволяют визуализировать процесс преобразования логических значений входных сигналов в значения выходных сигналов.

Полусумматор. Вспомним, что при сложении двоичных чисел образуется сумма в данном разряде, при этом возможен перенос в старший разряд. Обозначим слагаемые A, B, перенос P и сумму S. Таблица сложения одноразрядных двоичных чисел с учетом переноса в старший разряд выглядит следующим образом (табл. 1.3).

Таблица 1.3. **Таблица сложения с учетом переноса**

Слагаемые		Перенос	Сумма
A	*B*	*P*	*S*
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Из этой таблицы сразу видно, что перенос можно реализовать с помощью операции логического умножения:

$$P = A \,\&\, B.$$

Получим теперь формулу для вычисления суммы. Значения суммы более всего совпадают с результатом операции логического сложения (кроме случая, когда на входы подаются две единицы, а на выходе должен получиться нуль).

Нужный результат достигается, если результат логического сложения умножить на инвертированный перенос. Таким образом, для определения суммы можно применить следующее логическое выражение:

$$S = (A \vee B)\&\overline{(A \,\&\, B)}.$$

Построим таблицу истинности для данного логического выражения и убедимся в правильности нашего предположения (табл. 1.4).

Таблица 1.4. **Таблица истинности логической функции $S = (A \vee B)\&(\overline{A \,\&\, B})$**

A	B	$A \vee B$	$A \,\&\, B$	$\overline{A \,\&\, B}$	$(A \vee B)\&(\overline{A \,\&\, B})$
0	0	0	0	1	0
0	1	1	0	1	1
1	0	1	0	1	1
1	1	1	1	0	0

Теперь на основе полученных логических выражений можно построить из базовых логических элементов схему полусумматора. Данная схема называется полусумматором, так как реализует суммирование одноразрядных двоичных чисел A и B без учета переноса из младшего разряда.

По логической формуле переноса легко определить, что для получения переноса необходимо использовать логический элемент И.

Анализ логической формулы для суммы показывает, что на выходе должен стоять элемент логического умножения И, который имеет два входа. На один из входов подается результат логического сложения исходных величин $A \vee B$, т. е. на него должен подаваться сигнал с элемента логического сложения ИЛИ.

На второй вход требуется подать результат инвертированного логического умножения исходных сигналов $\overline{A \,\&\, B}$, т. е. на второй вход подается сигнал с элемента НЕ, на вход которого, в свою очередь, поступает сигнал с элемента логического умножения И (рис. 1.39).

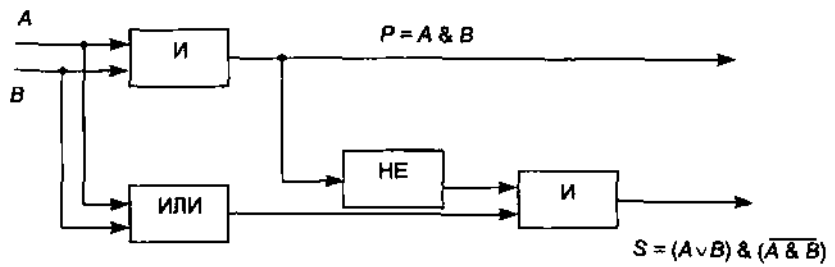


Рис. 1.39. Логическая схема полусумматора

Триггер. Оперативная память компьютера, а также внутренних регистров процессора состоит из ячеек, которые технически реализуются с помощью триггеров. Триггер — это устройство, которое позволяет записывать, хранить и считывать информацию (каждый триггер может хранить 1 бит информации).

Триггер можно построить из двух логических элементов ИЛИ и двух элементов НЕ (рис. 1.40).

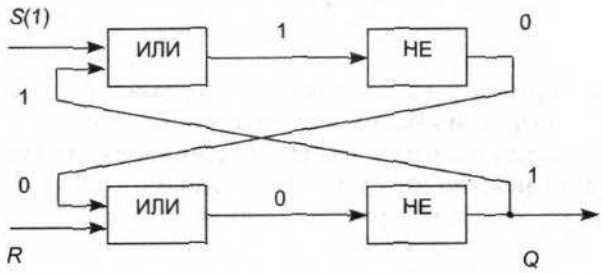


Рис. 1.40. Логическая схема триггера

В обычном состоянии на входы триггера подан сигнал 0, и триггер хранит 0. Для записи 1 на вход S (установочный) подается сигнал 1. Последовательно рассмотрев прохождение сигнала по схеме, увидим, что триггер переходит в это состояние и будет устойчиво находиться в нем и после того, как сигнал на входе S исчезнет. Триггер запомнил 1, т. е. *с выхода триггера Q можно считать 1.*

Для того чтобы сбросить информацию и подготовиться к приему новой информации, подается сигнал 1 на вход R (сброс), после чего триггер возвратится к исходному «нулевому» состоянию.

Вопросы для размышления

1. Можно ли логический элемент представить в виде таблицы истинности логического выражения?

1.8.2. Модели логических устройств компьютера на языке Visual Basic

В языке программирования Visual Basic основные логические операции могут быть реализованы с помощью логических операторов:

- **And** (логическое умножение);
- **Or** (логическое сложение);
- **Not** (логическое отрицание);
- **Xor** (исключающее **Or**, которое принимает логическое значение **True**, тогда и только тогда, когда лишь один из аргументов имеет значение **True**);
- **Eqv** (операция эквивалентности, которая принимает логическое значение **True**, когда оба аргумента имеют значения **True** или оба — **False**).

Логические операторы могут оперировать с логическими аргументами **True** (логическая единица) и **False** (логический нуль), а также с логическими переменными типа **Boolean**.

Построим компьютерную модель полусумматора с использованием языка программирования Visual Basic.

Проект «Полусумматор» на языке Visual Basic

1. Поместить на форму (рис. 1.41):
   - кнопку `Button1` для запуска обработчика события;
   - четыре надписи для изображения базовых логических элементов;
   - два текстовых поля `TextBoxA` и `TextBoxB` для ввода логических значений на входе полусумматора;
   - четыре надписи для вывода промежуточных логических значений `LabelOr` и `LabelNot`, а также итоговых значений суммы `LabelS` и переноса `LabelP`.
2. Создать обработчик события, реализующий определение логических значений на выходе каждого базового логического элемента и их вывод на надписи:

```
Dim A, B, P, S As Boolean
Sub cmd1_Click()
A=txtA.Text
B=txtB.Text
```

```
P=A And B
S=(A Or B) And Not (A And B)
LabelP.Text=P
LabelNot.Text=Not P
LabelOr.Text=A Or B
LabelS.Text=S
End Sub
```

3. Запустить проект, ввести в текстовые поля логические значения аргументов и щелкнуть по кнопке *Перенос и сумма*.
На надписи будут выведены логические значения на выходах логических элементов.

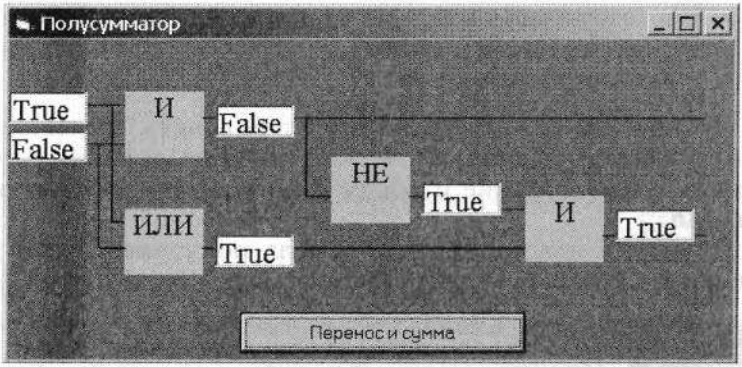
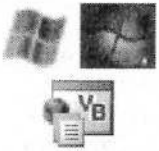


Рис. 1.41. Проект «Полусумматор» на языке Visual Basic

Проект хранится в папке
..\IIKT11Prof\VB2005\Полусумматор **Windows-CD**



Проект «Триггер» на языке Visual Basic

1. Поместить на форму (рис. 1.42):
   - два текстовых поля `TextBoxSet` и `TextBoxReset` для ввода начальных логических значений на входах триггера;
   - две надписи `LabelOr1` и `LabelOr2` для вывода промежуточных логических значений;
   - две надписи `LabelQ2` и `LabelQ1` для вывода состояний выходов триггера;

- четыре надписи для обозначения составляющих триггер логических элементов;
- четыре надписи для обозначения входов и выходов триггера;
- кнопку Button1 для запуска событийной процедуры установки значения триггера;
- кнопку Button2 для запуска событийной процедуры сброса значения триггера.

2. Определить логические переменные:

```
Dim S, R, Or1, Or2, Not1, Not2 As Boolean
```

3. Создать событийную процедуру установки значения триггера:

```
Private Sub Button1_Click(…)
S=TextBoxSet.Text
R=TextBoxReset.Text
Or1=S Or Not2
Not1=Not Or1
Or2=Not1 Or R
Not2=Not Or2
LabelOr1.Text=Or1
LabelOr2.Text=Or2
LabelQ2.Text=Not1
LabelQ1.Text=Not2
End Sub
```

4. Создать обработчик события сброса значения триггера:

```
Private Sub Button2_Click(…)
S=TextBoxSet.Text
R=TextBoxReset.Text
Or1=S Or Not2
Not1=Not Or1
Or2=Not1 Or R
Not2=Not Or2
LabelOr1.Text=Or1
LabelOr2.Text=Or2
LabelQ2.Text=Not1
LabelQ1.Text=Not2
End Sub
```

5. Запустить проект. Установить триггер — ввести в поле TextBoxSet значение **True**, проследить за установкой значений на элементах триггера.
Сбросить триггер — ввести в поле TextBoxReset значение **True**.

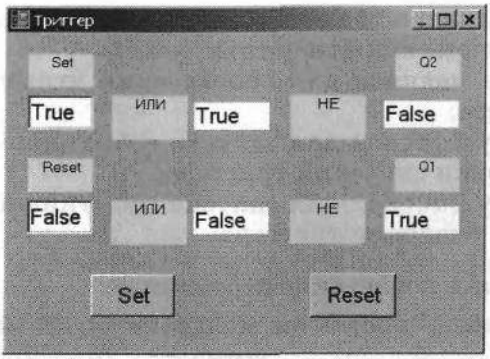


Рис. 1.42. Проект «Триггер» на языке Visual Basic

Проект хранится в папке ..\IIKT11Prof\VB2005\Trigger **Windows-CD**

Вопросы для размышления

1. Какие логические функции существуют в языке программирования Visual Basic?

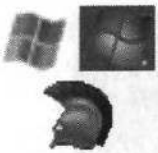
1.8.3. Модели логических устройств компьютера на языке Turbo Delphi

В языке программирования Turbo Delphi основные логические операции могут быть реализованы с помощью логических операторов:

- **And** (логическое умножение);
- **Or** (логическое сложение);
- **Not** (логическое отрицание);
- **Xor** (исключающее **Or**, которое принимает логическое значение **True**, тогда и только тогда, когда лишь один из аргументов имеет значение **True**);

Логические операторы могут оперировать с логическими аргументами **True** (логическая единица) и **False** (логический нуль), а также с логическими переменными типа `Boolean`.

Построим компьютерную модель полусумматора с использованием языка программирования Delphi.



Проект «Полусумматор» на языке Turbo Delphi

1. Поместить на форму (рис. 1.43):
   - кнопку Button1 для запуска событийной процедуры;
   - четыре надписи Label1, Label2, Label3 и Label4 для изображения базовых логических элементов;
   - два текстовых поля EditA и EditB для ввода логических значений на входе полусумматора;
   - четыре надписи для вывода промежуточных логических значений LabelOr и LabelNot, а также итоговых значений суммы LabelS и переноса LabelP.

2. Создать событийную процедуру, реализующую:
   - ввод значений на входе и преобразование их из строкового типа в логический с использованием функции StrToBool();
   - определение логических значений на выходе каждого базового логического элемента;
   - вывод полученных логических значений на надписи с использованием функции преобразования логического типа данных в строковый BoolToStr():

```
var
A: Boolean;
B: Boolean;
P: Boolean;
S: Boolean;
procedure TForm1.Button1Click(Sender: TObject);
begin
A:=StrToBool(EditA.Text);
B:=StrToBool(EditB.Text);
P:=(A And B);
S:=(A Or B) And (Not (A And B));
LabelP.Caption:=BoolToStr(P,True);
LabelNo.Caption:=BoolToStr(Not(P),True);
LabelOr.Caption:=BoolToStr((A Or B),True);
LabelS.Caption:=BoolToStr(S,True);
end;
end.
```

3. Запустить проект, ввести логические значения аргументов и щелкнуть по кнопке *Сумма и перенос*.
   На надписи будут выведены логические значения на выходах логических элементов.

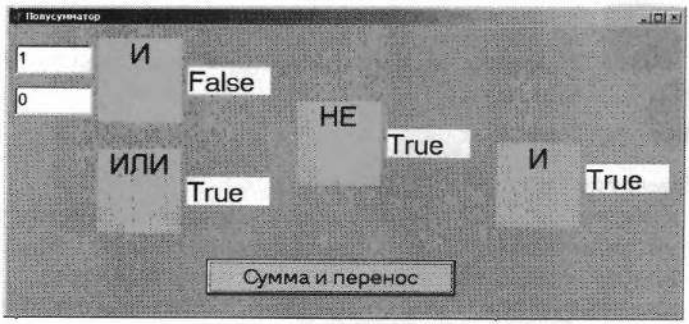
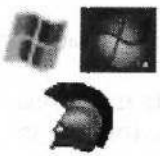


Рис. 1.43. Проект «Полусумматор» на языке Turbo Delphi

Проект хранится в папке ..\IIKT11Prof\Delphi\Summ **Windows-CD**



Проект «Триггер» на языке Turbo Delphi

1. Поместить на форму (рис. 1.44) шесть текстовых полей:
   - `EditSet` и `EditReset` — для ввода начальных логических значений на входах триггера;
   - `EditOr1Out` и `EditOr2Out` — для визуального контроля промежуточных логических значений;
   - `EditQ2` и `EditQ1` — для вывода состояния выходов триггера.
2. Поместить на форму надписи:
   - четыре для обозначения составляющих триггер логических элементов;
   - четыре для обозначения входов и выходов триггера.
3. Определить логические переменные:

```
var
S: boolean;
R: boolean;
Or1Out:  boolean;
Not1Out: boolean;
Or2Out:  boolean;
Not2Out: boolean;
```

4. Поместить на форму кнопку Button1 и создать для нее событийную процедуру установки значения триггера:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
S:=StrToBool(EditSet.Text);
R:=StrToBool(EditReset.Text);
Or1Out:=S Or Not2Out;
Not1Out:=Not Or1Out;
Or2Out:=Not1Out Or R;
Not2Out:=Not Or2Out;
EditOr1Out.Text:=BoolToStr(Or1Out,True);
EditOr2Out.Text:=BoolToStr(Or2Out,True);
EditQ2.Text:=BoolToStr(Not1Out,True);
EditQ1.Text:=BoolToStr(Not2Out,True);
end;
```

5. Запустить проект. Установить триггер, ввести в поле EditSet значение 1, а в поле EditReset — значение 0. Щелкнуть по кнопке *Set/Reset*. Проследить за установкой значений на элементах триггера. Сбросить триггер.

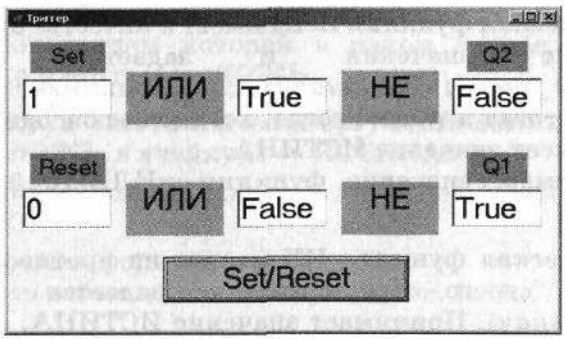


Рис. 1.44. Проект «Триггер» на языке Turbo Delphi

Проект хранится в папке ..\IIKT11Prof\Delphi\Trigger **Windows-CD**

Вопросы для размышления

1. Какие логические функции существуют в языке программирования Turbo Delphi?

1.8.4. Модели логических устройств компьютера в электронных таблицах

В электронных таблицах имеются логические функции, реализующие базовые логические операции. Аргументами и значениями логических функций являются логические значения ИСТИНА и ЛОЖЬ.

Логическое значение может быть получено как результат определения значения логического выражения. Например, для логического выражения 10>5 результатом будет логическое значение ИСТИНА, а для логического выражения A1<A2 (где в ячейке A1 хранится число 10, а в ячейке A2 — число 5) — значение ЛОЖЬ.

Логическая функция И имеет в качестве аргументов логические значения, которые могут быть истинными или ложными, и задается формулой =И(*лог_знач1; лог_знач2;* ...). Принимает значение ИСТИНА тогда и только тогда, когда все аргументы имеют значение ИСТИНА.

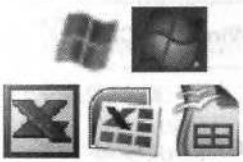
Например, значение функции =И(10>5;10<5) – ЛОЖЬ.

Логическая функция ИЛИ имеет в качестве аргументов логические значения и задается формулой =ИЛИ(*лог_знач1; лог_знач2;* ...). Принимает значение ИСТИНА тогда и только тогда, когда хотя бы один из аргументов имеет значение ИСТИНА.

Например, значение функции =ИЛИ(10>5;10<5) — ИСТИНА.

Логическая функция НЕ меняет на противоположное значение своего аргумента и задается формулой =НЕ(*лог_знач*). Принимает значение ИСТИНА, если аргумент имеет значение ЛОЖЬ, и наоборот.

Например, значение функции =НЕ(10>5) — ЛОЖЬ.



Таблицы истинности базовых логических операций

Получим таблицу истинности операции логического умножения, значением которой в трех случаях является ЛОЖЬ и только в одном — ИСТИНА.

1. В пары ячеек (A2, B2), (A3, B3), (A4, B4) (A5, B5) ввести пары значений аргументов логической операции (ЛОЖЬ, ЛОЖЬ), (ИСТИНА, ЛОЖЬ), (ЛОЖЬ, ИСТИНА) и (ИСТИНА, ИСТИНА).
2. В ячейку C2 ввести формулу логической функции И: =И(A2;B2)
3. Скопировать формулу в ячейки C3, C4 и C5.

Получим таблицу истинности операции логического сложения, значением которой в одном случае является ЛОЖЬ, в трех других — ИСТИНА.

4. В пары ячеек (E2, F2), (E3, F3), (E4, F4) (E5, F5) ввести пары значений аргументов логической операции (ЛОЖЬ, ЛОЖЬ), (ИСТИНА, ЛОЖЬ), (ЛОЖЬ, ИСТИНА) и (ИСТИНА, ИСТИНА).
5. В ячейку G2 ввести формулу логической функции ИЛИ: =ИЛИ(E2;F2).
6. Скопировать формулу в ячейки G3, G4 и G5.

Получим таблицу истинности операции логического отрицания, значением которой в одном случае является ИСТИНА, а в другом — ЛОЖЬ.

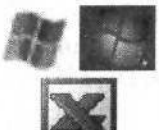
7. В ячейку I2 ввести значение аргумента логической операции ЛОЖЬ, а ячейку I3 — ИСТИНА.
8. В ячейку J2 ввести формулу логической функции НЕ: =НЕ(I2).
9. Скопировать формулу в ячейку J3.

Получим таблицу, показанную на рис. 1.45.

	A	B	C	D	E	F	G	H	I	J
1	Логическое функция "И"				Логическая функция "ИЛИ"				Логическая функция "НЕ"	
2	ЛОЖЬ	ЛОЖЬ	ЛОЖЬ		ЛОЖЬ	ЛОЖЬ	ЛОЖЬ		ЛОЖЬ	ИСТИНА
3	ИСТИНА	ЛОЖЬ	ЛОЖЬ		ИСТИНА	ЛОЖЬ	ИСТИНА		ИСТИНА	ЛОЖЬ
4	ЛОЖЬ	ИСТИНА	ЛОЖЬ		ЛОЖЬ	ИСТИНА	ИСТИНА			
5	ИСТИНА	ИСТИНА	ИСТИНА		ИСТИНА	ИСТИНА	ИСТИНА			

Рис. 1.45. Таблицы истинности базовых логических операций

Таблицы истинности хранятся в папке ..\IIKT11Prof в файле Model.xls на листе *Логические функции* **Windows-CD**



Проект «Полусумматор» в электронных таблицах Microsoft Excel 2003

1. На листе *Полусумматор* предусмотреть ввод логических значений аргументов в ячейки B1 и B2.
   В ячейку B3 ввести формулу =И(B1;B2).
   В ячейку B4 ввести формулу =НЕ(B3).
   В ячейку B5 ввести формулу =ИЛИ(B1;B2).
   В ячейку B6 ввести формулу =И(B5;B4).

	A	B
1	A=	ЛОЖЬ
2	B=	ИСТИНА
3	P=	=И(B1;B2)
4	Not P =	=НЕ(B3)
5	Or =	=ИЛИ(B1;B2)
6	S=	=И(B5;B4)

Рис. 1.46. Формулы для модели полусумматора

2. Для создания графического интерфейса проекта запустить систему объектно-ориентированного программирования Visual Basic командой [*Сервис-Макрос-Редактор Visual Basic*].

3. В окне системы программирования добавить форму командой [*Insert-UserForm*].

4. Поместить на форму (рис. 1.47):
   - кнопку `CommandButton1` для запуска событийной процедуры;
   - четыре надписи для отображения базовых логических элементов;
   - два текстовых поля `TextBoxA` и `TextBoxB` для ввода логических значений на входе полусумматора;
   - четыре надписи для вывода промежуточных логических значений, а также итоговых значений суммы и переноса.

5. Создать событийную процедуру:

```
Private Sub CommandButton1_Click()
Cells(1, 2)=TextBoxA.Text
Cells(2, 2)=TextBoxB.Text
Label1.Caption=Cells(3, 2)
Label3.Caption=Cells(4, 2)
Label2.Caption=Cells(5, 2)
Label4.Caption=Cells(6, 2)
End Sub
```

6. Запустить проект, ввести значения аргументов и щелкнуть по кнопке *Перенос и сумма*.

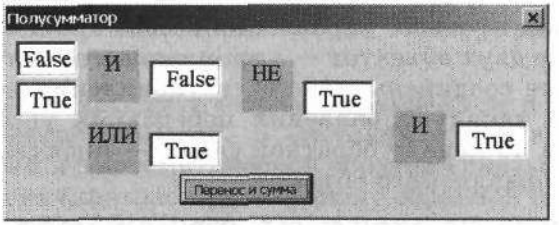


Рис. 1.47. Модель полусумматора в электронных таблицах

Проект хранится в папке ..\IIKT11Prof в файле Model1.xls на листе *Полусумматор* **Windows-CD**

Вопросы для размышления

1. Какие логические функции существуют в электронных таблицах?

1.9. Информационные модели управления объектами

1.9.1. Информационные модели систем управления

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 **управления** и **обратной связи**. По каналу управления передаются управляющие сигналы, а по каналу обратной связи — информация о состоянии управляемого объекта.

Системы управления без обратной связи (разомкнутые). 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. 1.48.

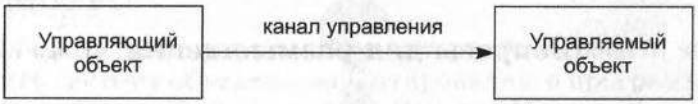


Рис. 1.48. Система управления без обратной связи

В качестве примера системы управления без обратной связи можно рассмотреть запуск неуправляемых ракет. Неуправляемая ракета поразит цель только в том случае, когда она запущена точно в мишень.

Системы управления с обратной связью (замкнутые). 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 реальных параметрах объекта управления. Это позволяет осуществлять управление с гораздо большей точностью.

Информационную модель системы управления с обратной связью можно наглядно представить с помощью схемы на рис. 1.49.

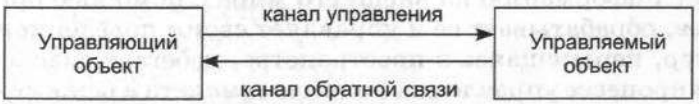


Рис. 1.49. Система управления с обратной связью

Примером использования системы управления с обратной связью являются управляемые ракеты. Оператор получает координаты цели и ракеты и может по каналу управления скорректировать направление полета ракеты так, чтобы она поразила мишень.

Системы с автоматическим управлением и автоматической обратной связью. В системах с автоматическим управлением и с автоматической обратной связью управляющий объект (компьютер) по каналу обратной связи получает информацию о реальных параметрах объекта управления и по прямому каналу управления производит необходимые действия над ним. Это позволяет осуществлять управление с гораздо большей точностью.

Информационную модель системы с автоматическим управлением и автоматической обратной связью можно наглядно представить с помощью схемы на рис. 1.50.

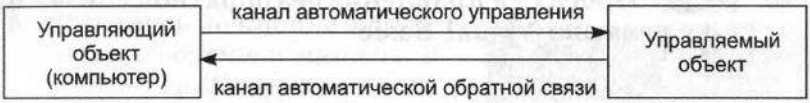


Рис. 1.50. Система управления с автоматическим управлением и автоматической обратной связью

Примером использования системы с автоматическим управлением и автоматической обратной связью являются самонаводящиеся ракеты. Оператор получает координаты цели и запускает ракету. Далее ракетой управляет компьютер, находящийся на земле или на самой ракете. Он может в реальном времени скорректировать направление полета ракеты так, чтобы она поразила мишень.

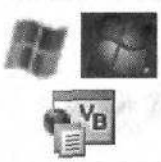
Вопросы для размышления

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?

1.9.2. Модели систем управления на языке Visual Basic

Системы управления без обратной связи. Для демонстрации принципа работы систем управления без обратной связи разработаем компьютерную модель на языке программирования Visual Basic.

Пусть управляемым объектом будет точка (закрашенный кружок), которую управляющий объект (пользователь) должен переместить в центр мишени (круга). Прямое управление положением точки будем производить путем нажатия кнопок, которые перемещают объект вверх, вниз, влево и вправо. Обратная связь будет отсутствовать, так как текущее положение управляемого объекта (точки) будет невидимым (точка рисоваться не будет).



Проект «Управление без обратной связи» на языке Visual Basic

1. Поместить на форму (рис. 1.51):
   - графическое поле `PictureBox1`, по которому будет перемещаться точка (закрашенный кружок);
   - кнопку `Button1` для запуска обработчика события вывода первоначального положения точки и круга;
   - кнопку `Button2` для запуска обработчика события вывода конечного положения точки;
   - четыре кнопки `Button3`, `Button4`, `Button5` и `Button6` для управления движением точки.

2. Создать обработчик события вывода первоначального положения управляемого объекта (точки). Обработчик должен включать случайную генерацию координат точки:

```
Dim Graph1 As Graphics
Dim Pen1 As New Pen(Color.Black, 3)
Dim Pen2 As New Pen(Color.Red, 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
```

3. Четыре обработчика события перемещения точки должны обеспечивать изменение координат точки. Обработчик события перемещения вправо примет вид:

```
Private Sub Button4_Click(…)
X=X+1
End Sub
```

4. Создать обработчик события, выводящий конечное положение управляемого объекта (точки):

```
Private Sub Button2_Click()
Graph1.DrawEllipse(Pen1, X, Y, 2, 2)
Graph1.FillEllipse(Brush1, X, Y, 2, 2)
End Sub
```

5. Щелкнуть по кнопке *Управляемый объект и мишень*. В центре графического поля появится окружность (мишень) и точка со случайными координатами (управляемый объект).
Переместить управляемый объект (точку) в центр мишени щелчками по кнопкам со стрелками.
Щелкнуть по кнопке *Результат*. Скорее всего, управляемый объект (точка) не попадет в центр мишени (см. рис. 1.51).

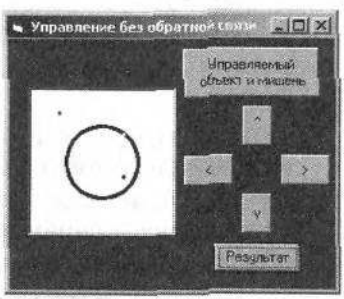


Рис. 1.51. Проект «Управление без обратной связи» на языке Visual Basic

Проект хранится в папке ..\IIKT11Prof\VB2005\Управление без обратной связи **Windows-CD**

Системы управления с обратной связью. Для демонстрации принципа работы систем управления с обратной связью разработаем компьютерную модель. Для осуществления обратной связи сделаем текущие положения управляемого объекта (точки) видимыми (точку будем рисовать красным цветом), а также будем выводить текущие значения координат точки (центра закрашенного кружка) в текстовые поля.

Проект «Управление с обратной связью» на языке Visual Basic

Усовершенствуем проект «Управление без обратной связи».

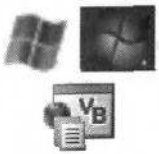
1. Поместить на форму дополнительно две надписи Label1 и Label2 для вывода текущих координат точки (рис. 1.52).
2. В программный код обработчиков событий перемещения точки добавить строки рисования точки и вывода на надписи ее текущих координат.
   Обработчик события перемещения точки влево примет вид:

```
Private Sub Button3_Click(…)
X=X-1
Graph1.DrawEllipse(Pen2, X, Y, 2, 2)
Graph1.FillEllipse(Brush1, X, Y, 2, 2)
Label1.Text=X
Label2.Text=Y
End Sub
```

3. Запустить проект и осуществить попадание управляемого объекта (точки) в мишень.
   Легко убедиться, что использование обратной связи обеспечивает гарантированное попадание управляемого объекта (точки) в мишень (см. рис. 1.52).

Рис. 1.52. Проект «Управление с обратной связью» на языке Visual Basic

Проект хранится в папке ..\IIKT11Prof\VB2005\Управление с обратной связью **Windows-CD**



Проект «Автоматическое управление с автоматической обратной связью» на языке Visual Basic

За основу возьмем проект «Управление с обратной связью».

1. Убрать из графического интерфейса (рис. 1.53): четыре кнопки Button3, Button4, Button5 и Button6 для управления движением точки и кнопку Button2, выводящую результат.
2. Поместить на графический интерфейс кнопку Button2 для создания автоматической обратной связи.
3. Для осуществления автоматической обратной связи использовать корректировку координат с использованием инструкции выбора **Select Case**. Мишень-круг имеет координаты центра (100, 100). Тогда обработчик события корректировки положения точки примет вид:

```
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(Pen2, X1, Y1, 2, 2)
Graph1.FillEllipse(Brush1, X1, Y1, 2, 2)
End Sub
```

4. Запустить проект и нажатием кнопки *Шаг* осуществить попадание точки в мишень — окружность, имеющую координаты центра (100,100) (см. рис. 1.53).

Рис. 1.53. Модель системы управления с автоматической обратной связью

Проект хранится в папке ..\IIKT11Prof\VB2005\Управление с автоматической обратной связью **Windows-CD**

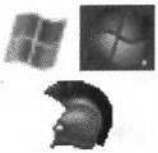
Вопросы для размышления

1. В чем состоит различие между системами управления с обратной связью и системами автоматического управления с автоматической обратной связью?

1.9.3. Модели систем управления на языке Turbo Delphi

Системы управления без обратной связи. Для демонстрации принципа работы систем управления без обратной связи разработаем компьютерную модель на языке программирования Turbo Delphi.

Пусть управляемым объектом будет точка, которую управляющий объект (пользователь) должен переместить в центр мишени (круга). Прямое управление положением точки будем производить путем нажатия кнопок, которые перемещают объект (точку) вверх, вниз, влево и вправо. Обратная связь будет отсутствовать, так как текущее положение управляемого объекта (точки) будет невидимым (точка рисоваться не будет).



Проект «Управление без обратной связи» на языке Turbo Delphi

1. Поместить на форму (рис. 1.54):
   - графическое поле Image1, по которому будет перемещаться точка (закрашенный кружок);
   - кнопку Button1 для вывода первоначального положения управляемого объекта (точки) и мишени (круга);
   - четыре кнопки ButtonUp, ButtonD, ButtonL и ButtonR для управления движением точки;
   - кнопку Button2 для вывода конечного положения управляемого объекта (точки).

2. Событийная процедура вывода первоначального положения управляемого объекта (точки) должна обеспечивать случайную генерацию координат центра закрашенного кружка (точки), выбор цвета и типа закрашивания, рисовать мишень в центре графического поля, а также стирать предыдущий вариант:

```
var
X1: integer;
Y1: integer;
procedure TForm1.Button1Click(Sender: TObject);
begin
//Стирание
Image1.Canvas.Brush.Color:=clWhite;
Image1.Canvas.Rectangle(0,0,200,200);
Image1.Canvas.FillRect(Rect(0,0,200,200));
Randomize;
X1:=Random(200);
Y1:=Random(200);
Form1.Image1.Canvas.Brush.Color:=clRed;
Form1.Image1.Canvas.Brush.Style:=bsSolid;
Form1.Image1.Canvas.Ellipse(X1-3,Y1-3,X1+3,Y1+3);
Form1.Image1.Canvas.Brush.Color:=clBlack;
Form1.Image1.Canvas.Brush.Style:=bsClear;
Form1.Image1.Canvas.Ellipse(80,80,120,120);
end;
```

3. Четыре событийные процедуры перемещения точки должны обеспечивать изменение ее координат:

```
procedure TForm1.ButtonLClick(Sender: TObject);
begin
X1:=X1-1;
end;
procedure TForm1.ButtonRClick(Sender: TObject);
begin
X1:=X1+1;
end;
procedure TForm1.ButtonUpClick(Sender:
TObject);
begin
Y1:=Y1-1;
end;
procedure TForm1.ButtonDClick(Sender: TObject);
begin
Y1:=Y1+1;
end;
```

4. Событийная процедура вывода конечного положения управляемого объекта (точки):

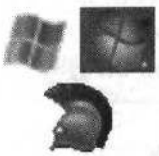
```
procedure TForm1.Button2Click(Sender: TObject);
begin
Form1.Image1.Canvas.Brush.Color:=clBlack;
Form1.Image1.Canvas.Brush.Style:=bsSolid;
Form1.Image1.Canvas.Ellipse(X1-3,Y1-3,X1+3,Y1+3);
end;
```

5. Щелкнуть по кнопке *Управляемый объект и мишень*. *Переместить управляемый объект (точку) в центр мишени* щелчками по кнопкам со стрелками.
Щелкнуть по кнопке *Результат*. Скорее всего, управляемый объект (точка) не попадет в центр мишени (см. рис. 1.54).

Рис. 1.54. Проект «Управление без обратной связи» на языке Turbo Delphi

Проект хранится в папке ..\IIKT11Prof\Delphi\Upr1 Windows-CD

Системы управления с обратной связью. Для демонстрации принципа работы систем управления с обратной связью разработаем компьютерную модель. Для осуществления обратной связи сделаем текущие положения управляемого объекта (точки) видимыми (точку будем рисовать красным цветом), а также будем выводить текущие значения координат точки (центра закрашенного кружка) в текстовые поля.



Проект «Управление с обратной связью» на языке Turbo Delphi

Усовершенствуем проект «Управление без обратной связи».

1. Поместить на форму дополнительно две надписи `LabelX` и `LabelY` для вывода текущих координат точки (рис. 1.55).

2. Четыре событийные процедуры перемещения точки должны обеспечивать изменение координат точки, а также рисовать ее текущие положения. Событийная процедура перемещения влево примет вид:

```
procedure TForm1.ButtonLClick(Sender: TObject);
begin
X1:=X1-1;
Form1.Image1.Canvas.Pen.Color:=clRed;
Form1.Image1.Canvas.Brush.Color:=clRed;
Form1.Image1.Canvas.Brush.Style:=bsSolid;
Form1.Image1.Canvas.Ellipse(X1-3,Y1-3,X1+3,Y1+3);
LabelX.Caption:=IntToStr(X1);
LabelY.Caption:=IntToStr(Y1);
end;
```

3. Запустить проект и осуществить попадание точки в мишень (круг, имеющий центр с координатами (100, 100), размещенный в графическом поле с размерами (200, 200)).
   Легко убедиться, что использование обратной связи обеспечивает гарантированное попадание точки в мишень (см. рис. 1.55).

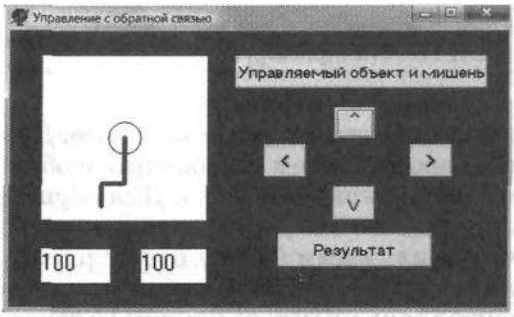


Рис. 1.55. Проект «Управление с обратной связью» на языке Turbo Delphi

Проект хранится в папке ..\IIKT11Prof\Delphi\Upr2 **Windows-CD**

Проект «Автоматическое управление с автоматической обратной связью» на языке Turbo Delphi

1. Поместим на форму (рис. 1.56):
   - две кнопки: `Button1` — для запуска графического интерфейса проекта и `Button2` — для запуска автоматической обратной связи;
   - две надписи `Label1` и `Label2` для вывода текущих координат управляемого объекта (точки);
   - графическое поле `Image1`, по которому будет перемещаться точка (закрашенный кружок).

2. Событийная процедура вывода первоначального положения управляемого объекта (точки) должна обеспечивать случайную генерацию координат точки (центра закрашенного кружка), выбор цвета и типа закрашивания, рисовать мишень (круг с координатами центра (100, 100)) в центре графического поля, а также стирать предыдущий вариант:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
//Стирание
Image1.Canvas.Brush.Color:=clWhite;
Image1.Canvas.Rectangle(0,0,200,200);
Image1.Canvas.FillRect(Rect(0,0,200,200));
Randomize;
```

```
X1:=Random(200);
Y1:=Random(200);
Form1.Image1.Canvas.Brush.Color:=clRed;
Form1.Image1.Canvas.Brush.Style:=bsSolid;
Form1.Image1.Canvas.Ellipse(X1-3,Y1-3,X1+3,Y1+3);
Form1.Image1.Canvas.Brush.Style:=bsClear;
Form1.Image1.Canvas.Ellipse(80,80,120,120);
end;
```

3. Для осуществления автоматической обратной связи использовать корректировку координат с использованием инструкции выбора **If-Then-Else**. Событийная процедура корректировки положения точки примет вид:

```
procedure TForm1.Button2Click(Sender: TObject);
begin
X2:=100;
Y2:=100;
//Автоматическая корректировка координаты X
If X2-X1>0 Then X1:=X1+1 Else X1:=X1-1;
//Автоматическая корректировка координаты Y
If Y2-Y1>0 Then Y1:=Y1+1 Else Y1:=Y1-1;
Label1.Caption:=IntToStr(X1);
Label2.Caption:=IntToStr(Y1);
Form1.Image1.Canvas.Brush.Color:=clRed;
Form1.Image1.Canvas.Pen.Color:=clRed;
Form1.Image1.Canvas.Brush.Style:=bsSolid;
Form1.Image1.Canvas.Ellipse(X1-3,Y1-3,X1+3,Y1+3);
end;
end.
```

4. Запустить проект и нажатием кнопки *Шаг* осуществить попадание точки в мишень — окружность, имеющую координаты центра (100,100) (см. рис. 1.56).

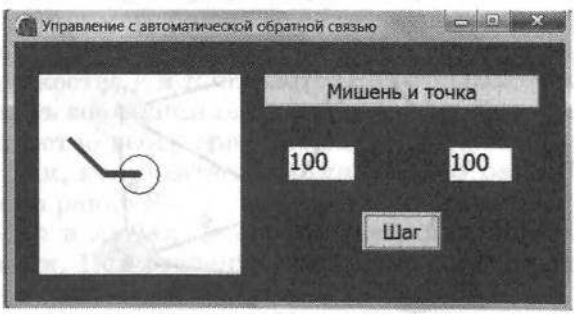


Рис. 1.56. Модель системы управления с автоматической обратной связью на языке Turbo Delphi

Проект хранится в папке ..\IIKT11Prof\Delphi\Upr3 Windows-CD

Вопросы для размышления

1. В чем состоит различие между системами управления с обратной связью и без обратной связи?

1.10. Графы и их исследование с использованием языков объектно-ориентированного программирования Visual Basic и Turbo Delphi

1.10.1. Введение в теорию графов

При проектировании компьютерных сетей, телефонных линий, трубопроводов и строительстве дорог необходимо минимизировать затраты на прокладку коммуникаций. Прежде всего, целесообразно выбрать минимальный по длине маршрут прокладки коммуникаций.

Например, необходимо соединить телефонным или компьютерным кабелем шесть зданий, расстояния между которыми различны (рис. 1.57). Возникает задача определения маршрута прокладки кабеля минимальной длины, но при этом подходящего к каждому зданию. Для решения таких задач часто используют теорию графов.

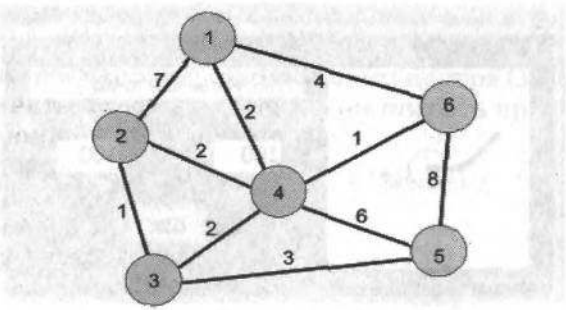


Рис. 1.57. Задача прокладки коммуникаций

Основные понятия теории графов. Граф G задается с помощью пары множеств $G = (V, R)$, где V — множество (совокупность) вершин, R — множество ребер, соединяющих пары вершин. Множество ребер может быть пустым, если ни одна из вершин не соединена с другими вершинами. Обычно граф представляют с помощью схемы, на которой некоторые вершины соединены линиями (ребрами) (рис. 1.58).

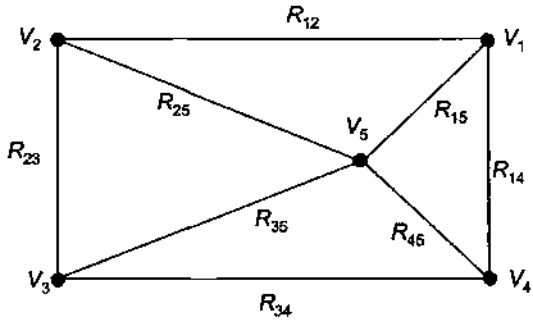


Рис. 1.58. Граф G в форме схемы

Вершинами могут служить объекты любой природы: будь то населенные пункты, компьютеры сети, элементы блок-схем алгоритмов и т. д. Под **ребрами** могут подразумеваться дороги между двумя соседними городами, стороны геометрических фигур, линии связи между компьютерами. Любую систему улиц в городе можно представить в виде графа. Здесь вершины — это перекрестки.

Вершины называются **смежными**, если их соединяет ребро. Например, на рис. 1.58 смежны вершины V_1 и V_2, так как их соединяет ребро R_{12}.

Множества V и R являются конечными — мы можем перечислить все вершины и ребра графа. Количество вершин и количество ребер графа определяют **мощности множеств** V и R. Так, количество вершин графа G равно 5, а количество ребер равно 8.

Ребро и любая из его двух вершин называются **инцидентными**. Под степенью вершины подразумевается количество инцидентных ей ребер. Так, степень вершины V_1 равна 3, а степень вершины V_5 равна 4.

Маршрут графа — это последовательность чередующихся вершин и ребер. В графах можно выделить различные маршруты. Маршрут является **замкнутым (циклом)**, если его начальная и конечная вершины совпадают. Например, в графе G можно выделить несколько циклических маршрутов, например $V_1R_{12}V_2R_{23}V_3R_{34}V_4R_{14}V_1$ и $V_2R_{23}V_3R_{35}V_5R_{25}V_2$.

Маршрут называется **простой цепью**, если все его вершины и ребра различны.

Длина маршрута равна количеству ребер, входящих в него.

Одна вершина достижима из другой, если между ними проложен маршрут. Граф считается **связным**, если каждая его вершина достижима из любой другой. Например, граф G является связным, так как между любыми двумя его вершинами можно проложить маршрут. Например, маршрут между вершинами V_1 и V_4 может быть следующим: $V_1R_{12}V_2R_{23}V_3R_{34}V_4$.

Вершины, которые не имеют инцидентных ребер, называются **изолированными вершинами**. Можно также сказать, что степень таких вершин нулевая. Из всего этого следует, что изолированные вершины недостижимы из любых других вершин.

Ориентированные графы. Сообразительный читатель может задать вопрос: имеет ли значение направление, ориентация ребра? Ведь в системе улиц можно найти дороги как с односторонним, так и с двусторонним движением. Именно поэтому в теории графов вводится понятие **орграфа — ориентированного графа**. В орграфе каждое ребро имеет одно направление. Такие ребра называются **дугами**. Другими словами, ребро — это неупорядоченная пара вершин, а дуга — упорядоченная. Очевидным является тот факт, что дуги R_{12} и R_{21} не совпадают в орграфе. Для орграфа вводятся такие понятия, как **входящая** и **исходящая степени вершины**. Это соответственно число входящих в вершину дуг и число исходящих из нее дуг.

Взвешенные графы. Очень часто в практических применениях желательно приписывать ребрам графа веса для того, чтобы моделировать такие величины, как расстояние, время перехода или стоимость доставки между двумя точками. **Взвешенный граф (сеть)** — это такой граф, ребрам или дугам которого поставлены в соответствие числовые величины. **Вес** сети равен сумме весов ее ребер.

Описание графа с помощью матрицы смежности. Для наглядного представления графа используются схемы (см. рис. 1.58), а для математических расчетов удобнее использовать представление графа в форме **матрицы смежности** (рис. 1.59, *а*). Матрицу смежности можно представить в виде таблицы, строки и столбцы которой соответствуют номерам вершин графа. Если вершины смежны, то элемент матрицы смежности равен 1, если вершины не смежны, то элемент матрицы равен 0. Диагональные элементы матрицы равны 0, так как вершины сами с собой не смежны (их не соединяет ребро).

Пусть в матрице смежности графа G строки нумеруются индексом n, а столбцы — индексом k, тогда элементы матрицы смежности обозначаются R_{nk}. Для смежных вершин элементы матрицы смежности (R_{12}, R_{14}, R_{15}, R_{21}, R_{23}, R_{25}, R_{32}, R_{34}, R_{35}, R_{41}, R_{43}, R_{45}, R_{51}, R_{52}, R_{53}, R_{54}) равны 1, а для не смежных вершин элементы матрицы (R_{13}, R_{24}, R_{31}, R_{42}) равны 0. Диагональные элементы матрицы (R_{11}, R_{22}, R_{33}, R_{44}, R_{55}) равны 0.

Преобразуем граф G в сеть, т. е. присвоим ребрам, соединяющим смежные вершины, числовые значения ($R_{12} = 50$, $R_{14} = 25$, $R_{15} = 10$, $R_{21} = 50$, $R_{23} = 25$, $R_{25} = 30$, $R_{32} = 25$, $R_{34} = 50$, $R_{35} = 35$, $R_{41} = 25$, $R_{43} = 50$, $R_{45} = 15$, $R_{51} = 10$, $R_{52} = 30$, $R_{53} = 35$, $R_{54} = 15$). Отобразим сеть G в форме матрицы смежности (рис. 1.59, *б*).

	1	2	3	4	5
1	0	1	0	1	1
2	1	0	1	0	1
3	0	1	0	1	1
4	1	0	1	0	1
5	1	1	1	1	0

а

	1	2	3	4	5
1	0	50	0	25	10
2	50	0	25	0	30
3	0	25	0	50	35
4	25	0	50	0	15
5	10	30	35	15	0

б

Рис. 1.59. Граф и сеть G в форме матрицы смежности

Подграфы и деревья. Подграфом графа G называется граф, у которого все вершины и ребра принадлежат графу G (*рис. 1.60, а*).

Остовной связный подграф — это подграф графа G, который содержит все его вершины и каждая его вершина достижима из любой другой (рис. 1.60, *б*).

Дерево — это граф, в котором нет циклов, т. е. граф, в котором нельзя из некоторой вершины пройти по нескольким различным ребрам и вернуться в ту же вершину. **Остовным связным деревом** называется подграф, включающий все вершины исходного графа G, каждая вершина которого достижима из любой другой, и при этом не содержащий циклов (рис. 1.60, *в*).

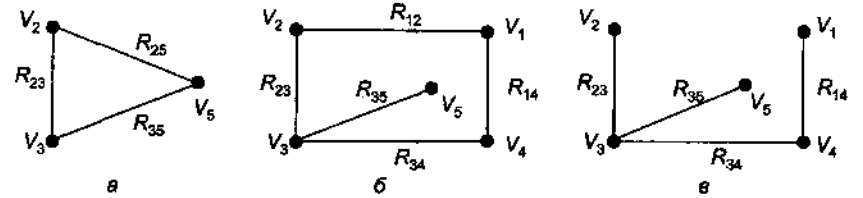


Рис. 1.60. *а* — подграф графа G, *б* — остовной связный подграф графа G, *в* — остовное связное дерево

Преобразование графа в остовное связное дерево минимального веса. Пусть $G = (V, R)$ — связный взвешенный неориентированный граф, где V — множество вершин, а R — множество ребер (см. рис. 1.58). Ребра графа не ориентированы, т. е. ребра R_{nk} и R_{kn} считаются одним и тем же ребром, поэтому в матрице смежности не учитываются дублирующие друг друга ребра. В результате граф G можно представить с помощью матрицы смежности, содержащей значения весов десяти ребер (на рис. 1.61 выделены жирным шрифтом).

	1	2	3	4	5
1	0	**50**	**0**	**25**	**10**
2	50	0	**25**	**0**	**30**
3	0	25	0	**50**	**35**
4	25	0	50	0	**15**
5	10	30	35	15	0

Рис.1.61. Матрица смежности связного взвешенного неориентированного графа G

Введем понятие цикломатического числа γ, показывающего, сколько ребер графа нужно удалить, чтобы в нем не осталось ни одного цикла. Цикломатическое число γ равно

увеличенной на единицу разности между количеством ребер и количеством вершин графа:

$$\gamma = m - n + 1,$$

где m — количество ребер, n — количество вершин.

Например, для графа, изображенного на рис. 1.58, цикломатическое число равно:

$$\gamma = m - n + 1 = 8 - 5 + 1 = 4 .$$

Это значит, что для получения остовного связного дерева в графе G необходимо убрать четыре ребра, и тогда в нем не останется ни одного цикла.

Для каждого графа обычно существует несколько остовных связных деревьев, которые обладают различными весами. На рис. 1.60, *в* представлено остовное связное дерево графа G, вес которого равен 135, а на рис. 1.62 представлены три остовных связных дерева графа G, веса которых равны 130, 100, 135.

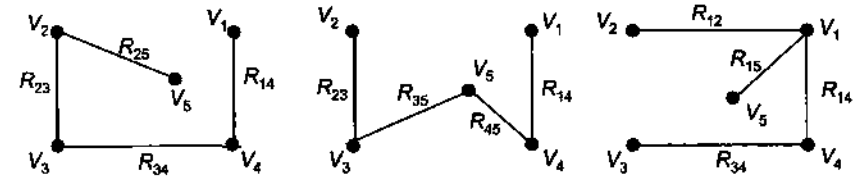


Рис. 1.62. Остовные связные деревья графа G

Для построения остовного связного дерева минимального веса используется алгоритм Крускала.

1. Первоначально из графа удаляются все ребра, получается остовной подграф, где все вершины изолированы. Каждая вершина такого графа помещается в одноэлементное подмножество.
2. Ребра сортируются по возрастанию весов.
3. Ребра последовательно, по возрастанию их весов, включаются в остовное дерево. Существуют четыре случая:

   а) обе вершины включаемого ребра принадлежат одноэлементным подмножествам, тогда они объединяются в новое, связное подмножество;

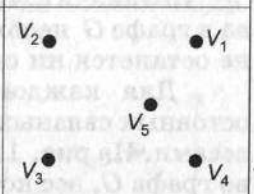
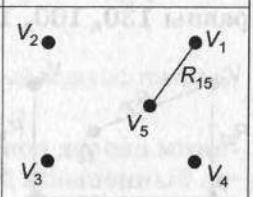
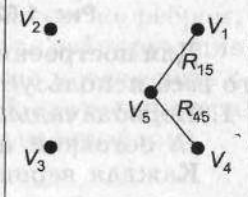
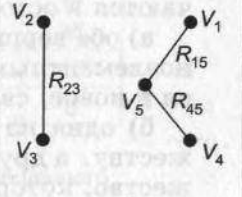
   б) одна из вершин принадлежит связному подмножеству, а другая нет, тогда включаем вторую в подмножество, которому принадлежит первая;

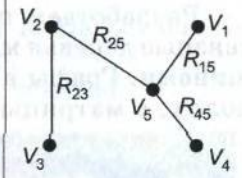
   в) обе вершины принадлежат разным связным подмножествам, тогда объединяем подмножества;

   г) обе вершины принадлежат одному связному подмножеству, тогда исключаем данное ребро.

4. Алгоритм заканчивает работу, когда все вершины будут объединены в одно множество, при этом оставшиеся ребра не включаются в остовное дерево.

Рассмотрим реализацию этого алгоритма на примере построения остовного дерева минимального веса для графа G.

№	Выполняемые действия	Множество вершин	Граф
1	Построим остовной подграф, содержащий только изолированные вершины	Каждая вершина исходного графа помещается в одноэлементное подмножество, получаем пять одноэлементных подмножеств: $\{V_1\}$, $\{V_2\}$, $\{V_3\}$, $\{V_4\}$, $\{V_5\}$	
2	Найдем ребро минимального веса (в данном случае R_{15}) и добавим его в остовной подграф	Образуется связное подмножество вершин: $\{V_1, V_5\}$. Сохраняются одноэлементные подмножества вершин: $\{V_2\}$, $\{V_3\}$, $\{V_4\}$	
3	Среди оставшихся ребер найдем ребро минимального веса (в данном случае R_{45}) и добавим его в остовной подграф	Так как одна вершина ребра входит в связное подмножество, добавим в него и вторую: $\{V_1, V_5, V_4\}$. Сохраняются одноэлементные подмножества вершин: $\{V_2\}$, $\{V_3\}$	
4	Среди оставшихся ребер найдем ребро минимального веса (в данном случае R_{23}) и добавим его в остовной подграф	Так как ни одна из вершин ребра не входит в связное подмножество, создадим второе связное подмножество: $\{V_2, V_3\}$. Существует также первое связное подмножество: $\{V_1, V_5, V_4\}$.	

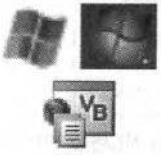
№	Выполняемые действия	Множество вершин	Граф
5	Среди оставшихся ребер найдем ребро минимального веса (в данном случае R_{25}) и добавим его в остовной подграф	Так как одна вершина ребра входит в одно связное подмножество, а другая вершина — в другое связное подмножество, эти подмножества объединяются в единое связное множество: $\{V_1, V_5, V_4, V_2, V_3\}$.	
6	Остальные ребра включать в граф не надо, так как все их вершины уже принадлежат одному связному множеству		
7	Получен граф, который: • является **остовным**, так как включает все вершины; • является **связным**, так как все вершины в нем можно соединить маршрутами; • является **деревом**, так как в нем отсутствуют циклы; • **обладает минимальным весом**, так как в него последовательно включались ребра, отсортированные по возрастанию весов		
8	Полученное остовное связное дерево обладает минимальным весом: $R_{23} + R_{25} + R_{15} + R_{45} = 25 + 30 + 10 + 15 = 80$		
9	Цикломатическое число графа G равно $\gamma = m - n + 1 = 8 - 5 + 1 = 4$, что соответствует количеству ребер, не включенных в остовное связное дерево		

Вопросы для размышления

1. В какой форме можно представить граф?
2. В чем состоит различие между ориентированными и неориентированными графами?
3. Какие графы являются деревьями?
4. Какой граф обладает минимальным весом?

1.10.2. Изучение графов на языке Visual Basic

Разработаем проект, позволяющий получать остовные связные деревья минимального веса для графов с пятью вершинами. Графы в виде схем будем рисовать в графических полях, а матрицы смежности выводить в поля списков.



Проект «Построение остовного связного дерева графа» на языке Visual Basic

1. Запустить систему объектно-ориентированного программирования Visual Basic командой [*Программы-Visual Basic 2005 Express Edition*].

Создадим графический интерфейс проекта.

2. Поместить на форму (рис. 1.63):
   - графическое поле `PictureBox1` для рисования первоначального графа;
   - графическое поле `PictureBox2` для рисования остовного связного дерева минимального веса;
   - кнопку `Button1` для запуска обработчика события, вывода в первое графическое поле вершин графа;
   - пять полей списков `ListBox1`, `ListBox2`, `ListBox3`, `ListBox4`, `ListBox5` для вывода элементов матрицы смежности связного взвешенного ориентированного графа;
   - кнопку `Button2` для запуска обработчика события, вывода элементов матрицы смежности взвешенного ориентированного графа;
   - пять полей списков `ListBox6`, `ListBox7`, `ListBox8`, `ListBox9`, `ListBox10` для вывода элементов матрицы смежности связного взвешенного неориентированного графа;
   - кнопку `Button3` для запуска обработчика события, вывода элементов матрицы смежности взвешенного неориентированного графа;
   - три поля списков `ListBox11`, `ListBox12`, `ListBox13` для вывода номеров вершин и весов ребер остовного связного дерева;

- кнопку `Button4` для запуска обработчика события, вывода во второе графическое поле полученного остовного связного дерева;
- надпись `Label1` для вывода суммы весов остовного связного дерева минимального веса;
- надписи для вывода пояснительных текстов.

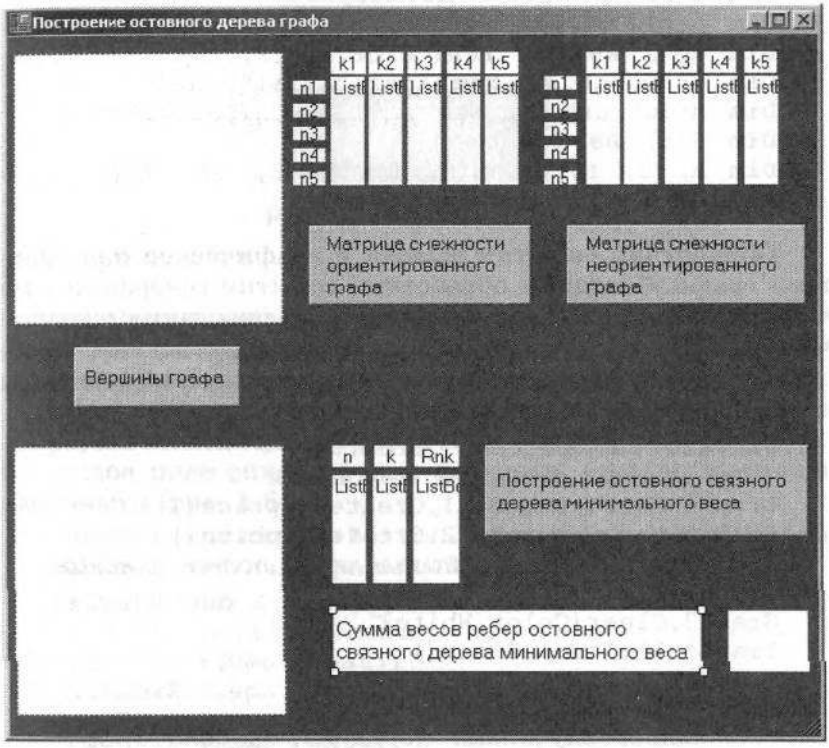


Рис. 1.63. Графический интерфейс проекта

3. Объявить:
   - области рисования `Graph1` и `Graph2`;
   - два пера `Pen1` и `Pen2` для рисования;
   - шрифт `drawFont` и кисть `drawBrush` для вывода номеров вершин в графических полях;
   - пять объектов типа точка `P(5)` для хранения координат вершин графа;
   - целочисленные переменные `I`, `N`, `K` для использования в качестве счетчиков циклов;

- целочисленную переменную A для хранения значения, возвращаемого функцией MsgBox();
- три целочисленных двумерных массива R(5, 5), R1(5, 5), R2(5, 5) для хранения весов ребер ориентированного графа, неориентированного графа и остовного связного дерева минимального веса.

```
Dim Graph1, Graph2 As Graphics
Dim Pen1 As New Pen(Color.Black, 1)
Dim Pen2 As New Pen(Color.Black, 3)
Dim drawFont As New Font("Arial", 12)
Dim drawBrush As New SolidBrush(Color.Black)
Dim P(5) As Point
Dim A, I, N, K, R(5, 5), R1(5, 5), R2(5, 5)
As Short
```

Обработчик события вывода в графическое поле вершин графа. Создадим обработчик события генерации случайных координат вершин графа и их рисования в графическом поле.

4. Щелкнуть по кнопке Button1 и в заготовку обработчика события ввести программный код:

```
Private Sub Button1_Click(…)
'Определение областей рисования
Graph1=Me.PictureBox1.CreateGraphics()
Graph2=Me.PictureBox2.CreateGraphics()
'Очистка областей рисования и полей списков
Graph1.Clear(Color.White)
Graph2.Clear(Color.White)
ListBox11.Items.Clear()
ListBox12.Items.Clear()
ListBox13.Items.Clear()
'Генерация случайных координат вершин графа и
их рисование
For I=1 To 5
  P(I).X=Int(Rnd()*200)
  P(I).Y=Int(Rnd()*200)
  Graph1.DrawEllipse(Pen2, P(I).X, P(I).Y, 2, 2)
  Graph1.DrawString(I, drawFont, drawBrush,
      P(I).X, P(I).Y)
Next I
End Sub
```

5. Запустить проект и осуществить щелчок по кнопке *Вершины графа*.

В графическое поле будут выведены вершины графа и их номера (рис. 1.64).

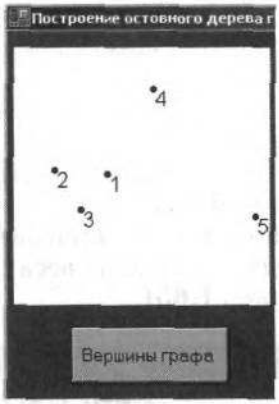


Рис. 1.64. Вершины графа

Обработчик события вывода элементов матрицы смежности взвешенного ориентированного графа. Во вложенном цикле со счетчиками N (строки матрицы смежности) и K (столбцы матрицы смежности) осуществим рисование ребер графа, вычисление весов ребер и их вывод в поля списков. В каждое поле списка будем выводить столбец матрицы смежности (рис. 1.65).

6. Щелкнуть по кнопке Button2 и в заготовку обработчика события ввести программный код.

```
Private Sub Button2_Click(…)
'Очистка полей списков
ListBox1.Items.Clear()
ListBox2.Items.Clear()
ListBox3.Items.Clear()
ListBox4.Items.Clear()
ListBox5.Items.Clear()
'Во вложенном цикле рисование ребер графа, вы-
числение весов ребер и их вывод в поля
списков
For N=1 To 5
For K=1 To 5
Graph1.DrawLine(Pen1, P(N).X, P(N).Y, P(K).X,
P(K).Y)
R(N,K)=Math.Sqrt((P(N).X-P(K).X)^2+(P(N).Y-
      P(K).Y)^2)
Next K
ListBox1.Items.Add(R(N,1))
ListBox2.Items.Add(R(N,2))
```

```
ListBox3.Items.Add(R(N,3))
ListBox4.Items.Add(R(N,4))
ListBox5.Items.Add(R(N,5))
Next N
End Sub
```

7. Осуществить щелчок по кнопке *Матрица смежности ориентированного графа*.
В графическом поле будут нарисованы ребра графа, а в поля списков будут выведены веса ребер ориентированного графа (см. рис. 1.65).

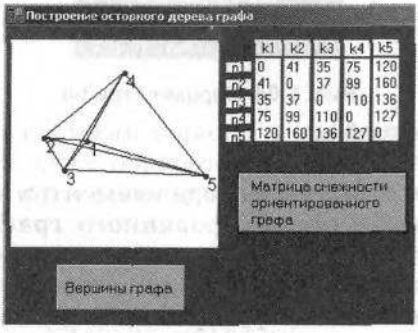


Рис. 1.65. Остовное дерево и матрица смежности ориентированного графа

Обработчик события вывода элементов матрицы смежности взвешенного неориентированного графа. Во вложенном цикле со счетчиками N (строки матрицы смежности) и K (столбцы матрицы смежности) осуществим вычисление весов ребер и их вывод в поля списков. Для вывода половины элементов матрицы смежности начальному значению счетчика вложенного цикла присвоим значение K = N + 1. В каждое поле списка выводить столбец матрицы смежности (рис. 1.66).

8. Щелкнуть по кнопке Button3 и в заготовку обработчика события ввести программный код:

```
Private Sub Button3_Click(…)
'Очистка полей списков
ListBox6.Items.Clear()
ListBox7.Items.Clear()
ListBox8.Items.Clear()
ListBox9.Items.Clear()
ListBox10.Items.Clear()
```

```
'Во вложенном цикле вычисление весов ребер и
их вывод в поля списков
For N=1 To 5
For K=N+1 To 5
Graph1.DrawLine(Pen1, P(N).X, P(N).Y, P(K).X,
P(K).Y)
R1(N,K)=Math.Sqrt((P(N).X-P(K).X)^2+(P(N).Y-
P(K).Y)^2)
Next K
ListBox6.Items.Add(R1(N,1))
ListBox7.Items.Add(R1(N,2))
ListBox8.Items.Add(R1(N,3))
ListBox9.Items.Add(R1(N,4))
ListBox10.Items.Add(R1(N,5))
Next N
End Sub
```

9. Осуществить щелчок по кнопке *Матрица смежности неориентированного графа*.
В поля списков будут выведены веса ребер неориентированного графа (см. рис. 1.66).

Рис. 1.66. Матрица смежности неориентированного графа

Обработчик события построения остовного связного дерева минимального веса. В этой процедуре пользователь строит остовное связное дерево минимального веса. Ребро минимального веса выбирается с использованием матрицы смежности неориентированного графа. Ввод номеров вершин ребра минимального веса осуществляется с помощью функции ввода данных `InputBox()`. Включение или невключение выбранного ребра в остовное дерево производится на основании пункта 3 алгоритма Крускала, который реализуется с использованием функции вывода сообщений `MsgBox()` и оператора условного перехода в сокращенной форме.

В результате остовное связное дерево минимального веса будет нарисовано в графическом поле, номера его вершин и веса ребер будут выведены в поля списков, суммарный вес ребер будет выведен на надпись.

10. Щелкнуть по кнопке `Button4` и в заготовку обработчика события ввести программный код:

```
Private Sub Button4_Click(…)
'Ввод номера первой вершины и ее рисование в
графическом поле
N=InputBox("Введите номер первой вершины:",
"Выбор ребра минимального веса")
Graph2.DrawEllipse(Pen2, P(N).X, P(N).Y, 2, 2)
Graph2.DrawString(N, drawFont, drawBrush,
P(N).X, P(N).Y)

'Ввод номера второй вершины и ее рисование в
графическом поле
K=InputBox ("Введите номер второй вершины:",
"Выбор ребра минимального веса")
Graph2.DrawEllipse(Pen2, P(K).X, P(K).Y, 2, 2)
Graph2.DrawString(K, drawFont, drawBrush,
P(K).X, P(K).Y)

'Реализация пункта 3 алгоритма Крускала
A=MsgBox("Ребро с вершинами " + Str(N) +
" и " + Str(K) + " включить в состав
остовного дерева, если выполняется хотя бы одно
из условий: (Обе вершины не входят в остовное
дерево) или (Одна из вершин не входит в
остовное дерево) или (Вершины входят в
различные подмножества остовного дерева).", 32
+ 4, "Включить ребро в остовное дерево?")
If A=6 Then
Graph2.DrawLine(Pen1, P(N).X, P(N).Y, P(K).X,
P(K).Y)
R2(N,K)=Math.Sqrt((P(N).X-P(K).X)^2+(P(N).Y-
P(K).Y)^2)
ListBox11.Items.Add(N)
ListBox12.Items.Add(K)
ListBox13.Items.Add(R2(N, K))
Label1.Text=R2(N,K)+Val(Label1.Text)
End If
End Sub
```

На основе анализа матрицы смежности неориентированного графа выберем ребро минимального веса.

11. Осуществить щелчок по кнопке *Построение остовного связного дерева минимального веса*.

В появившемся диалоговом окне *Выбор ребра минимального веса* ввести номер первой точки и щелкнуть по кнопке *OK* (рис. 1.67).

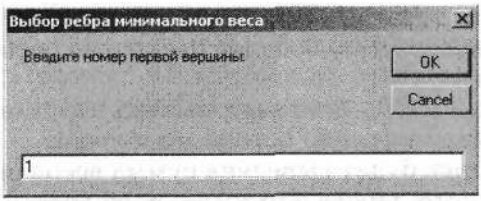


Рис. 1.67. Ввод первой вершины

12. В появившемся диалоговом окне *Выбор ребра минимального веса* ввести номер второй точки и щелкнуть по кнопке *OK* (рис. 1.68).

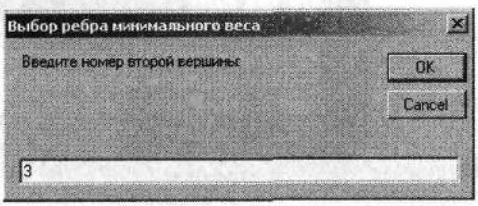


Рис. 1.68. Ввод второй вершины

На основе анализа остовного связного дерева в графическом поле выполним пункт 3 алгоритма Крускала и решим вопрос о включении или невключении выбранного ребра в остовное дерево.

13. В появившемся диалоговом окне *Включить ребро в остовное дерево?* подтвердить или опровергнуть истинность условий щелчком по кнопке *Да* или по кнопке *Нет* (рис. 1.69).

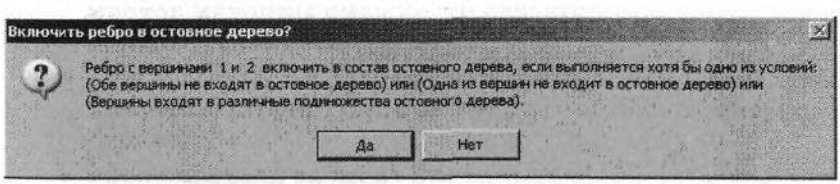


Рис.1.69. Включение ребра в остовное дерево

14. Выполнять пункты 11–13 до тех пор, пока остовное связное дерево с минимальным весом не будет построено, т. е. пока все вершины не войдут в связное множество.

В результате (рис. 1.70):

- в графическом поле будет построено остовное связное дерево;
- в полях списка появятся номера вершин и веса соответствующих ребер остовного дерева;
- на надпись будет выведена сумма весов ребер остовного связного дерева минимального веса.

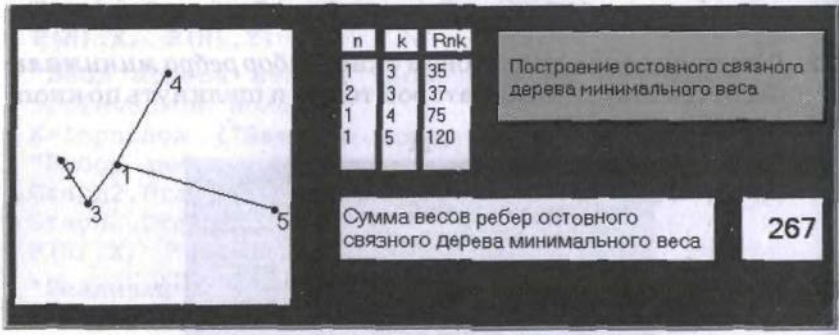


Рис. 1.70. Минимальный вес остовного дерева

Проект хранится в папке ..\IIKT11Prof\VB2005\Графы1 **Windows-CD**

Вопросы для размышления

1. Какие обработчики событий входят в проект определения минимального веса остовного дерева графа?

1.10.3. Изучение графов на языке Turbo Delphi

Разработаем проект, позволяющий получать остовные связные деревья минимального веса для графов с пятью вершинами. Графы в виде схем будем рисовать в графических полях, а матрицы смежности выводить в элемент управления, представляющий собой таблицу.

Проект «Построение остовного связного дерева графа» на языке Turbo Delphi

1. Запустить систему объектно-ориентированного программирования Turbo Delphi командой [*Программы-Borland Developer Studio 2006-Turbo Delphi*].

   Создадим графический интерфейс проекта.

2. Поместить на форму (рис.1.71):
   - графическое поле `Image1` для рисования первоначального графа;
   - графическое поле `Image2` для рисования остовного связного дерева минимального веса;
   - кнопку `Button1` для запуска событийной процедуры вывода вершин графа в первое графическое поле;
   - кнопку `Button2` для запуска событийной процедуры вывода элементов матрицы смежности взвешенного ориентированного графа;
   - управляющий элемент `StringGrid1` для вывода элементов матрицы смежности связного взвешенного ориентированного графа;
   - кнопку `Button3` для запуска событийной процедуры вывода элементов матрицы смежности взвешенного неориентированного графа;
   - управляющий элемент `StringGrid2` для вывода элементов матрицы смежности связного взвешенного неориентированного графа;
   - кнопку `Button4` для запуска событийной процедуры вывода во второе графическое поле остовного связного дерева;
   - управляющий элемент `StringGrid3` для вывода весов ребер остовного связного дерева;
   - надпись `Label1` для вывода суммы весов остовного связного дерева минимального веса;
   - надписи для вывода пояснительных текстов.

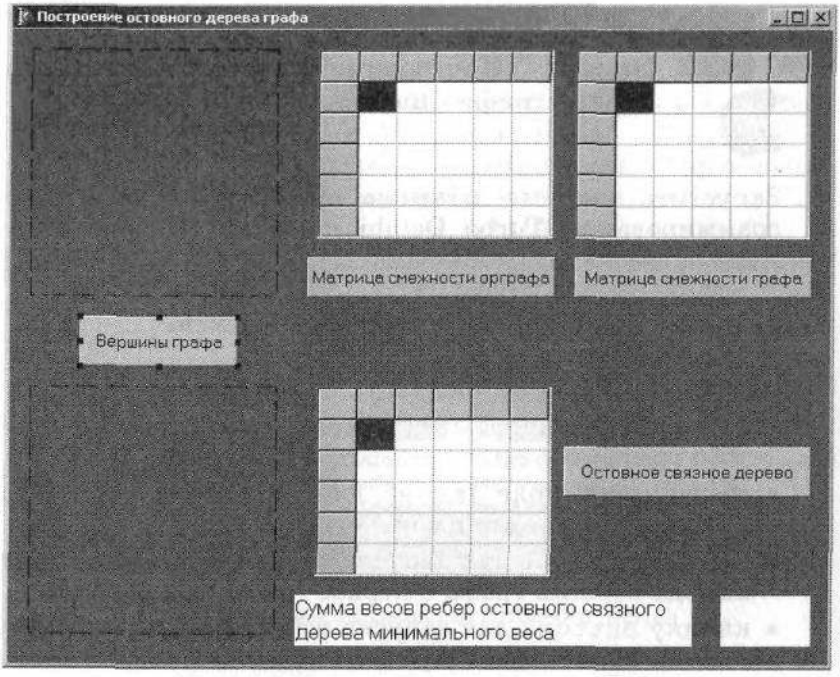


Рис. 1.71. Графический интерфейс проекта

3. Объявить:
   - целочисленные пятиэлементные массивы `X[I]` и `Y[I]` для хранения координат вершин графа;
   - целочисленный двумерный массив `R[I]` для хранения весов ребер графа;
   - целочисленный двумерный массив `R1[I]` для хранения весов ребер остовного связного дерева минимального веса;
   - целочисленные переменные `I`, `N`, и `K`, используемые в качестве счетчиков циклов;
   - целочисленную переменную `S` для хранения суммы весов ребер остовного связного дерева минимального веса;
   - целочисленную переменную `A` для хранения значения, возвращаемого функцией `MessageDlg()`;
   - строковые переменные `strN` и `strK` для хранения номеров точек, возвращаемых функцией `InputBox()`.

```
var
X:  array[1..5] of integer;
Y:  array[1..5] of integer;
R:  array[1..5,1..5] of integer;
R1: array[1..5,1..5] of integer;
I, N, K, A: byte;
S: integer;
strN, strK: string;
```

Событийная процедура вывода в графическое поле вершин графа. Создадим событийную процедуру генерации случайных координат вершин графа и их рисование в графическом поле (рис. 1.72).

4. Щелкнуть по кнопке Button1 и в заготовку событийной процедуры ввести программный код:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
//Очистка областей рисования и обнуление
//переменной
Image1.Canvas.Brush.Color:=clWhite;
Image1.Canvas.Rectangle(0,0,200,200);
Image2.Canvas.Brush.Color:=clWhite;
Image2.Canvas.Rectangle(0,0,200,200);
S:=0;
//Обозначение строк и столбцов в элементах
//управления StringGrid
For I:=0 To 6 Do
 begin
 StringGrid1.Cells[I,0]:=IntToStr(I);
 StringGrid1.Cells[0,I]:=IntToStr(I);
 StringGrid2.Cells[I,0]:=IntToStr(I);
 StringGrid2.Cells[0,I]:=IntToStr(I);
 StringGrid3.Cells[I,0]:=IntToStr(I);
 StringGrid3.Cells[0,I]:=IntToStr(I);
 end;
//Генерация случайных координат вершин графа
//и их рисование
 For I:=1 To 5 Do
  begin
  X[I]:=Random(200);
  Y[I]:=Random(200);
  Image1.Canvas.Pen.Width:=3;
  Image1.Canvas.Ellipse(X[I],Y[I],X[I]+4,Y[I]+4);
```

```
        Image1.Canvas.TextOut(X[I]+5,Y[I],
                              IntToStr(I));
    end;
end;
```

5. Запустить проект и осуществить щелчок по кнопке *Вершины графа*.
В графическое поле будут выведены вершины графа и их номера (рис. 1.72).

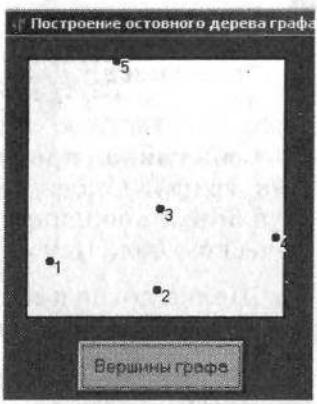


Рис. 1.72. Вершины графа

Событийная процедура вывода элементов матрицы смежности взвешенного ориентированного графа. Во вложенном цикле со счетчиками n (строки матрицы смежности) и k (столбцы матрицы смежности) осуществим рисование ребер ориентированного графа, вычисление весов ребер и их вывод в таблицу (рис. 1.73).

6. Щелкнуть по кнопке Button2 и в заготовку событийной процедуры ввести программный код:

```
procedure TForm1.Button2Click(Sender: TObject);
begin
//Во вложенном цикле рисование ребер
//ориентированного графа, вычисление весов
//ребер и их вывод в таблицу
For N:=1 To 5 Do
For K:=1 To 5 Do
begin
Image1.Canvas.Pen.Width:=1;
Image1.Canvas.MoveTo(X[N],Y[N]);
Image1.Canvas.LineTo(X[K],Y[K]);
R[N,K]:=Round(Sqrt(Sqr(X[N]-X[K])+Sqr(Y[N]-Y[K])));
StringGrid1.Cells[N,K]:=IntToStr(R[N,K]);
end;
end;
```

7. Осуществить щелчок по кнопке *Матрица смежности орграфа*.
В графическом поле будут нарисованы ребра графа, а в таблицу будут выведены веса ребер ориентированного графа (рис. 1.73).

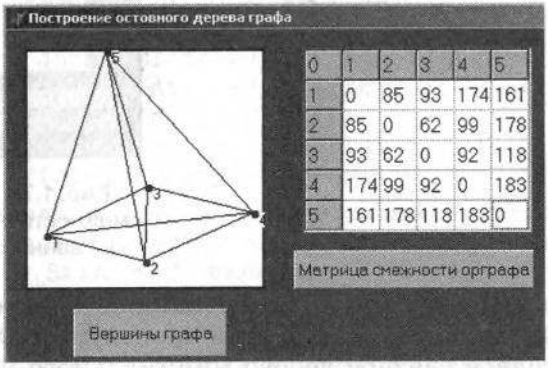


Рис. 1.73. Построение остовного дерева графа

Событийная процедура вывода элементов матрицы смежности взвешенного неориентированного графа. Во вложенном цикле со счетчиками N (строки матрицы смежности) и K (столбцы матрицы смежности) осуществим вычисление весов ребер неориентированного графа и их вывод в таблицу (рис. 1.74). Для вывода половины элементов матрицы смежности начальному значению счетчика вложенного цикла присвоим значение K=N+1.

8. Щелкнуть по кнопке Button3 и в заготовку событийной процедуры ввести программный код.

```
procedure TForm1.Button3Click(Sender: TObject);
begin
//Во вложенном цикле вычисление весов ребер
//неориентированного графа и их вывод в таблицу
For N:=1 To 5 Do
For K:=N+1 To 5 Do
begin
R[N,K]:=Round(Sqrt(Sqr(X[N]-X[K])+
Sqr(Y[N]-Y[K])));
StringGrid2.Cells[N,K]:=IntToStr(R[N,K]);
end;
end;
```

9. Осуществить щелчок по кнопке *Матрица смежности графа*. В таблицу будут выведены веса ребер неориентированного графа (рис. 1.74).

Рис. 1.74. Матрица смежности неориентированного графа

Событийная процедура построения остовного связного дерева минимального веса. В этой процедуре пользователь строит остовное связное дерево минимального веса. Ребро минимального веса выбирается с использованием матрицы смежности неориентированного графа. Ввод номеров вершин ребра минимального веса осуществляется с помощью функции ввода данных `InputBox()`. Включение или невключение выбранного ребра в остовное дерево производится на основании пункта 3 алгоритма Крускала, который реализуется с использованием функции вывода сообщений `MessageDlg()` и оператора условного перехода в сокращенной форме.

В результате остовное связное дерево минимального веса будет нарисовано в графическом поле, веса ребер будут выведены в таблицу, а суммарный вес ребер — на надпись.

10. Щелкнуть по кнопке `Button4` и в заготовку событийной процедуры ввести программный код:

```
procedure TForm1.Button4Click(Sender: TObject);
begin
//Ввод номера первой вершины и ее рисование
//в графическом поле
strN:=InputBox('Выбор ребра минимального
веса', 'Введите номер первой вершины:','');
Image2.Canvas.Pen.Width:=3;
Image2.Canvas.Ellipse(X[StrToInt(strN)],
      Y[StrToInt(strN)],X[StrToInt(strN)]+4,
      Y[StrToInt(strN)]+4);
```

```
Image2.Canvas.TextOut(X[StrToInt(strN)]+5,
      Y[StrToInt(strN)],strN);
//Ввод номера второй вершины и ее рисование
//в графическом поле
strK:=InputBox('Выбор ребра минимального веса',
'Введите номер второй вершины:','');
Image2.Canvas.Pen.Width:=3;
Image2.Canvas.Ellipse(X[StrToInt(strK)],
      Y[StrToInt(strK)],X[StrToInt(strK)]+4,
      Y[StrToInt(strK)]+4);
Image2.Canvas.TextOut(X[StrToInt(strK)]+5,
      Y[StrToInt(strK)],StrK);
//Реализация пункта 3 алгоритма Крускала
A:=MessageDlg('Ребро с вершинами ' + StrN +
' и ' + StrK + ' включить в состав остовного
дерева, если выполняется хотя бы одно из
условий: (Обе вершины не входят в остовное
дерево) или (Одна из вершин не входит
в остовное дерево) или (Вершины входят
в различные подмножества остовного дерева).',
MtConfirmation,[mbYes,mbNo], 0);
If A=mrYes
Then
begin
Image2.Canvas.Pen.Width:=1;
Image2.Canvas.MoveTo(X[StrToInt(strN)],
      Y[StrToInt(strN)]);
Image2.Canvas.LineTo(X[StrToInt(strK)],
      Y[StrToInt(strK)]);
R1[StrToInt(strN),StrToInt(strK)]:=
      Round(Sqrt(Sqr(X[StrToInt(strN)]-
      X[StrToInt(strK)])+
      Sqr(Y[StrToInt(strN)]-
      Y[StrToInt(strK)])));
StringGrid3.Cells[StrToInt(strN),StrToInt(strK)]:=
    IntToStr(R1[StrToInt(strN),StrToInt(strK)]);
```

На основе анализа матрицы смежности неориентированного графа выберем ребро минимального веса.

11. Осуществить щелчок по кнопке *Остовное связное дерево*.

В появившемся диалоговом окне *Выбор ребра минимального веса* ввести номер первой точки и щелкнуть по кнопке *OK* (рис. 1.75).

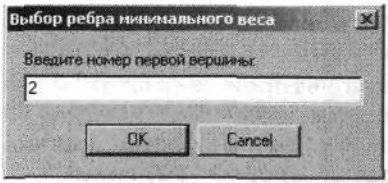


Рис. 1.75. Ввод первой вершины графа

12. В появившемся диалоговом окне *Выбор ребра минимального веса* ввести номер второй точки и щелкнуть по кнопке *OK* (рис. 1.76).

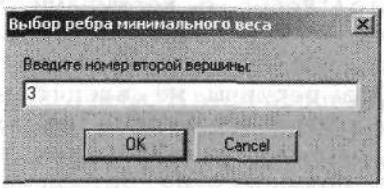


Рис. 1.76. Ввод вторвой вершины графа

На основе анализа остовного связного дерева в графическом поле выполним пункт 3 алгоритма Крускала и решим вопрос о включении или невключении выбранного ребра в остовное дерево.

13. В появившемся диалоговом окне *Confirm* подтвердить или опровергнуть истинность условий щелчком по кнопке *Yes* или по кнопке *No* (рис. 1.77).

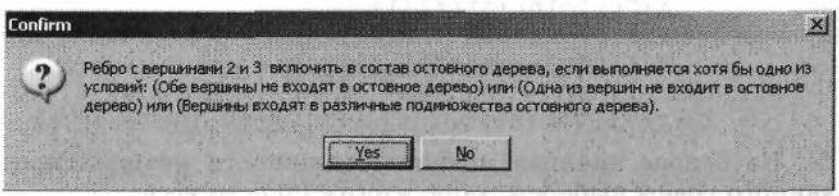


Рис. 1.77. Включение ребра в состав остовного дерева

14. Выполнять пункты 11–13 до тех пор, пока остовное связное дерево с минимальным весом не будет построено, т. е. пока все вершины не войдут в связное множество.

В результате (рис. 1.78):

- в графическом поле будет построено остовное связное дерево;
- в соответствующие ячейки таблицы будут выведены веса ребер остовного дерева;
- на надпись будет выведена сумма весов ребер остовного связного дерева минимального веса.

Рис. 1.78. Остовное дерево минимального веса

Проект хранится в папке ..\IIKT11Prof\Delphi\Графы **Windows-CD**

Вопросы для размышления

1. Какие событийные процедуры входят в проект определения минимального веса остовного дерева графа?

Глава 2

Технологии создания и обработки текстовой информации

При изучении данной главы рекомендуется установить следующее программное обеспечение для операционных систем Windows и Linux:

		Windows-CD	
	• браузеры SeaMonkey, Mozilla, Opera;		
	• OpenOffice.org Writer;		
	• настольную издательскую систему Scribus;		
		Первая помощь ПО 1.0.	
	• программу создания и редактирования файлов в формате PDF Adobe Acrobat Professional;		CD-31: 36
	• Microsoft Office 2007 (Word 2007);		CD-5
	• мультиязычный словарь ABBYY Lingvo;		CD-50: 51
	• систему оптического распознавания символов ABBYY FineReader;		CD-52
	• браузер Internet Explorer;		CD-2:4

- браузеры SeaMonkey, Mozilla;
- OpenOffice.org Writer;
- настольную издательскую систему Scribus;
- компьютерный словарь StarDict;
- систему оптического распознавания символов Kooka

Linux-DVD

2.1. Основные типы приложений для создания документов

Для обработки текстовой информации на компьютере используются текстовые редакторы, которые позволяют создавать, редактировать, форматировать, сохранять и распечатывать документы.

Простые текстовые редакторы. Простые текстовые редакторы (например, Блокнот в операционных системах Windows и Linux) позволяют создавать и редактировать тексты. Такие редакторы осуществляют простейшее форматирование шрифта, но не позволяют форматировать текст (абзацы).

Достоинством простых текстовых редакторов является то, что текст сохраняется в наиболее универсальном текстовом формате *Только текст* (расширение в имени файла txt) и содержит только коды символов и код перевода строки. Поэтому такие текстовые файлы читаются не только во всех текстовых редакторах, но и в различных операционных системах. Достоинством этого формата является также небольшой информационный объем файлов.

Текстовые редакторы (текстовые процессоры). Более совершенные текстовые редакторы (например, Microsoft Word в операционной системе Windows и OpenOffice.org Writer в операционных системах Windows и Linux) позволяют создавать довольно сложные документы (рефераты, брошюры и т. д.).

Такие редакторы имеют широкий спектр возможностей по созданию документов (форматирование шрифтов и абзацев, вставка списков, таблиц и оглавлений, проверка орфографии, сохранение исправлений, использование стилей, вставка объектов и др.).

Каждый из таких редакторов может сохранять документы в своем оригинальном формате. Формат *Документ Word* (расширение в имени файла doc) является оригинальным форматом текстового редактора Microsoft Word, в котором полностью сохраняет форматирование. Этот формат фактически является универсальным, так как понимается практически всеми текстовыми редакторами. В последней версии Microsoft Office используется формат DOCX, к счастью, для перевода в формат DOC существует конвертор.

В интегрированном офисном приложении OpenOffice.org используется открытый формат документов для офисных приложений OpenDocument Format (ODF), в том числе текстовых документов (ODT).

В данных текстовых редакторах может использоваться также *Расширенный текстовый формат* (расширение в имени файла rtf), который является также универсальным форматом текстовых файлов, в котором сохраняются результаты форматирования. Недостатком этого формата является большой информационный объем файлов.

Web-редакторы. Документы в Интернете публикуются в формате *Web-страниц* (расширение в имени файла htm или html), который использует для форматирования специальные управляющие символы (тэги). Достоинством этого формата является его универсальность, так как Web-страницы могут просматриваться с использованием специализированных программ (браузеров) в любых операционных системах.

В качестве Web-редакторов могут использоваться специализированные редакторы (в операционных системах Windows и Linux, например, Компоновщик, входящий в интегрированное приложение для работы в Интернете SeaMonkey; Microsoft FrontPage и Adobe Dreamweaver в операционной системе Windows и др.).

Также документы в формате *Web-страниц* могут сохраняться с использованием многих текстовых редакторов (например, вышеперечисленных Microsoft Word и OpenOffice.org Writer).

Настольные издательские системы. Для подготовки к изданию книг, брошюр, журналов, газет и плакатов используются мощные программы обработки текста и графики — настольные издательские системы (например, в операционных системах Windows и Linux свободно распространяемая система Scribus).

Настольные издательские системы позволяют произвольно размещать на странице блоки (фрагменты текста, изображения, таблицы и другие объекты). Если пользователю что-то не нравится, то он легко может переместить блоки или изменить их размеры.

Каждая настольная издательская система может сохранять документы в собственном оригинальном формате (например, система Scribus в формате SLA).

Существует универсальный формат для представления в электронном виде полиграфической продукции. Это формат Portable Document Format (PDF). Значительный ассортимент современного профессионального печатного оборудования может обрабатывать тексты в формате PDF непосредственно. Этот формат позволяет внедрять необходимые шрифты, векторные и растровые изображения, формы и мультимедиа вставки.

Формат PDF сохраняет расположение блоков и их форматирование в различных операционных системах. Более того, формат PDF обеспечивает одинаковое представление документа и при просмотре файла в компьютерной сети, и при выводе документа на печать. Также важно, что стандартными средствами редактировать документы в этом формате невозможно, для этих целей обычно применяют распознаватели текста.

Для просмотра документа в формате PDF можно использовать бесплатную программу Adobe Acrobat Reader, а для создания и редактирования документа в формате PDF требуется лицензионная программа Adobe Acrobat.

Можно также конвертировать документы в формат PDF из настольных издательских систем и текстовых процессоров (такой конвертор встроен в OpenOffice.org Writer, и его можно установить как дополнительный компонент в Microsoft Word 2007).

Вопросы для размышления

1. В чем состоят преимущества и недостатки различных приложений для создания и редактирования документов?
2. Каковы особенности формата документов PDF?

Практическая работа 2.1

Установка конвертора в формат PDF для Microsoft Office 2007

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows XP/Vista, подключенный к Интернету.

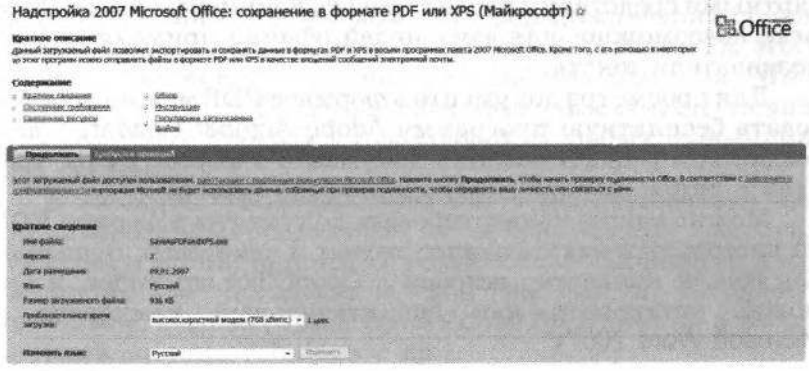
Цель работы. Научиться устанавливать конвертор в формат PDF для Microsoft Office 2007.

Задание. Установка конвертора в формат PDF для Microsoft Office 2007.

В систему Microsoft Office 2007 можно установить бесплатное дополнение для сохранения файлов в формате PDF.

Установка конвертора в формат PDF для Microsoft Office 2007

1. *В операционной системе Windows XP/Vista перейти по* ссылке на страницу *Надстройка Microsoft* для сохранения в формате PDF или XPS файлов, созданных в программах выпуска 2007 системы Microsoft Office.



2. В появившейся в окне браузера Web-странице в раскрывающемся списке установить нужный язык (например, *Русский*) и в другом раскрывающемся списке — скорость соединения с Интернетом (например, *высокоскоростной модем (768 кбит/с)*.
   Щелкнуть по кнопке *Продолжить*.
3. Если приложение Microsoft Office 2007 пройдет проверку на подлинность, то появится кнопка *Загрузить*.
   Щелкнуть по этой кнопке и выбрать место сохранения файла-конвертора SaveAsPDFandXFS.exe.
4. Запустить этот файл на исполнение, конвертор в формат PDF будет установлен.

2.1.1. Макет и верстка в настольных издательских системах

Макет. Чтобы документ смотрелся как единое целое, необходимо разработать макет — графический план верстки. Этот план наглядно показывает, как распределяются на документе текстовые и иллюстративные материалы, заголовочные комплексы, все детали оформления. На макете указываются линейки, маркеры, размер, насыщенность, наклон, плотность и гарнитура шрифта, выравнивание строк и другие компоненты.

Зная количество знаков во всем тексте и в одной строке, площадь, которую займет заголовочный комплекс, можно добиться точности макетирования и избежать потом (при верстке) неприятных сюрпризов, когда образуются свободные места, белые пятна, или «хвост» материала, не вместившегося в запланированное на макете место.

Ускорить макетирование позволяет модульная сетка — система вертикальных и горизонтальных линий, своеобразный каркас, определяющий структуру документа.

Выбор макета зависит от формата страницы, стиля оформления, особенностей издания. Компьютер избавляет от необходимости вычерчивать на бумаге подробные графические макеты. Страницы-шаблоны с модульной сеткой и основными элементами оформления могут храниться в памяти компьютера как угодно долго и при необходимости использоваться. Составив план-макет, на документе располагают (верстают) материалы.

Верстка. Версткой называют как способ расположения материалов при творческом создании документа, так и чисто технологический процесс, при котором из соответствующих блоков монтируются документы заданного размера и, в соответствии с макетом, размещается материал.

Верстка бывает прямой и ломаной, горизонтальной и вертикальной, симметричной и асимметричной. Иллюстрации можно использовать в форме не только прямоугольника, но и круга, овала, многоугольника. Настольные издательские системы позволяют и текстовые материалы (особенно для выделения частей публикации) верстать в виде пирамиды, треугольника, круга и т. д.

Раньше версткой занимался типографский работник, сейчас — редакционный оператор верстки.

В современных газетах и интернет-изданиях, где требуется особо оперативная подача информации, применяется «раздельная верстка», т. е. с места событий журналист передает уже сверстанный материал. Это требует от журналиста высокой технической квалификации: ему надо пользоваться модемом для передачи информации, часто использовать и настольно-издательские системы.

Дизайн печатных изданий. Форма верстки издания должна соответствовать его цели и содержанию. Ясно, что дизайн верстки книг должен отличаться от дизайна верстки журналов, а он, в свою очередь, должен отличаться от дизайна верстки плакатов и рекламы. Особый дизайн должны иметь Web-сайты, фирменные знаки, упаковка и т. д.

Однако существуют и общие рекомендации, которые сводятся к следующему:

- перед версткой документа оценить, какие статьи и фотографии в нем разместить;
- если объем или форма рекламных объявлений не соответствуют установленному дизайну, выяснить, можно ли их перенести в другое место;
- определить объем и формат основного текста и фотографий;
- фотографии помещать посреди текста;
- заголовок основной статьи должен быть на 40% крупнее других заголовков документа;
- следует группировать сходные по содержанию небольшие материалы вместе;

- не следует располагать статьи с заголовками одинаковой ширины друг над другом;
- не надо выстраивать заголовки в одну линию.

Началом книгопечатания считается изобретение гравирования изображений, подлежащих воспроизведению, на деревянной доске, которую затем закатывали краской, покрывали бумагой и получали нужное количество идентичных оттисков.

Рис. 2.1. Книгопечатание в Средние века

Следующим этапом развития книгопечатания явилось изобретение в XI веке подвижных литер (букв и цифр).

Печатный пресс сделал книгопечатание массовым (в Европе в 1440 г. его изобрел И. Гутенберг, а в России его применил в 1563 г. первопечатник И. Федоров).

XX век ознаменовался внедрением в полиграфию электронных машин, фотонабора, автоматизацией многих процессов производства. Конец XX века — это начало цифровых и лазерных технологий на базе компьютерных систем. Изменилась даже суть полиграфии: наблюдается переход от идентичных копий в тираже к многовариантности экземпляров издания в пределах тиража.

Вопросы для размышления

1. Для чего необходимо продумать макет документа перед его версткой?
2. Почему различается дизайн верстки у документов различных типов?

2.1.2. Параметры документа

Способы создания документов. Создание документов можно производить с помощью шаблонов, т. е. пустых заготовок документов определенного назначения. **Шаблон** задает структуру документа, которую пользователь заполняет определенным содержанием. Настольные издательские системы обычно имеют обширные библиотеки шаблонов для создания документов различного назначения (книга, журнал, плакат, Web-сайт и др.).

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

Выбор параметров страницы. Любой документ состоит из страниц, поэтому в начале работы над документом необходимо задать параметры страницы: размер бумаги, ориентацию и размеры полей. Ориентация позволяет выбрать расположение страницы на бумаге. Существуют две возможные ориентации страницы (книжная и альбомная). На странице можно установить требуемые размеры полей (верхнего и нижнего, правого и левого), которые определяют расстояния от краев страницы до границы текста.

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

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

Сноски. В конце страницы или документа можно разместить сноски, которые вносят необходимые разъяснения в текст документа. В тексте документа номера сносок оформляются в виде верхних индексов.

Стили форматирования. При создании многостраничных документов удобнее для каждого типа абзацев использовать определенный стиль форматирования (например, один стиль для текста параграфа, другой стиль для вопросов к параграфу). Каждому стилю форматирования присваивается название, и устанавливаются все необхо-

димые параметры форматирования шрифта, абзаца или списка.

Теперь для изменения параметров форматирования абзацев одного типа достаточно изменить параметры соответствующего стиля форматирования. Все абзацы данного стиля форматирования автоматически получат во всем документе новые параметры форматирования.

Оглавление документа. В процессе создания документа в нем создаются заголовки, для которых используются различные стили форматирования.

После создания объемного документа целесообразно вставить в документ оглавление, которое позволит читателю лучше ориентироваться в его содержании. Оглавление представляет собой список заголовков, содержащихся в документе, с указанием страниц.

Копирование, перемещение и удаление фрагментов документа. Редактирование документа производится путем копирования, перемещения или удаления выделенных символов или целых блоков. Копирование позволяет размножить выделенный фрагмент документа, т. е. вставить его копии в указанные места документа. Перемещение дает возможность вставить копии выделенного фрагмента документа в указанные места документа, но удаляет сам выделенный фрагмент. Удаление позволяет удалить выделенный фрагмент.

Блоки информации. На странице документа информация размещается в блоках (текстовых, графических, таблиц и т. д.). Каждый блок может иметь свой размер, форму, положение на странице и т. д. Можно упорядочить положение блоков путем установки свойства привязки к сетке страницы (сетка не будет отображаться в итоговом документе PDF).

Вопросы для размышления

1. Какие параметры документа можно и нужно задать?
2. Какие разновидности блоков информации можно создать в процессе верстки документа?

2.1.3. Текстовые блоки

Размер и позицию текстового, как и любого другого, блока можно задать или изменить с помощью мыши.

Можно изменить форму текстового блока (квадрат, многоугольник, окружность, овал и т. д.), а также толщину и вид линии границы блока.

Текст в блоке можно разместить как в одну, так и в несколько колонок.

Ввод текста в текстовый блок можно осуществить различными способами:

- ввести текст непосредственно в блок;
- скопировать текст из другого текстового блока;
- ввести текст с использованием встроенного в настольную издательскую систему текстового редактора;
- вести текст из текстового файла.

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

Шрифт — это полный набор символов (букв, цифр, знаков пунктуации, математических знаков, а также специальных символов) определенного рисунка. Для каждого исторического периода и разных стран характерны свои шрифты. Каждый шрифт имеет свое название, например Times New Roman, Arial, Courier New и др.

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

Единицей измерения размера шрифта является *пункт* (1 пт = 0,376 мм). Размеры шрифтов можно изменять в больших пределах (обычно от 1 до 1638 пунктов), причем во многих редакторах по умолчанию используется шрифт размером 10 пт.

Кроме обычного начертания символов может применяться **полужирное**, *курсивное* и ***полужирное курсивное***.

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

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

Буквица (капитель). Буквица широко применялась в старинных книгах в начале абзаца или главы (рис. 2.2). Можно установить врезанную в текст буквицу (она называется капителью) высотой в любое количество строк. Можно ввести букву, которые вы желаете сделать капителью, либо непосредственно, либо с помощью встроенного текстового редактора.

Рис. 2.2. Буквица из книги «Апостол» первопечатника Ивана Федорова

Поиск и замена. В процессе работы над документом иногда бывает необходимо заменить одно многократно встречающееся слово на другое. Если делать это вручную, то процесс замены отнимет много времени и сил. В большинстве текстовых редакторов существует операция *Найти и заменить*, которая обеспечивает автоматический поиск и замену слов во всем документе.

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 абзаца (выключка) отражает расположение текста относительно границ полей страницы. Чаще всего абзац начинается отступом первой строки. Весь абзац целиком может иметь отступы слева и справа, которые отмеряются от границ полей страницы.

Расстояние между строками документа можно изменять, задавая различные значения междустрочных интервалов. Для визуального отделения абзацев друг от друга можно устанавливать увеличенные интервалы до и после абзацев.

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

В нумерованных списках элементы списка последовательно обозначаются с помощью чисел (арабских или римских) и букв (русского или латинского алфавитов). 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.

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

Многоуровневые списки можно использовать для отображения иерархических перечней (например, иерархической файловой системы). 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.

Вопросы для размышления

1. Как можно ввести текст в текстовый блок?
2. Какие параметры форматирования текста можно выделить как самые главные?

2.1.4. Блоки изображений

Размер и позицию блока изображения можно задать или изменить с помощью мыши. Пунктирными линиями обозначаются границы будущего блока. Внутри блока изображения вы видите диагонали внутри (рис. 2.3). Эти диагонали не будут печататься, это традиция печатной промышленности, созданная для того, чтобы отличить блок изображения от блока текста.

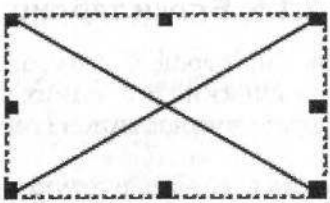


Рис. 2.3. Блок изображения

Можно изменить форму блока изображения (квадрат, многоугольник, окружность, овал и т. д.), а также толщину и вид линии границы блока.

Теперь можно вставить в блок растровое изображение или векторный рисунок. Их можно загрузить из графического файла, а можно нарисовать непосредственно в блоке, используя встроенный в настольную издательскую систему графический редактор. Можно вписать изображение в рамку блока или, наоборот, подогнать рамку блока под выбранный масштаб изображения.

В настольных издательских системах обычно используют несжатые растровые изображения в форматах TIFF или BMP. Они имеют довольно большой объем, но при этом быстро обрабатываются при выводе на экран или принтер и

обеспечивают хорошее качество изображения. Тем не менее можно использовать также сжатые форматы растровой графики JPEG, PNG, GIF и другие, которые имеют существенно меньший информационный объем, но приемлемое качество изображения.

Из векторных форматов чаще всего используется SVG (Scalable Vector Graphics) — открытый формат, разработанный World Wide Web Consortium, и формат WMF (Windows Metafile), предложенный корпорацией Microsoft.

Вопросы для размышления

1. Можно ли подогнать размер изображения под размер блока изображения?
2. Можно ли подогнать размер блока изображения под размер изображения?

2.1.5. Блоки таблиц

Таблицы состоят из строк и столбцов, на пересечении которых образуются ячейки. В ячейках таблиц могут быть размещены данные различных типов (текст, числа, изображения).

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

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 шаблоны, или настроить вручную.

Вопросы для размышления

1. Какие параметры таблиц можно изменять?

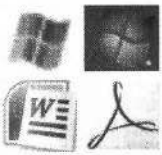
Практическая работа 2.2

Создание плаката в Microsoft Word 2007

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows XP/Vista.

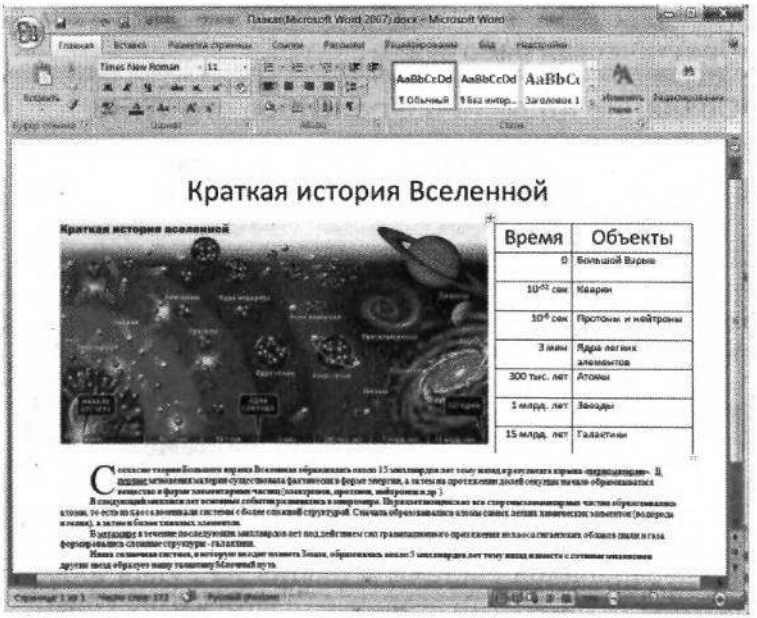
Цель работы. Научиться создавать плакаты в Microsoft Word 2007 и преобразовывать их в формат PDF.

Задание. Создать плакат «Краткая история Вселенной» и преобразовать его в формат PDF.

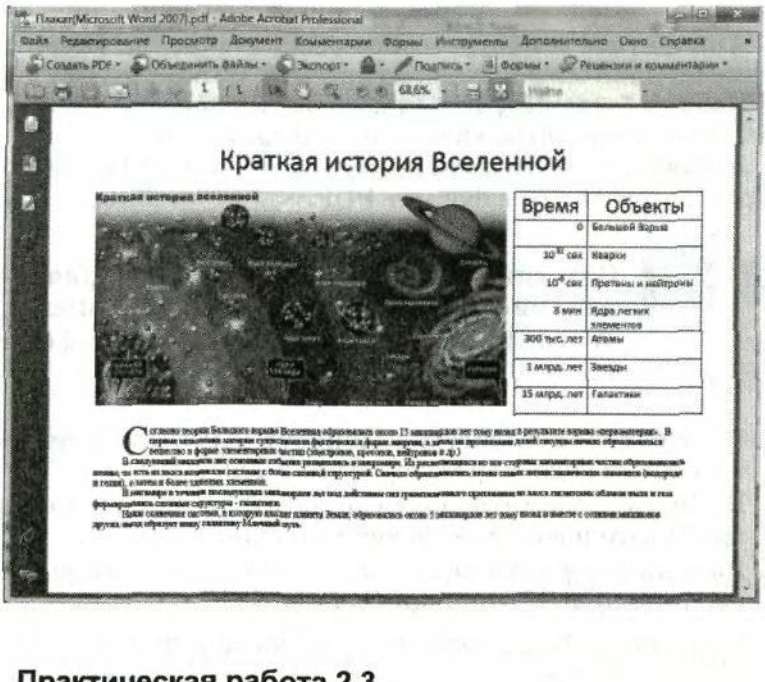


Создание плаката «Краткая история Вселенной» в текстовом редакторе Microsoft Word 2007 и преобразование его в формат PDF

1. В операционной системе Windows XP/Vista запустить текстовый редактор Microsoft Word 2007.
2. Выбрать альбомную ориентацию страницы и поместить на нее заголовок, изображение, таблицу и текст.
3. Выделить первую букву текста и преобразовать ее в буквицу командой [*Вставка-Буквица*].



4. Преобразовать плакат в формат PDF командой [*Сохранить как-PDF*].
Просмотреть его в Adobe Acrobat Professional.



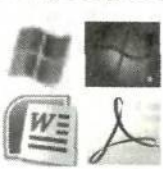
Практическая работа 2.3

Создание плаката в OpenOffice.org Writer

Аппаратное и программное обеспечение. Компьютер с *установленной операционной системой Windows XP/Vista* или Linux.

Цель работы. Научиться создавать плакаты в OpenOffice.org Writer и преобразовывать их в формат PDF.

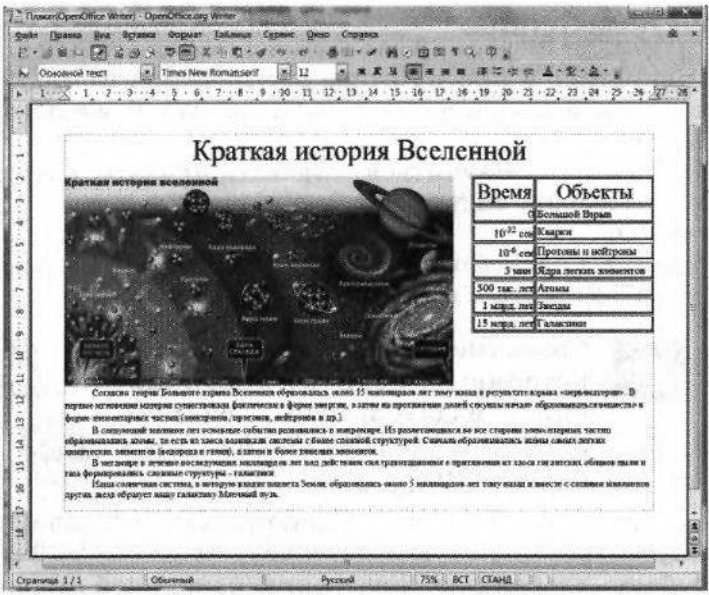
Задание. Создать плакат «Краткая история Вселенной» и преобразовать его в формат PDF.



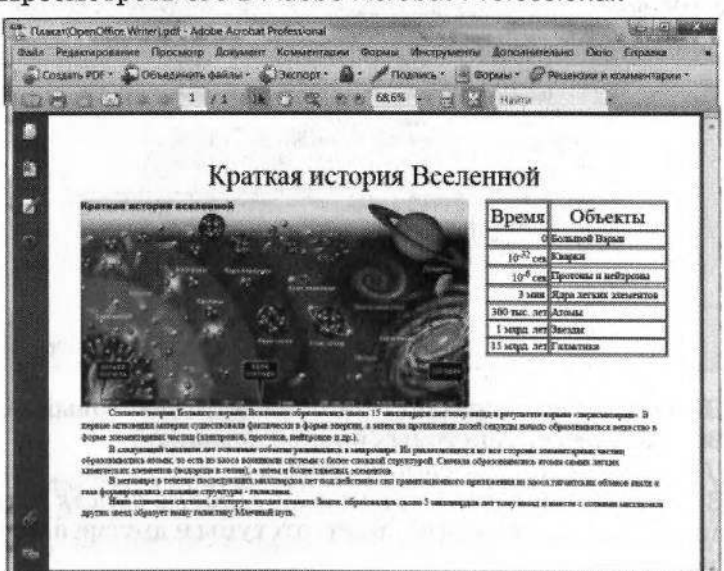
Создание плаката «Краткая история Вселенной» в системе OpenOffice.org Writer и преобразование его в формат PDF

1. В операционной системе Windows XP/Vista или Linux запустить текстовый редактор OpenOffice.org Writer.

2. Выбрать альбомную ориентацию страницы и поместить на нее заголовок, изображение, таблицу и текст.



3. Преобразовать плакат в формат PDF командой [*Сохранить как-PDF*].
Просмотреть его в Adobe Acrobat Professional.



Практическая работа 2.4

Создание плаката в настольной издательской системе Scribus

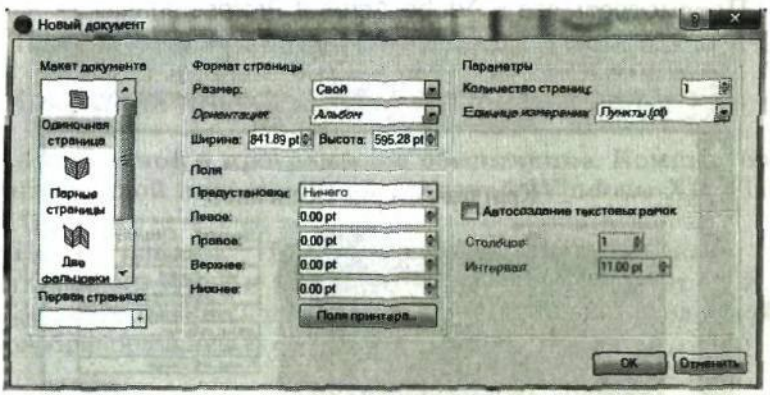
Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows XP/Vista или Linux.

Цель работы. Научиться создавать плакаты в настольной издательской системе Scribus и преобразовывать их в формат PDF.

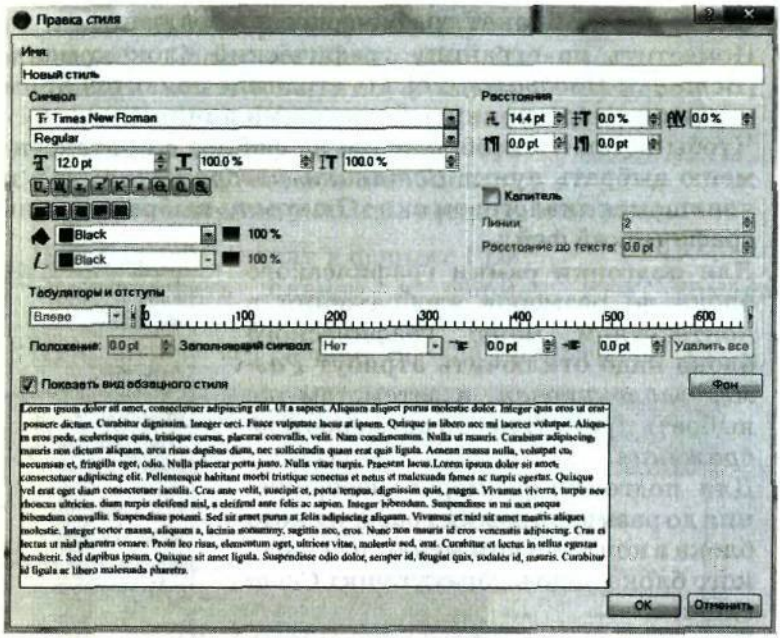
Задание. Создать плакат «Краткая история Вселенной» и преобразовать его в формат PDF.

Создание плаката «Краткая история Вселенной» в настольной издательской системе Scribus и преобразование его в формат PDF

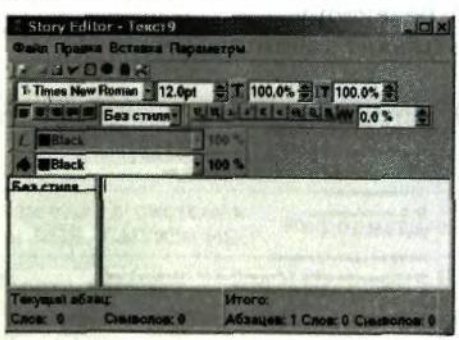
1. В операционной системе Windows XP/Vista или Linux запустить настольную издательскую систему Scribus.
2. Выбрать альбомную ориентацию одиночной страницы, ее размер, поля и т. д.
   Поместим на плакат заголовок и текстовый блок.



3. Поместить на страницу заголовок и текстовый блок. Выбрать стиль текстовых блоков командой [*Правка-Абзацные стили-Новый*].
   В появившемся диалоговом окне *Правка стиля* установить шрифт, его размер, цвет, отступы и другие параметры.



4. Ввести текст в текстовые блоки либо непосредственно, либо из текстового файла, либо скопировать из другого блока, либо с помощью встроенного текстового редактора Story Editor.
Для того чтобы запустить встроенный текстовый редактор Story Editor, в контекстном меню текстового блока выбрать пункт *Изменить текст*.

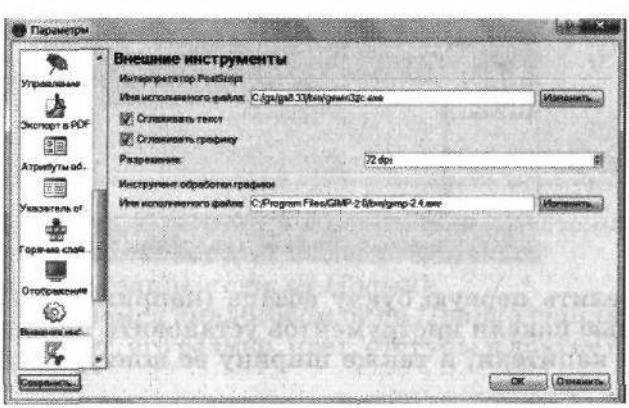


5. Выделить первую букву абзаца (например, С) и с помощью панели инструментов установить ширину и высоту капители, а также ширину ее контура и наличие тени.

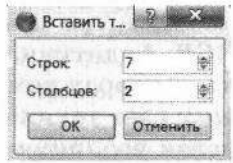
Поместим на плакат графическое изображение.

6. Поместить на страницу графический блок командой [*Вставка-Изображение*]. На странице появится прямоугольник, перечеркнутый красными линиями.
   Чтобы вставить изображение из файла, в контекстном меню выбрать пункт *Вставить изображение...* и в появившемся диалоговом окне *Открыть* выбрать нужный графический файл.
7. Для подгонки рамки графического блока до размеров изображения в контекстном меню графического блока надо отключить атрибут *Размер заблокирован*, а затем там же выбрать пункт *Блок до размеров изображения*.
   Для подгонки размеров изображения до размеров рамки графического блока в контекстном меню графического блока надо выбрать пункт *Свойства*.
   В появившемся диалоговом окне *Свойства* выбрать пункт *Изображение* и установить флажок *Масштабировать до размера*.

8. Ввести команду [*Файл-Настроить Scribus*] и выбрать в меню значков *Внешние инструменты*.
   Ввести пути в файловой системе к интерпретатору PostScript и выбранному для обработки изображений графическому редактору.
   Вставим в плакат таблицу.



9. Поместить на страницу таблицу командой [*Вставка-Таблица*]. На странице появится прямоугольник. В появившемся диалоговом окне *Вставить таблицу* указать требуемое количество строк и столбцов. Вставить в ячейки таблицы нужные значения.



Преобразуем плакат в формат PDF.

10. Преобразовать плакат в формат PDF командой [*Файл-Экпортировать-Сохранить как PDF...*]. Просмотреть его в Adobe Acrobat Professional.

2.1.6. Палитры цветов в системах цветопередачи RGB и CMYK

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

Информатика и ИКТ-9

Человек воспринимает свет с помощью цветовых рецепторов (так называемых колбочек), находящихся на сетчатке глаза. Колбочки чувствительны к красному, зеленому и синему цветам. 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 аддитивной системе цветопередачи RGB. Аддитивная система цветопередачи RGB применяется в мониторах компьютеров, в телевизорах и других *излучающих* свет технических устройствах. 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**, которые могут иметь различную интенсивность. Цвет палитры *Color* можно определить с помощью формулы (2.1). При этом надо учитывать глубину цвета — количество битов, отводимое в компьютере для кодирования цвета. Глубина цвета в 24 бита означает, что на кодирование каждого из трех 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.

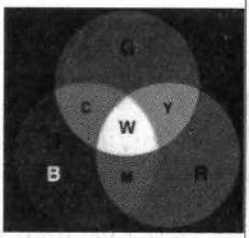
$$Color = R + G + B,$$
для глубины цвета 24 бита (2.1)
$$0 \le R < 256,\ 0 \le G < 256,\ 0 \le B < 256$$

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*), красного и зеленого — желтый цвет (*Yellow*), красного и синего — пурпурный цвет (*Magenta*) (табл. 2.1).

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

Таблица 2.1. **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



Палитра цветов в субстрактивной системе цветопередачи CMYK. Субстрактивная система цветопередачи CMYK применяется в полиграфии. 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.

Окружающие нас предметы частично поглощают свет, а частично отражают. Белый листок бумаги выглядит белым потому, что он отражает все цвета спектра (составляющие белый свет) и ни один не поглощает. Если осветить белый листок синим светом, бумага будет выглядеть синей. При освещении белым светом листа зеленой бумаги, бумага будет выглядеть зеленой, так как она поглощает все цвета, кроме зеленого. Если осветить красную бумагу синим светом, то она будет выглядеть черной, потому что синий свет она не отражает, и направленный на нее свет полностью поглощается.

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 (2.2), в которой интенсивность каждой краски задается в процентах.

$$Color = C + M + Y,$$
$$\text{где } 0\% \le C \le 100\%,\ 0\% \le M \le 100\%,\ 0\% \le Y \le 100\% \quad (2.2)$$

Цвета в палитре CMY формируются путем **вычитания из белого цвета 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.

Смешивая попарно краски системы CMY, мы получим базовые цвета 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.

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 название черного цвета *blacK*, т. е. K. Расширенная палитра получила название CMYK.

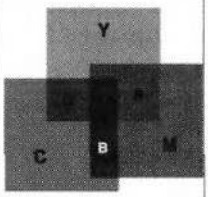
В струйных принтерах для получения изображений высокого качества используются четыре картриджа, содержащие базовые краски системы цветопередачи CMYK.

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

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.

Таблица 2.2. **Формирование цветов в системе цветопередачи CMYK**

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



Вопросы для размышления

1. Чем различаются и где применяются системы цветопередачи RGB и CMYK?

2.1.7. Цветоделение в полиграфии

Многокрасочная печать. Цветной оригинал труднее воспроизвести, чем штриховой и полутоновый, так как для этого требуется цветоделение. Цвета аддитивного смешения — синий, зеленый и красный — образуются при наложении друг на друга соответственно голубого и пурпурного, голубого и желтого, пурпурного и желтого цветов. Чтобы точно воспроизвести требуемый цвет, например зеленый или оранжевый, нужно точно воспроизвести соотношение в нем трех цветовых составляющих — желтого, голубого и пурпурного.

Цветоделение в современной полиграфии — процесс подготовки цветных изображений к печати несколькими красками. Данная технология использует принцип субтрактивного синтеза цвета, предполагающий, что на материал, отражающий или пропускающий свет (например, бумагу или прозрачную пленку), наносятся слои цветных красителей, каждый из которых «вычитает» из белого цвета свою долю спектра.

Традиционно цветоделение осуществлялось в типографиях с помощью оптических фильтров, в настоящее время процесс автоматизирован и реализован программно в различных компьютерных системах, предназначенных для верстки или обработки изображений для печати, в частности в издательских пакетах и системе обработки документов в формате PDF (например, Adobe Acrobat Professional).

Вопросы для размышления

1. Какая система цветопередачи используется в полиграфии?

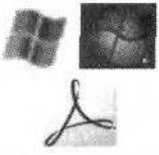
Практическая работа 2.5

Цветоделение

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows XP/Vista.

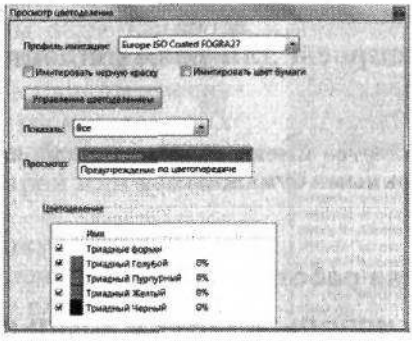
Цель работы. Научиться создавать цветоделение в формате PDF в Adobe Acrobat Professional.

Задание. Реализовать цветоделение плаката «Краткая история Вселенной» в формате PDF.



Цветоделение плаката «Краткая история Вселенной» в формате PDF

1. В операционной системе Windows XP/Vista запустить редактор PDF формата Adobe Acrobat Professional и открыть в нем плакат «Краткая история Вселенной» в формате PDF.
2. Ввести команду
   [*Дополнительно-Допечатная подготовка-Просмотр цветоделения*].
   В появившемся диалоговом окне *Просмотр цветоделения* последовательно оставить флажок только у одного субтрактивного цвета (голубого, пурпурного, желтого, черного).



Просмотреть результат цветоделения.

2.2. Компьютерные языковые словари

Существуют разные виды компьютерных языковых словарей: толковые, семантические, переводные и другие. Переводные компьютерные словари вляются многоязычными, так как дают возможность пользователю выбрать языки (например, современный словарь ABBYY Lingvo содержит 10 языков) и направление перевода (например, англо-русский, испанско-русский и т. д.).

Компьютерные словари позволяют также осуществлять быстрый поиск нужных слов, часто с учетом морфологических форм и с возможностью поиска сочетаний слов (примеров употребления).

Кроме основного словаря общеупотребительных слов компьютерные словари могут содержать десятки специализированных словарей по областям знаний (например, современный словарь ABBYY Lingvo содержит 128 словарей по разным тематикам: технике, медицине, информатике и др.).

Также компьютерные словари могут являться мультимедийными, т. е. предоставлять пользователю возможность прослушивания слов в исполнении дикторов, носителей языка.

Вопросы для размышления

1. Какие преимущества имеют компьютерные языковые словари перед традиционными бумажными?

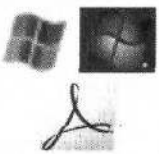
Практическая работа 2.6

Перевод с использованием компьютерных словарей

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows XP/Vista или Linux.

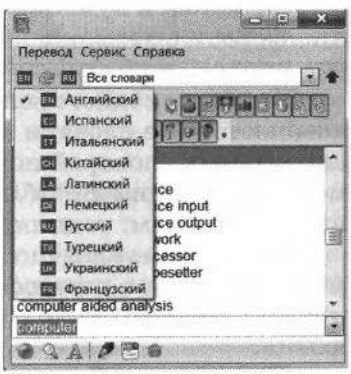
Цель работы. Научиться переводить слова и словосочетания с использованием компьютерного словаря ABBYY Lingvo в операционной системе Windows и компьютерного словаря StarDict в операционной системе Linux.

Задание. Реализовать перевод слов и словосочетаний с использованием компьютерного словаря ABBYY Lingvo в операционной системе Windows и компьютерного словаря StarDict в операционной системе Linux.



Перевод с использованием компьютерного словаря ABBYY Lingvo

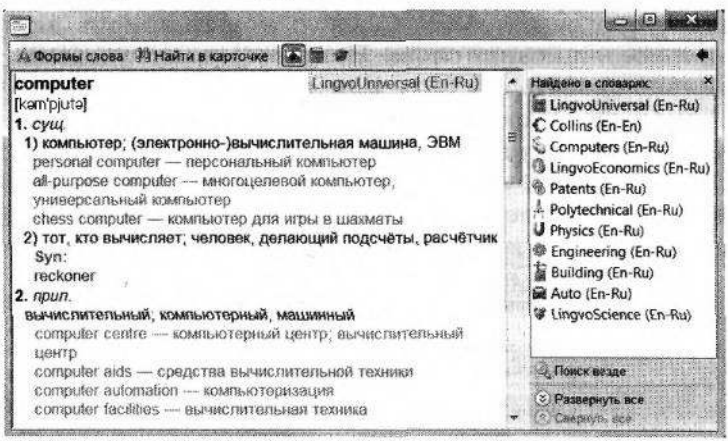
1. В операционной системе Windows XP/Vista запустить компьютерный словарь ABBYY Lingvo.
2. В появившемся диалоговом окне *Lingvo* с использованием соответствующих кнопок панели инструментов выбрать языки и направление перевода (например, английский-русский).



3. В диалоговом окне *Lingvo* с использованием соответствующих кнопок панели инструментов или раскрывающихся списков выбрать словарь, группу словарей или все словари.
Ввести в строку ввода или выбрать из списка английское слово или словосочетание (например, computer).

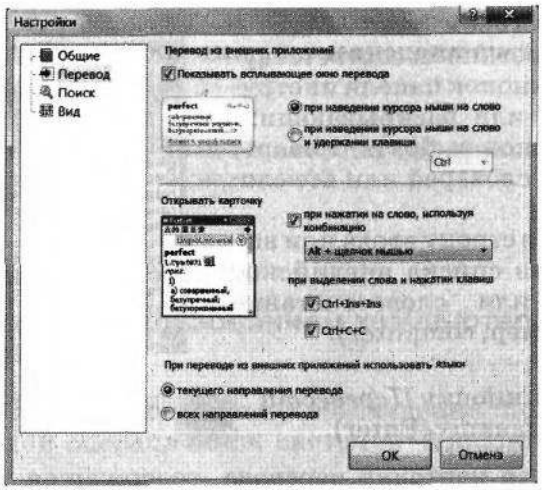
4. Нажать кнопку *Перевести*  или на клавиатуре нажать клавишу {Enter}.

5. Появится карточка перевода, содержащая транскрипцию, варианты перевода и тематические словари, где был найден перевод данного слова.



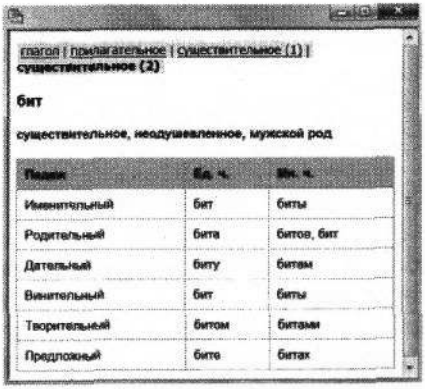
ABBYY Lingvo существенно сократит время на поиск перевода при работе во внешних приложениях. С ABBYY Lingvo можно переводить незнакомые слова или словосочетания в иностранном тексте, не отрываясь от чтения и не переключаясь в другое приложение. В результате перевода можно получить всплывающее окно перевода или карточку с переводом.

6. В диалоговом окне *Lingvo* ввести команду [*Сервис-Настройки*].
В появившемся диалоговом окне *Настройки* выбрать пункт *Перевод* и установить соответствующие флажки.



С помощью электронного словаря компании ABBYY можно просмотреть грамматические формы слова. Таким образом, программа ABBYY Lingvo может помочь выбрать правильное написание слова. Например, вы не уверены, как правильно сказать: «5 бит» или «5 битов».

7. В диалоговом окне *Lingvo* в строку ввода ввести «бит» и нажать кнопку *Показать формы слова* .
Данный словарь показывает, что можно сказать и «бит», и «битов».

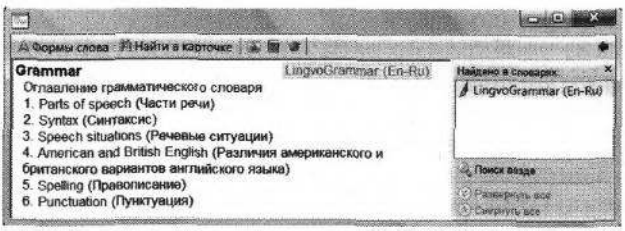


Падеж	Ед. ч.	Мн. ч.
Именительный	бит	биты
Родительный	бита	битов, бит
Дательный	биту	битам
Винительный	бит	биты
Творительный	битом	битами
Предложный	бите	битах

В состав системных словарей ABBYY Lingvo входит грамматический словарь, являющийся гипертекстовым справочником по грамматике английского языка. Грамматический словарь английского языка (LingvoGrammar (En-Ru)) содержит около 500 грамматических тем (статей), инфор-

мацию о нормах речевого этикета и о различиях между британским и американским вариантами английского языка (в частности, о различиях в произношении некоторых слов и т. д.).

8. Чтобы открыть LingvoGrammar, наберите в строке ввода слово *Grammar* и нажмите клавишу {Enter}. В результате откроется карточка, содержащая оглавление грамматического словаря.



Перевод с использованием компьютерного словаря StarDict

1. В операционной системе Linux запустить компьютерный словарь StarDict.
2. В диалоговом окне словаря *StarDict* ввести в строку ввода или выбрать из списка (нажать кнопку *Список*) английское слово или словосочетание (например, computer). На клавиатуре нажать клавишу {*Enter*}.
3. Появится карточка перевода, в правой части содержащая варианты перевода и тематические словари, где был найден перевод данного слова, а в левой части — перечень используемых словарей.

2.3. Системы оптического распознавания символов

При создании электронных библиотек и архивов путем перевода книг и документов в цифровой формат, при переходе предприятий от бумажного документооборота к электронному, при необходимости отредактировать полученный по факсу документ используются **системы оптического распознавания символов.**

Сначала необходимо распознать структуру размещения текста на странице: выделить колонки, таблицы, изображения и т. д. Далее выделенные текстовые фрагменты графического изображения страницы необходимо преобразовать в текст. Текст, преобразованный из графической формы в символьную (текстовую), можно далее обрабатывать любыми текстовыми редакторами. Системы оптического распознавания символов экспортируют результаты распознавания в популярные офисные приложения (Microsoft Office, OpenOffice.org и др.), причем распознанный текст можно сохранить в различных текстовых форматах: DOCX, DOC, ODT, RTF, TXT, HTML и др.

Оптическое распознавание символов (англ. *Optical Character Recognition,* ***OCR***) — электронное конвертирование изображений символов и букв в текст, редактируемый на компьютере. Перевод осуществляется программным путем, после получения изображения со сканера или фото.

Алгоритмы оптического распознавания. Если исходный документ имеет типографское качество (достаточно крупный шрифт, отсутствие плохо напечатанных символов или исправлений), то распознавание осуществляется методом сравнения с **шаблонами символов.** Сначала растровое изображение страницы разделяется на изображения отдельных символов. Затем каждый из них последовательно накладывается на шаблоны символов, имеющихся в памяти системы, и выбирается шаблон с наименьшим количеством точек, отличных от входного изображения.

При распознавании документов с низким качеством печати (машинописный текст, факс и т. д.) используется метод распознавания символов по наличию в них определенных

структурных элементов (отрезков, колец, дуг и др.). Любой символ можно описать через набор параметров, определяющих взаимное расположение его элементов. Например, буква Н и буква И состоят из трех отрезков, два из которых расположены параллельно друг другу, а третий соединяет эти отрезки. Различие между буквами состоит в величине углов, которые образует третий отрезок с двумя другими. При распознавании структурным методом в искаженном символьном изображении выделяются характерные детали и сравниваются со структурными шаблонами символов. В результате выбирается тот символ, для которого совокупность всех структурных элементов и их расположение больше всего соответствует распознаваемому символу.

Наиболее распространенные системы оптического распознавания символов используют как растровый, так и структурный метод распознавания. Кроме того, эти системы являются «обучающимися» (для каждого конкретного документа они создают соответствующий набор шаблонов символов), и поэтому скорость и качество распознавания многостраничного документа постепенно возрастают.

Оптическое распознавание документов. Интеллектуальные системы оптического распознавания позволяют быстро и точно переводить бумажные документы, цифровые фотографии документов и PDF-файлы в электронный вид. При распознавании они полностью сохраняют оформление документа: иллюстрации, картинки, списки, таблицы и т. д. Полученные результаты можно исправлять в текстовых редакторах, сохранять в разных форматах, отправлять по электронной почте и публиковать в Интернете (рис. 2.4).

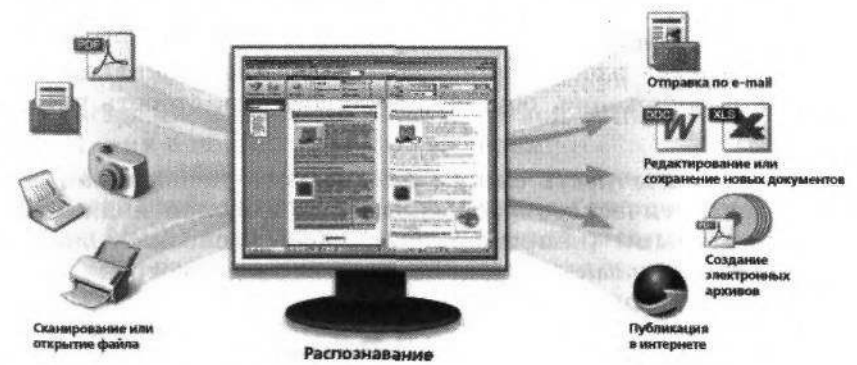


Рис. 2.4. Оптическое распознавание символов

Анализ и обработка документа целиком, а не постранично, позволяют распознать такие элементы его внутренней структуры, как верхние и нижние колонтитулы, сноски, гиперссылки, подписи к картинкам и диаграммам, стили, шрифты и т. д. Таким образом, система оптического распознавания точно распознает и максимально полно сохраняет исходное оформление любого документа.

Оптическое распознавание изображений. Системы оптического распознавания символов работают со всеми популярными моделями сканеров, а теперь для распознавания необязательно оснащать компьютер сканером, так как современные системы позволяют распознавать фотографии документов, сделанные цифровой камерой. Существует множество случаев, когда для получения изображения удобнее использовать фотоаппарат, нежели сканер. Например, во время деловой встречи вне офиса, при распознавании вывесок или объявлений, в библиотеке, особенно при работе с толстыми или старинными книгами. Не говоря уже о том, что цифровой фотоаппарат работает в несколько раз быстрее любого сканера.

Системы оптического распознавания символов работают с большим количеством графических файлов распространенных форматов: PDF, BMP, JPEG, TIFF, PNG и других. Для сканирования большого количества страниц в программах предусмотрен специальный режим, позволяющий работать как с автоподатчиком сканера, так и без него.

Системы оптического распознавания символов позволяют даже предварительно обработать изображения, чтобы повысить качество распознавания и упростить дальнейшую работу с документом. Программы могут очистить изображение от «мусора», устранить перекосы и искажение строк, инвертировать изображение, повернуть или зеркально отразить изображение, обрезать изображение, стереть часть изображения.

Мультиязычность систем оптического распознавания. Системы оптического распознавания символов являются многоязычными (например, FineReader распознает документы на 184 языках, а для 38 языков предусмотрена проверка орфографии).

Системы оптического распознавания форм. При заполнении налоговых деклараций, при проведении переписей *населения и т. д. используются различного вида бланки с*

полями. Рукопечатные тексты (данные вводятся в поля печатными буквами от руки) распознаются с помощью систем оптического распознавания форм и вносятся в компьютерные базы данных.

Сложность состоит в том, что необходимо распознавать написанные от руки символы, довольно сильно различающиеся у разных людей. Кроме того, система должна определить, к какому полю относится распознаваемый текст.

Системы распознавания рукописного текста. 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.

Вопросы для размышления

1. Какие алгоритмы оптического распознавания символов используются?
2. Какие форматы графических файлов документов чаще всего используются на входе систем оптического распознавания, и какие форматы текстовых файлов документов могут существовать на их выходе?

Практическая работа 2.7

Оптическое распознавание документов в формате изображений

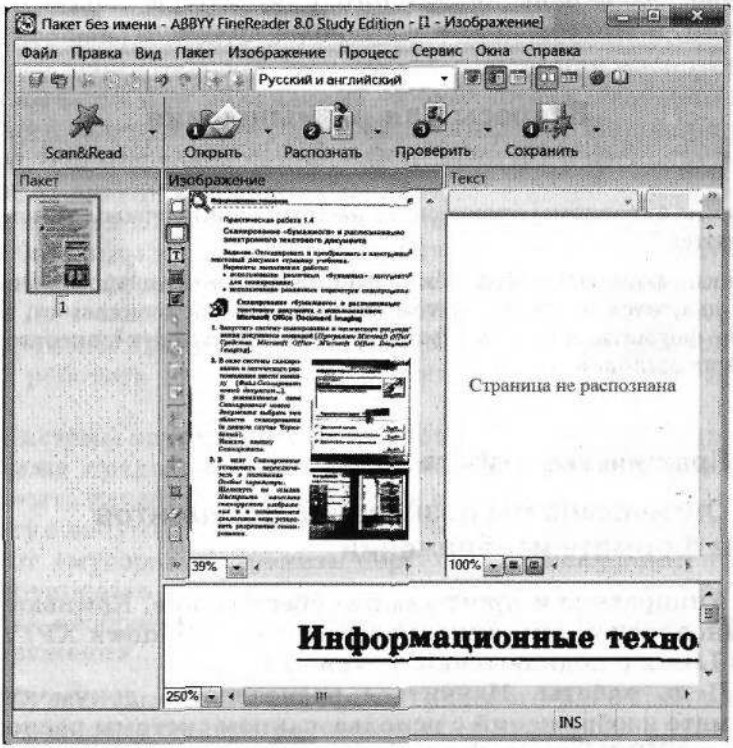
Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows XP/Vista или Linux с подключенным сканером.

Цель работы. Научиться распознавать документы в формате изображений с использованием системы распознавания ABBYY FineReader в операционной системе Windows и системы распознавания Kooka в операционной системе Linux.

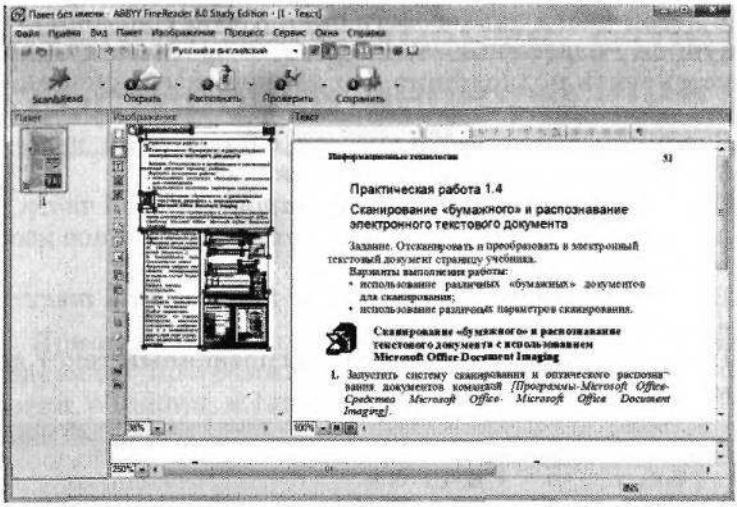
Задание. Распознать документы в формате изображений с использованием системы распознавания ABBYY FineReader в операционной системе Windows и системы распознавания Kooka в операционной системе Linux.

Распознавание с использованием системы распознавания ABBYY FineFeader

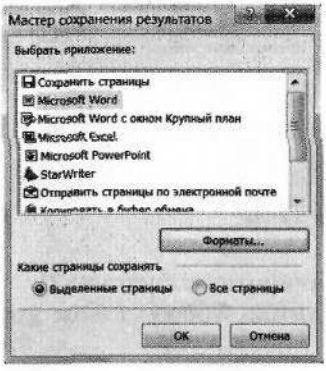
1. В операционной системе Windows XP/Vista запустить систему распознавания ABBYY FineReader.
2. В появившемся диалоговом окне *FineReader* получить графическое изображение документа путем сканирования (щелкнуть по кнопке *Сканировать*), либо путем открытия готового графического файла (щелкнуть по кнопке *Открыть*).

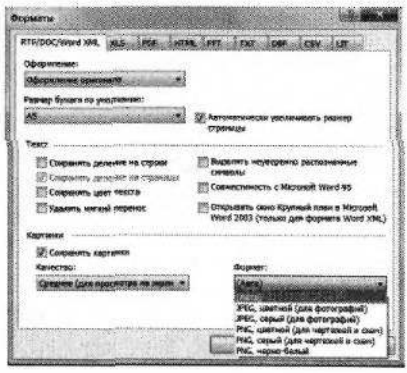


3. Для преобразования графического изображения документа в текстовый формат щелкнуть по кнопке *Распознать*.
   В правой части окна *FineReader* появится текстовый документ.



4. Для сохранения отсканированного и распознанного текстового документа в определенном приложении выбрать из списка это приложение (например, *Microsoft Word*). Для выбора формата сохраняемого текстового документа щелкнуть по кнопке *Форматы* и в появившемся диалоговом окне *Форматы* выбрать формат.

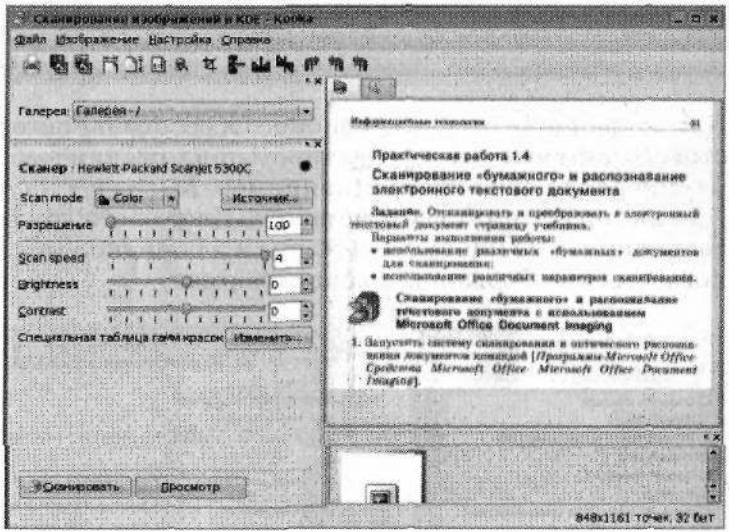




Распознавание с использованием системы распознавания Kooka

1. В операционной системе Linux запустить систему распознавания Kooka.

2. В левой части диалогового окна *Kooka* с помощью ползунков *Разрешение*, *Scan speed*, *Brightness* и *Contrast* установить подходящие для данного изображения текста значения.
С помощью раскрывающегося списка *Scan mode* выбрать нужный тип сканирования.
Щелкнуть по кнопке *Сканировать*, в правой части диалогового окна *Kooka* появится отсканированное изображение документа.
Ввести команду [*Изображение-Распознать текст из всего изображения...*].
Появится окно, содержащее распознанный текст документа.



Глава 3

Технология хранения, отбора и сортировки информации

При изучении данной главы рекомендуется установить следующее программное обеспечение для операционных систем Windows и Linux:

- Microsoft Office 2007 (Access 2007).;
- Редактор реестра;
- MyHeritage Family Tree Builder

Первая помощь ПО 1.0.

CD-5

Операционная система Windows

Windows-CD

3.1. Базы данных

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

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

База данных является информационной моделью организации данных предметной области.

Базы данных можно классифицировать по различным признакам. Рассмотрим их классификацию по используемой **модели данных**. Принято выделять реляционные, иерархические и сетевые модели данных. В настоящее время реляционные БД являются доминирующими, поэтому о них в основном и пойдет речь в учебнике.

Реляционные базы данных. Реляционная модель базы данных была предложена в 1969 г. математиком, научным сотрудником фирмы IBM Э. Ф. Коддом.

Слово «реляционная» происходит от английского «relation» — отношение. Это строгое математическое понятие, относящееся к теории множеств. Для пользователя базы данных отношения удобно представлять в виде неупорядоченных таблиц. Таблицы состоят из столбцов и строк и содержат данные.

Строка таблицы соответствует некоторому объекту моделируемой предметной области. Строки таблицы называются **записями**. Записи разбиты на поля столбцами таблицы. Столбец таблицы описывает некоторый атрибут объектов и содержит значения этого атрибута. Столбцы таблицы называются **полями**. Таким образом, каждая запись представляет собой набор значений атрибутов данного объекта; значение каждого атрибута — в своем столбце. Каждое поле характеризуется своим именем (именем соответствующего атрибута) и типом данных, представляющих собой значения данного атрибута.

Поле базы данных — это столбец таблицы, соответствующий некоторому атрибуту объектов предметной области, содержащий значения этого атрибута.

Запись базы данных — это строка таблицы, соответствующая некоторому объекту моделируемой предметной области, содержащая набор значений атрибута этого объекта.

Каждая строка в таблице должна быть уникальной, т. е. иметь значения полей, отличающие ее от всех других строк. Набор полей, значения в которых уникальны для каждой строки, называется **первичным ключом**. Каждая таблица

должна иметь первичный ключ, что позволяет однозначно идентифицировать каждую запись в таблице. Если такой ключ не объявлен, то им считаются все поля таблицы в совокупности.

Первичный ключ — это поле (группа полей), значение которого однозначно определяет запись в таблице.

В качестве первичного ключа можно использовать поле, содержащее данные типа *счетчик*, а также другие поля таблицы: код товара, инвентарный номер и т. п.

Тип поля определяется типом данных, которые оно содержит. Поля могут содержать данные следующих основных типов:

- *Текстовый*. Содержит до 255 символов.
- *Числовой*. Число.
- *Счетчик*. Вид числового типа. Последовательность целых чисел, которые задаются автоматически при вводе записей. Эти числа не могут быть изменены пользователем.
- *Денежный*. Вид числового типа. Число в денежном формате.
- *Дата/Время*. Дата и/или время.
- *Логический*. Значение *Истина (Да)* или *Ложь (Нет)*.
- *Гиперссылка* (например, в интерфейсе БД Microsoft Access). Ссылка на информационный ресурс в Интернете (например, Web-сайт).

Поле каждого типа имеет свой набор свойств. Наиболее важными свойствами полей являются:

- *Размер поля*. Определяет максимальную длину текстового или числового поля.
- *Формат поля*. Устанавливает формат данных.
- *Непустое поле*. Указывает на то, что данное поле обязательно надо заполнить.

Рассмотрим, например, базу данных «Процессоры», которая содержит перечень объектов (процессоров). Для описания свойств в базу данных можно включить следующие поля различных типов: *№ п/п* (счетчик), *Название процессора* (текстовое поле), *Частота* (числовое поле), *Год выпуска* (поле даты), *Наличие нескольких ядер* (логическое поле) и *Сайт производителя* (гиперссылка) (табл. 3.1).

Таблица 3.1. **Реляционная база данных, представленная в виде таблицы**

№ п/п	Название процессора	Частота, МГц	Год выпуска	Наличие нескольких ядер	Сайт производителя
1	Intel Pentium	266	1993	Нет	www.intel.com
2	AMD Duron	1300	1999	Нет	www.amd.com
3	Intel Pentium 4	3200	2000	Нет	www.intel.com
4	AMD Antlon X2	3200	2005	Да	www.amd.com
5	Intel Core 2 Quad	2900	2008	Да	www.intel.com

Иерархическая модель данных. Иерархическая модель данных графически может быть представлена как перевернутое дерево, состоящее из объектов различных уровней. Верхний уровень (корень) занимает один объект, второй — объекты второго уровня и т. д.

Между объектами существуют связи, каждый объект может быть связан с несколькими объектами более низкого уровня. Такие объекты находятся в отношении **предка** (объект, более близкий к корню) к **потомку** (объект более низкого уровня), при этом объект-предок может не иметь потомков или иметь их несколько, тогда как объект-потомок обязательно имеет только одного предка. Объекты, имеющие общего предка, называются **близнецами**.

Базой данных, основанной на иерархической модели, является *Реестр Windows*, в котором хранится вся информация, необходимая для нормального функционирования компьютерной системы (данные о конфигурации компьютера и установленных драйверах, сведения об установленных программах, настройки графического интерфейса и др.). Содержание реестра автоматически обновляется при установке нового оборудования, инсталляции программ и т. д.

Сетевая модель данных. Сетевая модель данных является обобщением иерархической за счет допущения объектов, имеющих более одного предка, т. е. каждый элемент вышестоящего уровня может быть связан одновременно с любыми элементами следующего уровня. Вообще, на связи между объектами в сетевых моделях не накладывается никаких ограничений. Примером сетевой модели данных является, например, генеалогическое древо семьи.

Вопросы для размышления

1. В чем заключается разница между записью и полем в реляционной базе данных?
2. Поля каких типов могут присутствовать в базе данных?
3. Чем отличается первичный ключ от остальных полей базы данных?
4. В чем состоит главное отличие иерархической модели данных от сетевой модели данных?

3.2. Системы управления базами данных

Системы управления базами данных (СУБД). Создание баз данных, а также обработка данных (в том числе операции отбора и сортировки данных) выполняются специальными комплексами программ — **системами управления базами данных (СУБД)**. Таким образом, необходимо различать собственно базы данных (БД), которые являются определенным образом организованными наборами данных, и системы управления базами данных (СУБД) — программы, управляющие хранением и обработкой данных.

Система управления базами данных (СУБД) — это комплекс программ, позволяющий создавать базы данных, а также обеспечивающий обработку данных (в том числе дополнение, отбор, модификацию данных).

Реляционные СУБД. В качестве примеров СУБД, использующих реляционную модель данных, можно привести в операционной системе Windows приложение Access, входящее в Microsoft Office, а в операционной системе Linux — систему управления базами данных Knoda.

Таблица. Как было сказано выше, в реляционных базах данных вся информация представлена в виде таблиц. Это базовый объект БД, все остальные объекты (запросы, формы, отчеты и пр.) создаются на основе существующих таблиц (производные объекты).

Запросы. В реляционных СУБД запросы являются важным инструментом для пользователя. Главное предназначение запросов — это отбор данных, удовлетворяющих определенным условиям, в отсортированном (упорядоченном) по желанию пользователя виде.

Формы. Формы позволяют отображать данные, содержащиеся в таблицах или запросах, в более удобном для восприятия виде. При помощи форм можно добавлять в таблицы новые данные, а также редактировать или удалять существующие. Форма может содержать рисунки, графики и другие внедренные объекты.

Отчеты. Они предназначены для печати данных, содержащихся в таблицах и запросах, в красиво оформленном виде.

Иерархические и сетевые СУБД. Это СУБД, использующие, соответственно, иерархическую и сетевую модели данных. В иерархических и сетевых СУБД могут существовать все вышеперечисленные формы представления данных.

Вопросы для размышления

1. Какие функции выполняют различные объекты (таблицы, формы, запросы, отчеты и макросы) в реляционной базе данных?

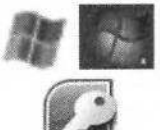
Практическая работа 3.1

Создание реляционной базы данных

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows XP/Vista.

Цель работы. Научиться создавать реляционные базы данных в СУБД Microsoft Access 2007 в операционной системе Windows.

Задание. Создать реляционную базу данных «Процессоры» (см. табл. 3.1) в операционной системе Windows с помощью СУБД Microsoft Access 2007.



Создание реляционной базы данных с помощью СУБД Microsoft Access 2007

1. В операционной системе Windows XP/Vista запустить СУБД Microsoft Access 2007.

Прежде всего необходимо определить структуру базы данных, т. е. определить количество полей, их названия и типы данных, в них хранящихся. База данных «Процессоры» будет содержать следующие поля:

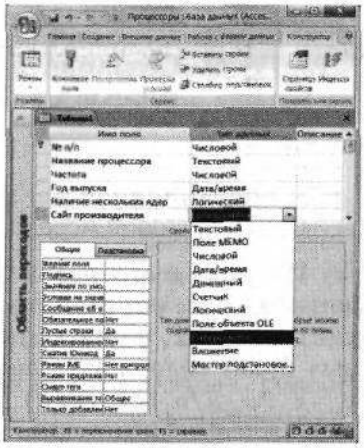
- *№ п/п* (счетчик) — первичный ключ, однозначно идентифицирующий запись.
- *Название процессора* (текстовый) — содержит название процессора.
- *Частота* (числовой) — содержит частоту процессора.
- *Год выпуска* (дата) — содержит год выпуска процессора.
- *Наличие нескольких ядер* (логический) — принимает значение *Да* (два или четыре ядра) или значение *Нет* (одно ядро).
- *Сайт производителя* (гиперссылка) — содержит ссылку на сайт производителя процессора в Интернете.

2. Щелкнуть по значку *Кнопка Microsoft Office*  .

В появившемся диалоговом окне щелкнуть по кнопке *Параметры Access*.
В появившемся диалоговом окне выбрать пункт *Основные*.
В текстовом поле *Рабочий каталог:* ввести путь к создаваемой базе данных.
Щелкнуть по кнопке *ОК*.

3. Щелкнуть по значку *Кнопка Microsoft Office*  .
Создать в СУБД Microsoft Access 2007 новую базу данных с помощью команды [*Создать*].
В появившемся диалоговом окне в текстовом поле *Имя файла:* присвоить файлу базы данных имя Процессоры.accdb.
Щелкнуть по кнопке *Создать*.

4. В окне *Процессоры: база данных* ввести команду [*Режим-Конструктор*].
Ввести имена полей базы данных в столбце *Имя поля*.
В столбце *Тип данных* с помощью раскрывающегося списка для каждого поля установить требуемый тип данных.
Если это необходимо, в нижней части окна внести коррективы в свойства поля (в окне *Свойства поля*).



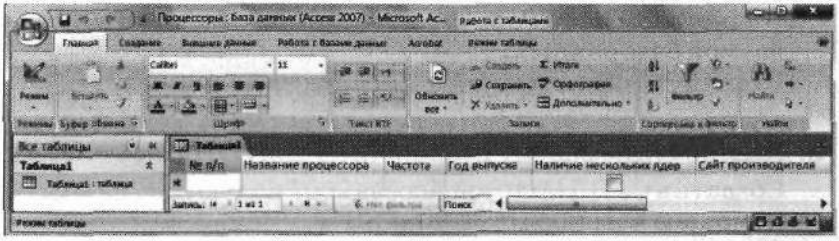
Щелкнуть по кнопке *Ключевое поле* и задать первичный ключ, которым в данном случае является поле *№ п/п*.

5. Для сохранения таблицы ввести команду [*Сохранить*].

6. После создания таблицы ее имя добавляется в окно базы данных, и таблицу можно легко открыть либо в режиме *Конструктор* (команда [*Режим-Конструктор*]), либо в режиме *Таблица* (команда [*Режим-Режим таблица*]).

Режим *Таблица* позволяет просматривать и изменять структуру таблицы, а также вводить и редактировать данные.

7. Для просмотра структуры таблицы в окне базы данных «Процессоры» на вкладке *Таблицы* дважды щелкнуть на значке *Таблица1*. Появится окно таблицы:



Ввод данных в таблицу базы данных и их редактирование мало чем отличается от аналогичных действий в других офисных приложениях.

При вводе данных в режиме *Таблица* в поле маркера записи, которое расположено слева от полей таблицы, может отображаться один из следующих символов:

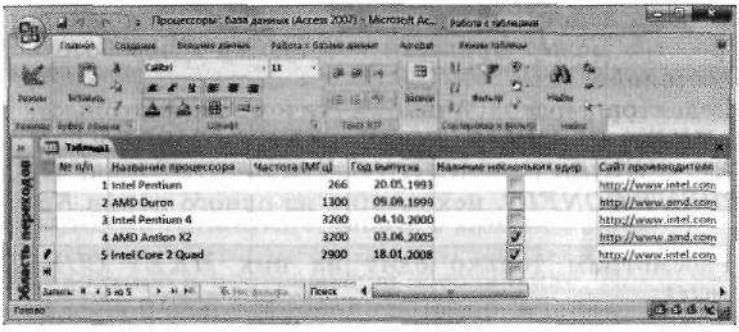
* (звездочка) обозначает пустую запись в конце таблицы;

▸ (стрелка) обозначает выделенную (активную) запись;

✎ (карандаш) обозначает, что в записи были сделаны изменения.

Введем в таблицу данные.

8. Заполнить базу данных, последовательно вводя записи о процессорах:



Перемещение между записями можно осуществлять с помощью мыши, клавиш управления курсором или полосы прокрутки. *Для быстрого перемещения между записями в* базе данных можно использовать кнопки перемещения на панели *Запись*, которая находится в нижней части окна таблицы.

Реляционная база данных «Процессоры» хранится в папке ..\IIKT11prof **Windows-CD**

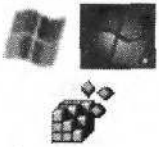
Практическая работа 3.2

Редактирование системного реестра Windows

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой WindowsXP/Vista.

Цель работы. Получить представление о редактировании базы данных, основанной на иерархической модели данных, — системного реестра Windows.

Задание. В системном реестре Windows изменить размер системного шрифта.



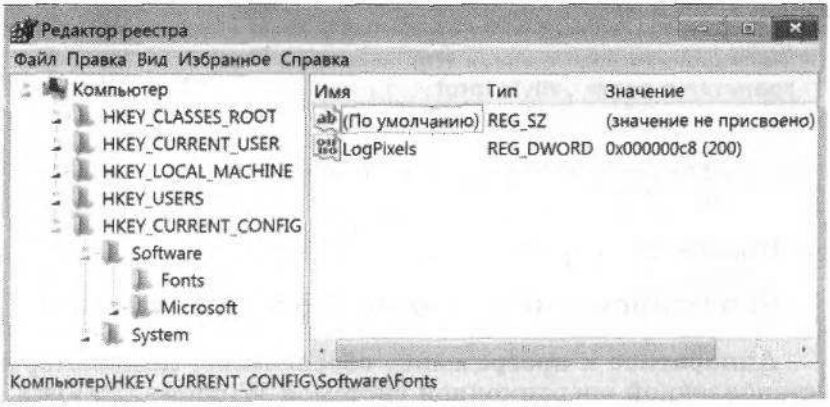
Редактирование системного реестра Windows

Для просмотра и редактирования системного реестра Windows можно использовать программу regedit.exe. Однако редактирование реестра следует проводить только в случае крайней необходимости и при условии понимания выполняемых действий! Неквалифицированное редактирование реестра может привести компьютер в неработоспособное состояние.

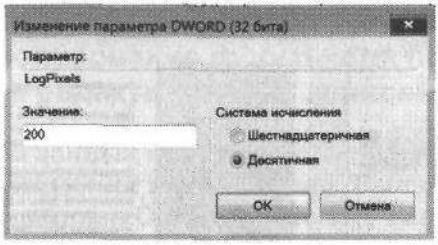
Редактор реестра Windows состоит из нескольких ветвей *HKEY_CLASSES ROOT*, *HKEY_CURRENT USER*, *HKEY_CURRENT MACHINE*, *HKEY_USERS* и *HKEY_CURRENT CONFIG*, исходящих из одного корня *Компьютер*.

Рассмотрим только одну из них *HKEY_CURRENT CONFIG*.

1. Запустить редактор реестра Windows regedit.exe.
2. Раскрыть ветвь *Software*, а в ней — ветвь *Fonts*.
   В правой части реестра откроется таблица, которая содержит пункт *LogPixels*, который определяет размер системного шрифта.



3. Изменить его размер, например увеличить до 200 в десятичной системе.
   После перезагрузки системы системный шрифт увеличится.



4. Чтобы вернуть начальное значение размера шрифта, необходимо снова выполнить все операции и ввести размер системного шрифта 120 в десятичной системе счисления.

Практическая работа 3.3

Создание генеалогического древа семьи

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой WindowsXP/Vista.

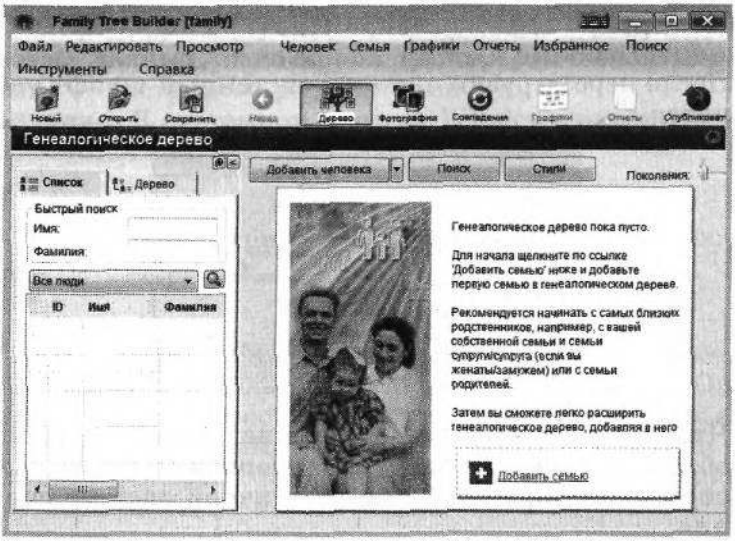
Цель работы. Научиться создавать генеалогическое древо семьи.

Задание. Создать базу данных «Генеалогическое древо семьи», основанную на сетевой модели данных.

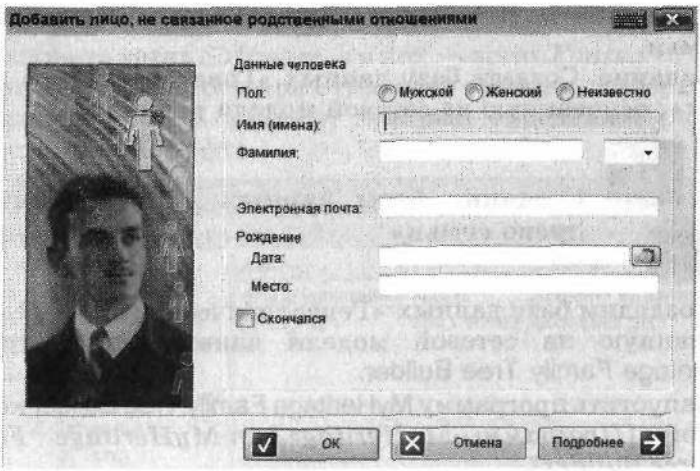
Создание базы данных «Генеалогическое древо семьи»

Создадим базу данных «Генеалогическое древо семьи», основанную на сетевой модели данных, в программе MyHeritage Family Tree Builder.

1. Запустить программу MyHeritage Family Tree Builder командой [*Программы-MyHeritage.com-MyHeritage Family Tree Builder*].
2. В появившемся диалоговом окне программы на *Панели инструментов* выбрать *Добавить человека* или в низу диалогового окна выбрать *Добавить семью*.



3. Вставить в генеалогическое древо своих родителей.
4. Поочередно выделить родителей и вставить в генеалогическое древо родителей своих родителей, т. е. дедушек и бабушек.
5. Поочередно выделить всех родственников в генеалогическом древе семьи и в контекстном меню объектов ввести сведения о них.



В результате мы получим генеалогическое древо семьи, которое является примером базы данных, основанной на сетевой модели данных.

База данных «Генеалогическое древо семьи» хранится в папке ..\IIKT11prof **Windows-CD** 

3.2.1. Использование формы для просмотра и редактирования записей

Записи базы данных можно просматривать и редактировать в виде таблицы или в виде формы. Выше мы работали с таблицей базы данных. Однако часто вид *Таблица* не позволяет видеть полностью всю информацию на экране. Если база данных содержит достаточно много полей, а значения полей содержат много символов, то не все поля таблицы могут умещаться на экране, а значения полей могут быть видны не полностью.

Форма одновременно отображает одну запись в удобном для пользователя виде. В процессе создания формы можно указать, какие поля базы данных включить в форму, как расположить поля в окне формы, а также как можно сделать форму визуально привлекательной.

Фактически с помощью формы создается графический интерфейс доступа к базе данных, который может содержать различные *управляющие элементы* (текстовые поля, кнопки, переключатели и т. д.), а также *надписи*. Обычно на форме размещаются *надписи*, являющиеся именами полей базы данных, и *текстовые поля*, содержащие данные из базы данных.

Пользователь может изменять *дизайн* формы (размер, цвет и т. д.) управляющих элементов и надписей.

Примерами форм могут являться *Визитка* в базе данных «Записная книжка» или *Карточка* в базе данных «Библиотечный каталог», которые содержат лишь одну запись базы данных, зато представленную в удобном для пользователя виде.

Вопросы для размышления

1. Какие достоинства и недостатки имеют таблицы и формы при отображении реляционной базы данных?

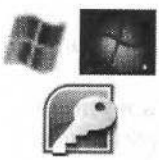
Практическая работа 3.4

Создание формы для реляционной базы данных

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows XP/Vista.

Цель работы. Научиться создавать формы для реляционных баз данных в СУБД Microsoft Access 2007 в операционной системе Windows.

Задание. Создать форму для реляционной базы данных «Процессоры» (см. практическую работу 3.1) в операционной системе Windows с помощью СУБД Microsoft Access 2007.

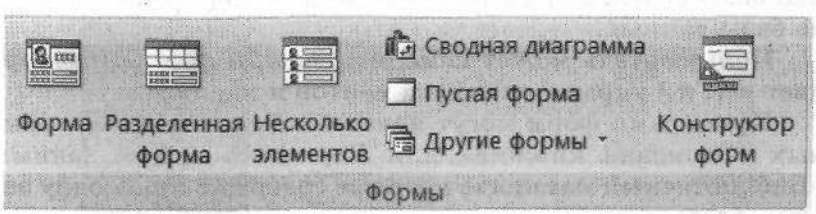


Создание формы для реляционной базы данных с помощью СУБД Microsoft Access 2007

1. В операционной системе Windows XP/Vista запустить СУБД Microsoft Access 2007.

Создание формы можно проводить различными способами.

2. Ввести команду [*Создание*] и на появившейся панели инструментов *Формы* выбрать способ создания формы.

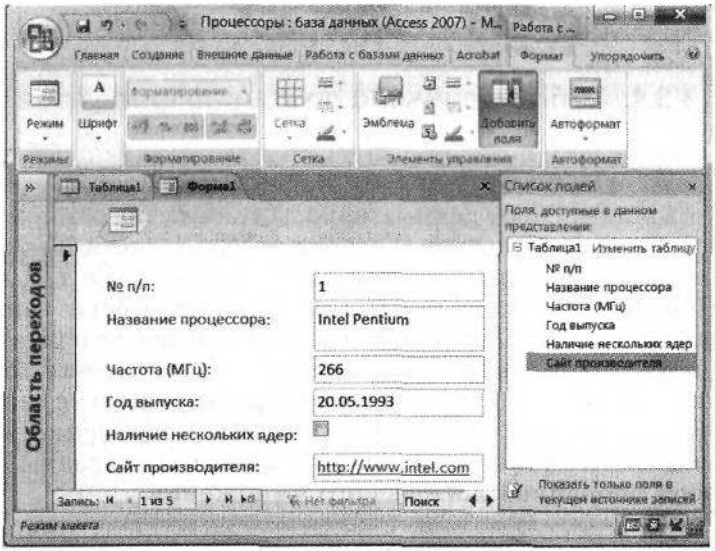


Создадим форму для базы данных «Процессоры» с помощью *Пустой формы*.

3. Щелкнуть по значку *Форма* и ввести команду [*Добавить поля*].

В правой части диалогового окна *Процессоры: база данных* появится перечень полей из *Таблицы1*.

Перетащить нужные поля на заготовку формы.



4. В результате появится окно *Форма1* базы данных «Процессоры», которое содержит *надписи* (названия полей базы данных) и *текстовые поля*, в которых отображаются данные одной записи базы данных.

Теперь форма для базы данных «Процессоры» готова. С ее помощью можно просматривать записи, редактировать их или вводить новые.

5. После открытия форма содержит запись № 1. При работе с формой для перехода от одной записи к другой необходимо воспользоваться панелью *Запись*, которая находится в нижней части окна формы. Панель *Запись* содержит кнопки со стрелками, щелчки по которым позволяют перемещаться по записям, а также поле номера записи, позволяющее ввести номер искомой записи.

Вид формы можно изменять в режиме *Конструктор*.

6. В окне *Процессоры: база данных* выделить *Форму1* и щелкнуть по пункту меню *Конструктор*.
На появившейся панели инструментов *Элементы управления* можно выбрать любой элемент управления и поместить на форму.

3.3. Отбор и сортировка данных

3.3.1. Отбор данных с помощью фильтров

СУБД Access позволяет производить отбор записей, в которых значения определенных полей удовлетворяют заданным условиям. Отбор данных из базы данных можно производить с помощью фильтров. Условия отбора записей создаются с использованием операторов сравнения (=, >, < и т. д.).

Простые фильтры содержат условие отбора записей только для одного поля. **Сложные фильтры** содержат несколько условий для различных полей. В результате применения сложного фильтра будут отобраны только те записи, которые удовлетворяют всем условиям одновременно. Можно сказать, что условия в сложных фильтрах связаны между собой операцией логического умножения.

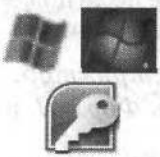
Практическая работа 3.5

Отбор данных с помощью фильтров из реляционной базы данных

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows XP/Vista.

Цель работы. Научиться отбирать данные с помощью фильтров в реляционных базах данных в СУБД Microsoft Access 2007 в операционной системе Windows.

Задание. В базе данных «Процессоры» с помощью фильтров отобрать записи, содержащие сведения о процессорах, имеющих несколько ядер, и частота которых больше 3000 МГц.



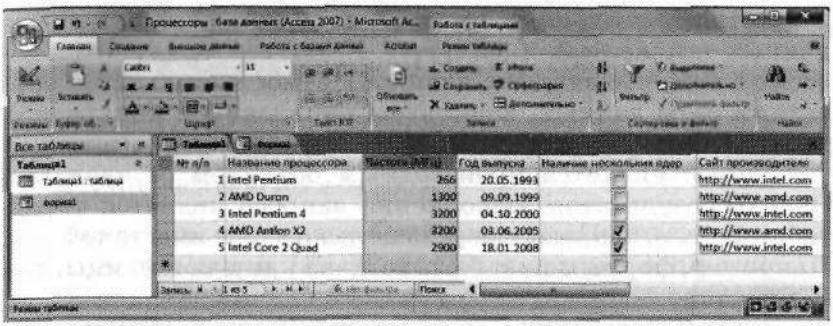
Отбор данных с помощью фильтров из реляционной базы данных с помощью СУБД Microsoft Access 2007

1. В операционной системе Windows XP/Vista запустить СУБД Microsoft Access 2007.

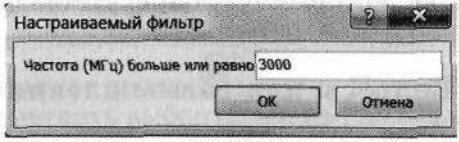
Создадим сложный фильтр для отбора данных из базы данных «Процессоры».

2. Открыть таблицу базы данных «Процессоры», дважды щелкнув по соответствующему значку в окне базы данных.

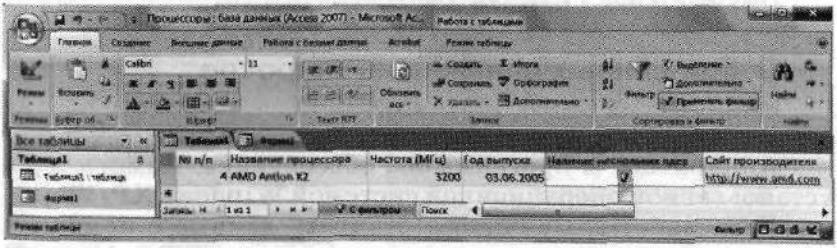
В появившейся *Таблице1* выделить поле *Частота*.



3. Ввести команду [*Главная-Фильтр*]. В появившемся списке выбрать пункт *Числовые условия* и условие отбора *Больше...*
   В появившемся диалоговом окне в текстовое поле ввести 3000.



4. Аналогично выполнить пункты 2–3 для поля *Наличие нескольких ядер*.
5. В появившемся диалоговом окне таблицы будут выведены записи, удовлетворяющие условиям отбора. В данном случае найден лишь один такой процессор — AMD Antlon X2.



6. Для того чтобы увидеть таблицу целиком, необходимо отменить фильтры командой [*Главная-Дополнительно-Очистить все фильтры*].

3.3.2. Отбор данных с помощью запросов

Запросы позволяют осуществлять отбор данных из баз данных так же, как и фильтры. Различие между ними состоит в том, что запросы являются самостоятельными объектами базы данных, а фильтры привязаны к конкретной таблице или форме.

Запрос является производным объектом от таблицы. Однако результатом выполнения запроса является также таблица, т. е. запросы могут использоваться вместо таблиц. Например, форма может быть создана как для таблицы, так и для запроса.

Запросы позволяют отобрать те записи, которые удовлетворяют заданным условиям. Запросы, как и фильтры, бывают простые и сложные. Простой запрос содержит одно условие, а сложный запрос — несколько условий для различных полей.

В процессе создания запроса можно отбирать не только записи, но и поля, которые будут присутствовать в запросе.

Вопросы для размышления

1. Чем отличаются запросы от фильтров при отборе данных из табличной базы данных?

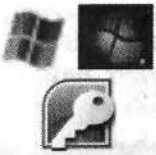
Практическая работа 3.6

Отбор данных с помощью запросов из реляционной базы данных

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows XP/Vista.

Цель работы. Научиться отбирать данные с помощью запросов из реляционных баз данных в СУБД Microsoft Access 2007 в операционной системе Windows.

Задание. Из базы данных «Процессоры» с помощью запроса отобрать записи, содержащие сведения о процессорах, имеющих несколько ядер, и частота которых больше 3000 МГц.

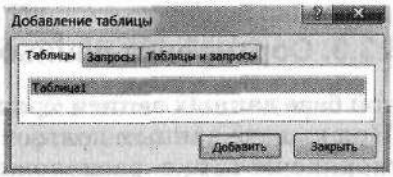


Отбор данных с помощью запросов из реляционной базы данных с помощью СУБД Microsoft Access 2007

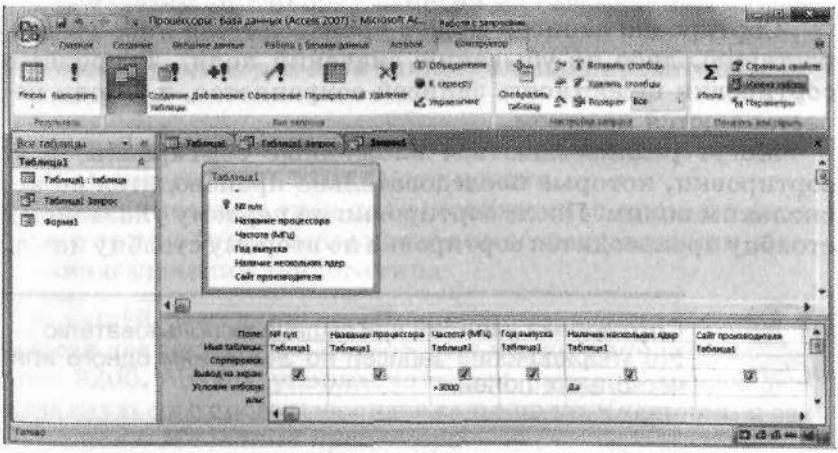
1. В операционной системе Windows XP/Vista запустить СУБД Microsoft Access 2007.

Создадим запрос для базы данных «Процессоры».

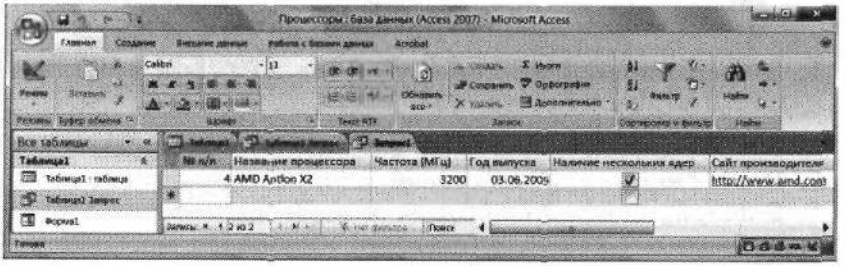
2. Ввести команду [*Создание-Конструктор запросов*].
В появившемся диалоговом окне *Добавление таблицы* выбрать пункт *Таблица1*.
Щелкнуть по кнопке *Добавить*.



3. В появившемся диалоговом окне *Процессоры: база данных* осуществить выбор полей из *Таблицы1*, которые будут отображаться в таблице запроса (в строке *Вывод на экран:* необходимо у этих полей поставить флажки).
В полях базы данных, по которым будут отбираться записи для запроса, в строке *Условие отбора:* необходимо указать условия, которым должны удовлетворять данные в этих полях.



В появившемся диалоговом окне запроса будут выведены записи, удовлетворяющие условиям поиска. В данном случае (как и с помощью фильтров) будет найден лишь один такой процессор — AMD Antlon X2.



3.3.3. Сортировка данных

В реляционной базе данных записи хранятся в неупорядоченном виде. При выдаче данных пользователю часто бывает удобно их упорядочить, т. е. расположить в определенной последовательности. Упорядочение записей называется **сортировкой**.

Сортировка записей производится по какому-либо полю. Значения, содержащиеся в этом поле, располагаются в определенном порядке, который определяется типом поля:

- по алфавиту, если поле текстовое;
- по величине числа, если поле числовое;
- по дате, если тип поля Дата/время и т. д.

Сортировка записей может производиться либо по возрастанию, либо по убыванию значений поля. В процессе сортировки целостность записей сохраняется, т. е. они переставляются целиком.

Могут реализовываться **вложенные сортировки**, т. е. сортировки, которые последовательно производятся по нескольким полям. После сортировки по первому указанному столбцу производится сортировка по второму столбцу и т. д.

Сортировка записей при выдаче их пользователю — это упорядочение записей по значениям одного или нескольких полей.

Вопросы для размышления

1. Чем отличается сортировка записей от вложенной сортировки?

Практическая работа 3.7

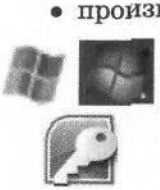
Сортировка данных в реляционной СУБД

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows XP/Vista.

Цель работы. Научиться сортировать данные в реляционной СУБД Microsoft Access 2007 в операционной системе Windows.

Задание. В базе данных «Процессоры»:

- произвести сортировку по одиночному полю;
- произвести вложенную сортировку с помощью запроса.



Сортировка данных в СУБД Microsoft Access 2007

1. В операционной системе Windows XP/Vista запустить СУБД Microsoft Access 2007.

Произведем сортировку записей из базы данных «Процессоры» по произвольному полю.

2. В окне *Процессоры: база данных* активизировать объект *Таблица1*.

3. Выделить произвольное поле базы данных и в контекстном меню выбрать пункт *Сортировка...*.
В зависимости от типа поля (хранящихся в нем данных) сортировка будет выполняться по возрастанию или убыванию значений данного типа.

В нашей базе данных «Процессоры» в поле *Частота* имеются две записи (3 и 4), которые имеют одинаковое значение 3200. Чтобы упорядочить эти записи, произведем вложенную сортировку — сначала по полю *Частота*, а затем по полю *Название процессора*.

Access позволяет выполнять вложенные сортировки с помощью запросов.

4. В окне *Процессоры: база данных* активизировать объект *Таблица1 Запрос*.

5. В нижней части запроса в строке *Сортировка:* ввести из раскрывающихся списков параметры сортировки в выбранных полях.

Поле:	[№ п/п]	[Название процессора]	[Частота (МГц)]	[Год выпуска]	[Наличие нескольких ядер]	[Сайт производителя]
Имя таблицы:	Таблица1	Таблица1	Таблица1	Таблица1	Таблица1	Таблица1
Сортировка:		по убыванию	по возрастанию			
Вывод на экран:	☑	☑	☑	☑	☑	☑
Условие отбора:						

Чтобы выполнить сортировку по двум или более полям, укажем, какие из полей будут использоваться в качестве внутренних и внешних полей сортировки.

6. В режиме *Конструктор* щелкнуть по кнопке *Страница свойств*.

   В появившемся *Окне свойств* в пункте *Порядок сортировки* указать последовательность названий полей базы данных, которая будет соответствовать порядку вложенной сортировки.

Окно свойств ×

Возможен выбор: Свойства запроса

Общие

Описание	
Режим по умолчанию	Режим таблицы
Вывод всех полей	Нет
Набор значений	Все
Уникальные значения	Нет
Уникальные записи	Нет
База данных-источник	(текущая)
Строка подключения-источник	
Блокировка записей	Отсутствует
Тип набора записей	Динамический набор
Время ожидания ODBC	60
Фильтр	
Порядок сортировки	[Частота (Мгц)], [Название процессора]

В результате получим последовательно отсортированные по двум полям записи из базы данных «Процессоры».

Таблица1 | Таблица1 Запрос | Запрос1

№ п/п	Название процессора	Частота (МГц)	Год выпуска	Наличие нескольких ядер	Сайт производителя
1	Intel Pentium	266	20.05.1993	☐	http://www.intel.com
2	AMD Duron	1300	09.09.1999	☐	http://www.amd.com
5	Intel Core 2 Quad	2900	18.01.2008	☑	http://www.intel.com
4	AMD Antlon X2	3200	03.06.2005	☑	http://www.amd.com
3	Intel Pentium 4	3200	04.10.2000	☐	http://www.intel.com

3.3.4. Печать данных с помощью отчетов

Можно осуществлять печать непосредственно таблиц, форм и запросов. Однако для красивой печати документов целесообразно использовать отчеты. Отчеты являются производными объектами баз данных и создаются на основе таблиц, форм и запросов.

Практическая работа 3.8

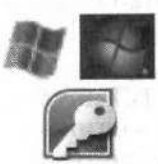
Подготовка отчетов

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows XP/Vista.

Цель работы. Научиться создавать простейшие отчеты из реляционных баз данных в СУБД Microsoft Access 2007 в операционной системе Windows.

Задание. Напечатать содержимое базы данных «Процессоры»:

- традиционным простым способом;
- с помощью отчета.



Печать содержимого реляционной базы данных с помощью СУБД Microsoft Access 2007

1. В операционной системе Windows XP/Vista запустить СУБД Microsoft Access 2007.
2. Щелкнуть по значку *Кнопка Microsoft Office*  . Ввести команду [*Печать-Печать*]. Содержимое базы данных «Процессоры» будет напечатано, однако не очень красиво.

Таблица1

№ п/п	Название процессора	Частота (МГц)	Год выпуска	Наличие нескольких ядер	Сайт производителя
1	Intel Pentium	266	20.05.1993	☐	http://www.intel.com
2	AMD Duron	1300	09.09.1999	☐	http://www.amd.com
3	Intel Pentium 4	3200	04.10.2000	☐	http://www.intel.com
4	AMD Antlon X2	3200	03.06.2005	☑	http://www.amd.com
5	Intel Core 2 Quad	2900	18.01.2008	☑	http://www.intel.com

Создадим отчет, который будет красиво распечатывать содержимое базы данных «Процессоры».

3. В окне *Процессоры: база данных* ввести команду [*Создание-Отчет*].
4. Для улучшения отчета ввести команду [*Режим-Режим макета*].

Появится отчет базы данных «Процессоры», который может быть распечатан.

Таблица1

23 июля 2008 г.
22:56:23

№ п/п	Название процессора	Частота (МГц)	Год выпуска	Наличие нескольких ядер	Сайт производителя
1	Intel Pentium	266	20.05.1993	☐	http://www.intel.com
2	AMD Duron	1300	09.09.1999	☐	http://www.amd.com
3	Intel Pentium 4	3200	04.10.2000	☐	http://www.intel.com
4	AMD Antlon X2	3200	03.06.2005	☑	http://www.amd.com
5	Intel Core 2 Quad	2900	18.01.2008	☑	http://www.intel.com

Страница 1 из 1

3.4. Многотабличные базы данных

Достаточно часто встречается ситуация, когда хранить все данные в одной таблице реляционной БД неудобно и нерационально. Таблица может содержать слишком большое количество полей. При этом во многих полях могут дублироваться данные, что увеличивает необходимое для хранения место и замедляет процедуры обработки данных.

Поясним это на примере. Пусть реляционная база данных «Комплектующие компьютера и поставщики» содержит информацию о различных комплектующих и имеет поля: *Счетчик, Наименование, Описание, Название фирмы, Адрес, Цена* (табл. 3.2).

Данные записей 1 и 2, 3 и 4, 5 и 6, 7 и 8 дублируются в полях *Наименование*, *Описание*, *Название фирмы* и *Адрес*. Таким образом, почти половину объема таблицы составляет избыточная, дублированная информация.

Проанализируем причину дублирования. Комплектующие компьютера имеют два неотъемлемых свойства: *Наименование* и *Описание*. *Название фирмы*, *Адрес* и *Цена* не являются свойствами комплектующих компьютера, они являются свойствами поставщика.

Таблица 3.2. **Комплектующие компьютера и поставщики**

Счетчик	Наименование	Описание	Название фирмы	Адрес	Цена, руб.
1	Системный блок	Intel Core 2 Quad	Фирма1	Адрес1	9000
2	Системный блок	Intel Core 2 Quad	Фирма2	Адрес2	10000
3	Монитор	19"	Фирма1	Адрес1	5000
4	Монитор	19"	Фирма2	Адрес2	6000
5	Клавиатура	104 кл.	Фирма1	Адрес1	250
6	Клавиатура	104 кл.	Фирма2	Адрес2	300
7	Мышь	Беспроводная	Фирма1	Адрес1	300
8	Мышь	Беспроводная	Фирма2	Адрес2	350

Естественно выделить из исходной таблицы две отдельные таблицы: «Комплектующие» и «Поставщики».

Чтобы однозначно идентифицировать записи в этих таблицах, введем в них первичные ключи.

В таблицу «Комплектующие» введем поле *Код комплектующих* (табл. 3.3).

Таблица 3.3. **Комплектующие**

Код комплектующих	Наименование	Описание
К1	Системный блок	Intel Core 2 Quad
К2	Монитор	19"
К3	Клавиатура	104 кл.
К4	Мышь	Беспроводная

В таблицу «Поставщики» введем поле *Код поставщика* (табл. 3.4).

Таблица 3.4. **Поставщики**

Код поставщика	Название фирмы	Адрес
П1	Фирма1	Адрес1
П2	Фирма2	Адрес2

О том, куда поместить информацию о цене, поговорим *позже*.

Вопросы для размышления

1. В каких случаях использование многотабличной базы данных дает преимущества над базой данных, состоящей из одной таблицы?

3.4.1. Связывание таблиц

После создания множества таблиц, содержащих данные, относящиеся к разным аспектам базы данных, необходимо обеспечить целостность базы данных. Для этого необходимо **связать** таблицы между собой.

При связи в отношении **«один-ко-многим»** каждой записи в одной (главной) таблице могут соответствовать несколько записей в другой (подчиненной) таблице, а запись в подчиненной таблице не может иметь более одной соответствующей ей записи в главной таблице.

Если одной записи в первой таблице могут соответствовать несколько записей во второй таблице, а одной записи во второй таблице — несколько записей в первой таблице, то реализуется связь в отношении **«многие-ко-многим»**.

В нашем случае реализуется именно такая связь. Одной записи в таблице «Комплектующие» соответствуют две записи в таблице «Поставщики», так как устройства одного вида продаются двумя фирмами. Одной же записи таблицы «Поставщики» соответствуют четыре записи таблицы «Комплектующие», так как одна фирма продает устройства четырех разных видов.

Две таблицы, находящиеся в отношении «многие-ко-многим», могут быть связаны только с помощью третьей (связующей) таблицы. Таблицы «Комплектующие» и «Поставщики» можно связать в отношении «многие-ко-многим» путем создания двух связей с отношением «один-ко-многим» для таблицы «Цена».

Таблицы «Комплектующие» и «Поставщики» будут главными по отношению к таблице «Цена».

Связь между таблицами устанавливает отношения между совпадающими значениями в полях с одинаковыми именами. С первичным ключом главной таблицы связывается одноименное поле подчиненной таблицы (*внешний ключ*).

В главной таблице «Комплектующие» поле *Код комплектующих* является первичным ключом, соответственно в подчиненной таблице «Цена» должно существовать одноименное поле, которое является внешним ключом.

Таблица «Поставщики» также является главной по отношению к таблице «Цена». Ее поле *Код поставщика* является первичным ключом, соответственно в подчиненной таблице «Цена» должно существовать одноименное поле, которое является внешним ключом.

Таким образом, таблица «Цена» должна содержать следующие поля (табл. 3.5):

- *Счетчик* (первичный ключ);
- *Код комплектующих* (поле внешнего ключа для таблицы «Комплектующие»);
- *Код поставщика* (поле внешнего ключа для таблицы «Поставщики»);
- *Цена* (числовое поле).

Таблица 3.5. **Цена**

Счетчик	Код комплектующих	Код поставщика	Цена, руб.
1	К1	П1	9000
2	К1	П2	10000
3	К2	П1	5000
4	К2	П2	6000
5	К3	П1	250
6	К3	П2	300
7	К4	П1	300
8	К4	П2	350

Межтабличная связь **обеспечивает целостность данных.** Связанные таблицы представляют собой единую базу данных, *в которой можно создавать новые таблицы, а также* запросы и отчеты, содержащие данные из связанных таблиц.

Прежде чем приступить к созданию многотабличной базы данных, необходимо продумать ее проект. Проект представляет собой модель будущей базы данных, состоящей из объектов и их связей, необходимых для выполнения поставленных задач.

Процесс проектирования включает, прежде всего, определение перечня необходимых таблиц и задание их структуры, а также установки типа связей между этими таблицами.

Вопросы для размышления

1. Какие типы связей между таблицами возможны в многотабличных базах данных?

Практическая работа 3.9

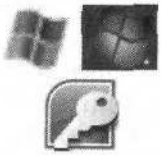
Многотабличные базы данных

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows XP/Vista.

Цель работы. Научиться создавать реляционные многотабличные базы данных и создавать к ним запросы в СУБД Microsoft Access 2007 в операционной системе Windows.

Задание:

- создать реляционную многотабличную базу данных «Компьютеры», в качестве основных объектов которой будут использованы три таблицы: «Комплектующие», «Поставщики» и «Цена». Таблицы «Комплектующие» и «Поставщики» должны быть связаны отношением «многие-ко-многим» с помощью таблицы «Цена»;
- создать запрос, который осуществляет отбор данных, необходимых для закупки дешевого системного блока.



Создание реляционной многотабличной базы данных с помощью СУБД Microsoft Access 2007

1. В операционной системе Windows XP/Vista запустить СУБД Microsoft Access 2007.

Система управления базами данных Microsoft Access позволяет создавать многотабличные базы данных, а также обеспечивать их обработку с помощью запросов, форм и отчетов.

Итак, прежде всего, необходимо создать три таблицы: «Комплектующие» «Поставщики» и «Цена».

2. Щелкнуть по значку *Кнопка Microsoft Office*  .

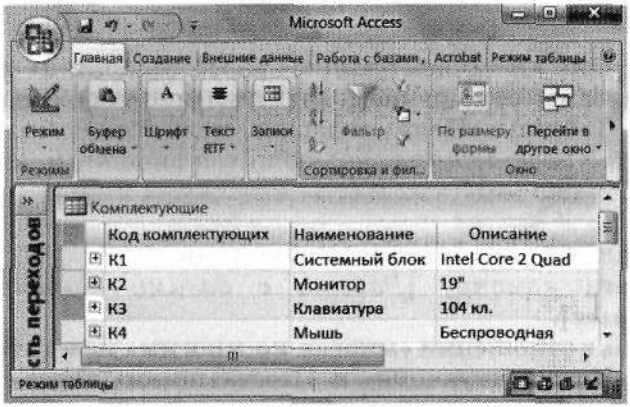
Создать в СУБД Microsoft Access 2007 новую базу данных с помощью команды [*Создать*].

В появившемся диалоговом окне в текстовом поле *Имя файла:* присвоить файлу базы данных имя Компьютеры.accdb.

Щелкнуть по кнопке *Создать*.

Таблица «Комплектующие» должна содержать три текстовых поля: *Код комплектующих*, *Наименование* и *Описание*.

3. В диалоговом окне *Компьютеры: база данных* создать таблицу «Комплектующие» и ввести данные.



4. Ввести команду [*Режим-Конструктор*].

В качестве первичного ключа задать поле *Код комплектующих*.

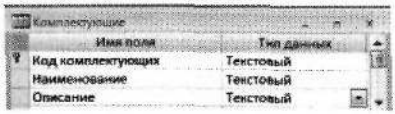


Таблица «Поставщики» должна содержать три текстовых поля: *Код поставщика*, *Название фирмы* и *Адрес*. Первичным ключом является поле *Код поставщика*.

5. Создать таблицу «Поставщики», выполнив рассмотренную выше последовательность действий. Ввести данные.

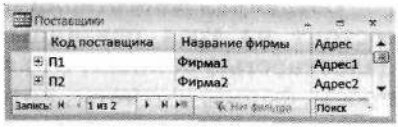
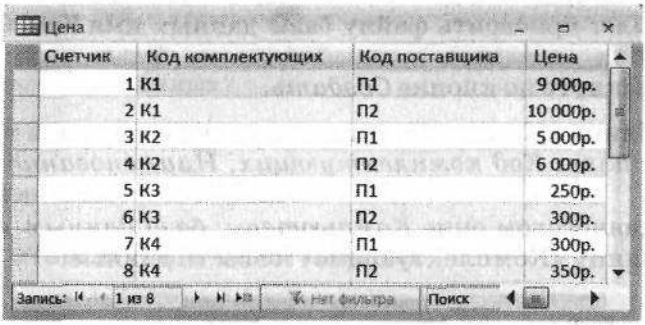


Таблица «Цена» должна содержать поля *Счетчик*, *Код комплектующих*, *Код поставщика*, а также поле *Цена*. В качестве первичного ключа этой таблицы будет использоваться поле *Счетчик*.

6. С помощью аналогичных действий создать таблицу «Цена» и ввести данные.



Цена

Счетчик	Код комплектующих	Код поставщика	Цена
1	К1	П1	9 000р.
2	К1	П2	10 000р.
3	К2	П1	5 000р.
4	К2	П2	6 000р.
5	К3	П1	250р.
6	К3	П2	300р.
7	К4	П1	300р.
8	К4	П2	350р.

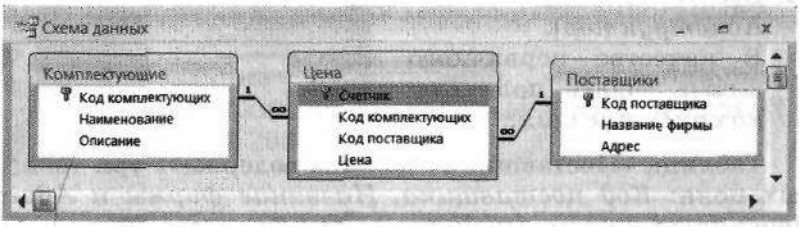
Запись: 1 из 8 | Нет фильтра | Поиск

Таблицы «Комплектующие» и «Поставщики» должны быть связаны отношением «один-ко-многим» с таблицей «Цена». Таблица «Цена» содержит поля *Код комплектующих* и *Код поставщика*, являющиеся внешними ключами исходных таблиц.

Установим связи между таблицами.

7. Ввести команду [*Работа с базами данных-Схема данных*].

Связь в отношении «многие-ко-многим» между таблицами «Комплектующие» и «Поставщики» через таблицу «Цена» будет установлена.

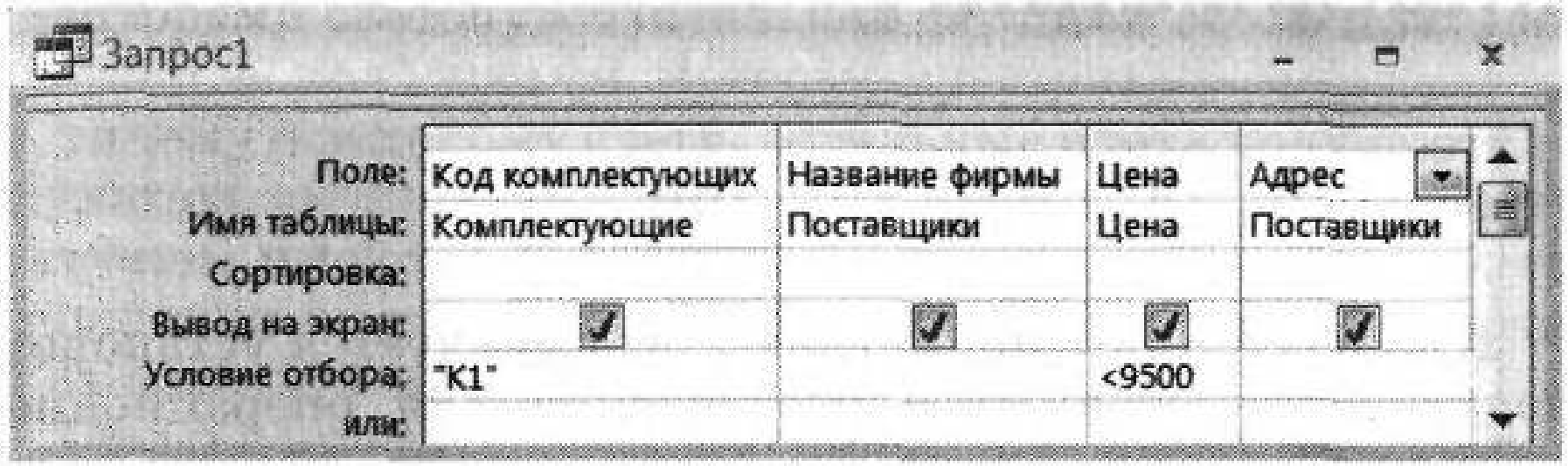


Созданная многотабличная база данных «Компьютеры» состоит из трех связанных таблиц и поэтому обладает целостностью данных. Это значит, что можно создавать запросы, формы и отчеты, которые используют данные из разных таблиц.

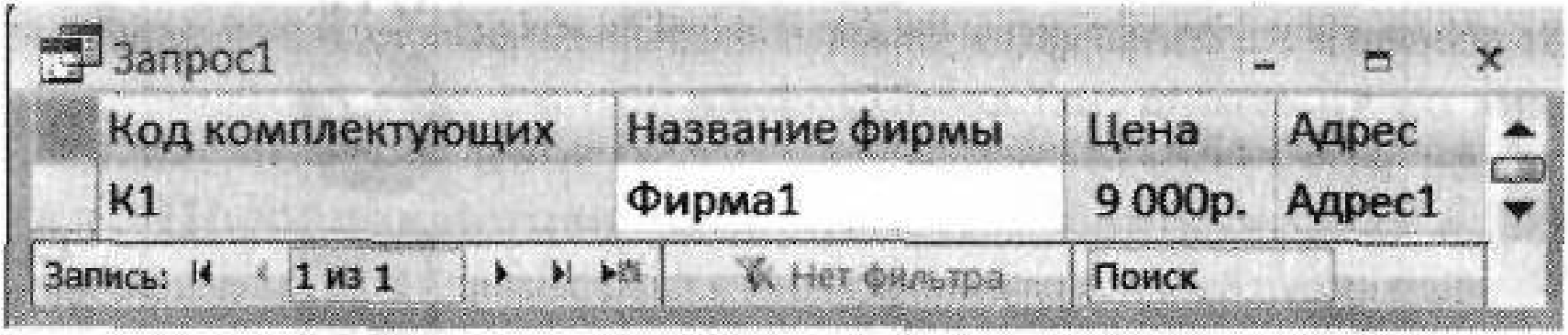
Создадим, например, запрос, который осуществляет отбор данных, необходимых для закупки дешевого системного блока.

8. Ввести команду [*Режим-Конструктор*].

9. В таблице «Комплектующие» для поля *Код комплектующих* ввести условие равно "К1", в таблице «Поставщики» для полей *Название фирмы* и *Адрес* установить вывод на экран, в таблице «Цена» для поля *Цена* ввести условие <9500.



10. Ввести команду [*Режим-Режим таблицы*]. Появится результат выполнения запроса.



Глава 4

Технология создания и обработки графической информации

При изучении данной главы рекомендуется установить следующее программное обеспечение для операционной системы Windows:

	• система векторной графики CorelDraw; • система растровой графики Adobe Photoshop	Первая помощь ПО 1.0.  	 CD-44 CD 31-36

Глава 1. Кодирование и обработка графической и мультимедийной информации **Информатика и ИКТ-9** 

4.1. Цветовой охват

Восприятие цвета человеком. Основной объем информации человек получает в виде зрительных образов с помощью света. Светом принято называть электромагнитное излучение с длиной волны от 440 до 700 нм. Только в этом диапазоне глаз может воспринимать электромагнитные волны. Меньшие значения соответствуют синей части спектра, а бóльшие значения — красной части спектра. Волны за пределами этого диапазона называются ультрафиолетовыми (УФ) и инфракрасными (ИК).

При работе на компьютере очень важно, чтобы выводимая информация была правильно представлена. При этом решающее значение имеет цвет. Для начала необходимо понять, как человек воспринимает цвета.

Согласно теории цветного зрения, цвет воспринимается рецепторами светочувствительной сетчатой оболочки глаза человека — колбочками, которые чувствительны к красному, синему и зеленому свету, остальные цвета получаются в результате смешивания этих трех цветов. Кроме колбочек в сетчатке имеются более чувствительные к свету рецепторы-палочки, которые не способны различать цвета и отвечают за восприятия оттенков серого.

Обычно при световосприятии раздражаются все три или два вида нервных окончаний (колбочек), и тогда глаз воспринимает сложный цвет. Глаз ощущает белый цвет, когда все виды нервных окончаний раздражаются одновременно и в одинаковой степени. Серый цвет ощущается глазом при одновременном раздражении нервных окончаний, но меньшей силы; черный цвет получается при отсутствии раздражения. Преобладающее раздражение какого-либо одного рецептора вызывает восприятие соответствующего цветового оттенка (рис. 4.1).

Рис. 4.1. Восприятие цвета человеком

Таким образом, человек воспринимает цвет как сумму излучений трех базовых цветов: красного, синего и зеленого.

Цветовой охват различных устройств. Цвет может быть представлен в природе, на экране монитора, на бумаге. Во всех этих случаях возможный диапазон цветов, или цветовой охват, будет разным.

Самым широким диапазоном восприятия цвета (наибольшим цветовым охватом) располагает нормальный человеческий глаз. Цветовой охват обычно представляется моделью Lab — цветовым пространством. В этой модели

значение светлоты отделено от значения хроматической составляющей цвета (тон, насыщенность). Светлота задана координатой E (изменяется от 0 до 100, т. е. от самого темного до самого светлого), хроматическая составляющая — координатами x и y. Первая обозначает положение цвета в диапазоне от фиолетового до зеленого, вторая — от фиолетового до пурпурного (рис. 4.2).

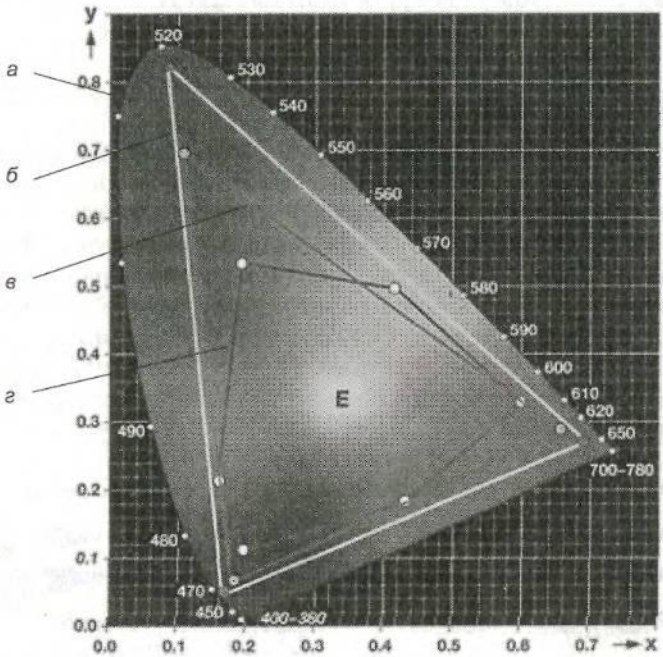


Рис. 4.2. Цветовой охват человеческого глаза (*а* — фигура целиком), высококачественного слайда (*б* — белый треугольник), монитора (*в* — темный треугольник), печати (*г* — темный многоугольник)

У цветной пленки цветовой диапазон шире, чем у монитора (у него проблемы с чистыми голубыми и желтыми цветами), который в свою очередь имеет более широкий диапазон (цветовой охват), чем устройства цветной печати (у них проблемы с цветами, составляющие которых имеют очень низкую плотность). Цветовой охват различных типов устройств, воспроизводящих цвет (начиная с цифровой камеры и заканчивая сложнейшими устройствами цифровой печати), показан на рис. 4.3.

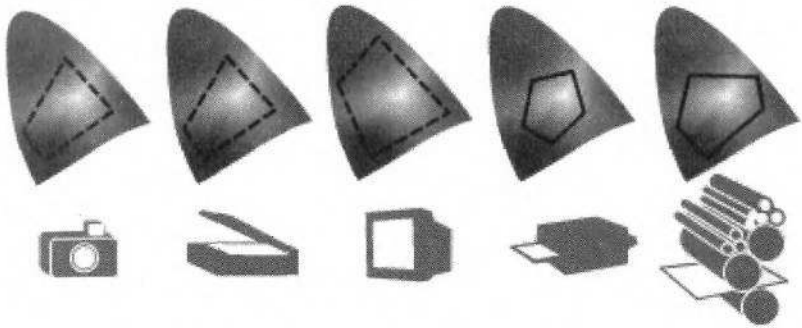


Рис. 4.3. Цветовой охват различных типов устройств

Задачей конструкторов является расширение цветового охвата устройства. Например, в струйных принтерах увеличивается количество цветных картриджей, которые способны передать тона и полутона.

Цветовая температура. Цвет имеет непосредственное отношение к температуре. При высокой температуре горения пламя имеет синий цвет. При низкой температуре горения цвет будет красным. Критерий измерения **цветовой температуры** используется для присваивания объективных числовых значений условиям освещения, при которых мы видим цвет.

Цветовая температура выражается в градусах по шкале Кельвина. Солнце в полдень имеет цветовую температуру 5500 К, а утром и вечером его температура составляет 4000 К. Люминесцентная лампа дневного света имеет температуру 6500 К (рис. 4.4). (Считается, что именно такой свет испускает нагретое до данной температуры абсолютно черное тело.)

Чем ниже цветовая температура, тем цвет ближе к красному; чем выше цветовая температура, тем цвет ближе к синему. Это объясняет, почему один и тот же красный элемент одежды будет выглядеть по-разному на улице и внутри при люминесцентном освещении.

100 000 К — цвет источника с «бесконечной температурой»;

7500 К — дневной свет с большой долей рассеянного от чистого голубого неба;

6500 К — стандартный источник дневного белого света;

5500 К — дневной свет, прямой солнечный;

3200–3250 К — киносъемочные лампы;

Рис. 4.4. Цветовая температура некоторых источников света

2800—2854 К — газонаполненные лампы накаливания с вольфрамовой спиралью;
2360 К — лампа накаливания, вакуумная;
2000 К — свет пламени свечи.

Мы можем регулировать цветовую температуру экрана монитора компьютера (обычно от 5000 К до 9300 К). Таким образом, мы можем смещать цветовой охват монитора либо в красную область, либо в фиолетовую область.

Вопросы для размышления

1. Насколько различается цветовой охват различных устройств?
2. Что такое цветовая температура? Какова цветовая температура различных источников света?

4.2. Палитры RGB и CMY

2.1.6. Палитры цветов в системах цветопередачи RGB и CMYK

Палитра RGB. Для моделирования цвета при обработке изображения используются палитры. Палитра — это способ описания цвета. Для *излучаемого* света (экран монитора) используется палитра RGB (Red — красный, Green — зеленый, Blue — синий).

Палитра RGB описывает три основных цвета, воспринимаемых глазом человека. Каждая составляющая (красная, зеленая, синяя) имеет определенную интенсивность (яркость свечения соответствующей окрашенной лампочки), которая выражается числом от 0 до 255 и занимает один байт.

Например, запись:

R:0 G:255 B:0 описывает зеленый цвет (красной и синей составляющей нет, зеленая максимальная);

R:255 G:255 B:0 описывает желтый цвет (красные и зеленые световые лучи одинаковой интенсивности воспринимаются как желтый цвет);

Белый цвет получается, если все элементы имеют максимальную яркость — 255 (R:255, G:255, B:255).

Палитра CMY. Палитра CMY используется для описания *отраженного* света (печать на бумаге) (Cyan — голубой, Magenta — пурпурный, Yellow — желтый).

Краски поглощают одну часть спектра и отражают другую. Например, желтая краска поглощает синий цвет и отражает красный и зеленый. Смешение всех трех красок дает черный цвет (поглощаются все составляющие спектра белого света). На практике для получения стопроцентного черного цвета используется палитра CMYK, которая получается путем добавления черной краски (blacK — черный). Это связано с экономией краски и невозможностью точного наложения красок друг на друга в процессе печати. Кроме того, незначительные изменения плотности нанесения краски приведут к тому, что черный цвет будет получаться коричневым.

На рис. 4.5 показан пример печати черного цвета тремя красками CMY и одной черной (K). Видно, что при печати текста или линий краски вылезают друг из-под друга, буквы становятся неровными, а линии — толще.

Рис. 4.5. Печать черного цвета

Интенсивность каждой краски задается в процентах. Например, запись C:100% M:0% Y:100% K:0% описывает зеленый цвет.

Соответствие между палитрами RGB и CMYK показано на рис. 4.6.

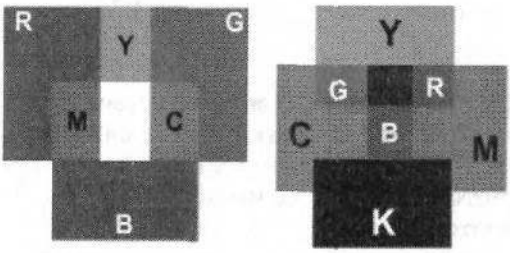


Рис. 4.6. Соответствие палитр RGB и CMYK

Цветовые справочники PANTONE®. Отображение цвета зависит от многих факторов. На практике нет однозначного соответствия палитр CMYK и RGB при передаче цвета на мониторе и на бумаге. В свою очередь, цветопередача разных мониторов может значительно различаться, также различается цветопередача при печати на бумаге различных типов.

Фактическим стандартом в области идентификации цветов являются цветовые справочники фирмы PANTONE®. PANTONE® является разработчиком и производителем технологических решений в области выбора цвета и точной цветовой коммуникации. Уже более 40 лет имя PANTONE® известно во всем мире как универсальный цветовой язык для общения заказчиков, дизайнеров и производителей полиграфической, текстильной и прочей продукции. Идентификация цветов распространяется не только на *изображения, выведенные на бумагу, но и на*

изображения, получаемые на экране. Пантонные справочники — это наборы листов с тестовой печатью с указанием числовых значений цвета в системе RGB или CMYK.

Существуют различные варианты справочников с образцами печати на обычной, мелованной, глянцевой или матовой бумаге.

www.pantone.ru

При подготовке изображения для печати на недорогих массовых мониторах цвета будут искажены. Воспользовавшись справочником, надо указать нужные значения цвета. Цвета, полученные в результате печати, должны соответствовать цветам в справочнике.

Имея цветовые справочники, можно откалибровать монитор. Надо нарисовать в программе верстки прямоугольники, закрасив их цветами согласно цветовому справочнику, а затем, используя регулировки монитора или выбирая различные профили в программах верстки, попытаться привести цвета на экране в соответствие с цветами в справочнике. Практически все современные видеокарты поставляются с драйверами, позволяющими производить настройку *цветопередачи*.

Естественно, что такая калибровка будет достаточно приблизительной. Для правильной калибровки монитора, сканера и принтера используются специальные устройства — калибраторы. Калибраторы устанавливаются перед экраном монитора и подключаются к компьютеру с помощью USB-порта (рис. 4.7).

Рис. 4.7. Монитор с калибратором и защитным кожухом

Вопросы для размышления

1. Чем отличаются системы цветопередачи RGB и CMY?
2. Для чего необходимы цветовые справочники PANTONE®?
3. Для чего используются калибраторы?

4.3. Растровая и векторная графика

1.2. Растровая и векторная графика Информатика и ИКТ-9

Все компьютерные изображения разделяются на два типа: растровые и векторные. Они отличаются друг от друга способом описания изображения.

В растровой графике все изображение делится на множество точек и сохраняется информация о цвете каждой точки. Качество изображения зависит от разрешения изображения — количества точек в строке на дюйм (1 дюйм = = 2,54 см). Например, разрешение 300 dpi (от англ. dots per inch — «точек на дюйм) означает, что на отрезке 2,54 см располагается 300 точек. Растровый файл содержит информацию о разрешении, системе цвета (RGB, CMYK и т. п.) и информацию о цвете каждой точки. Для вывода изображения на экран достаточно разрешения 72 dpi, для получения качественных отпечатков на бумаге необходимо разрешение 300 dpi.

Растровые файлы теряют качество изображения при изменении масштаба. Если файл имел разрешение 100 dpi и размер 5 × 5 см, то при увеличении размера в 5 раз (25 × 25 см) разрешение станет 20 dpi. Даже если при этом изменить разрешение на большее, качество изображения не улучшится. Например, указав новое разрешение 40 dpi, при размере 25 × 25 см получим изображение, где фактически четыре стоящие рядом точки имеют одно значение цвета. Программа, конечно, попытается оптимизировать изображение, сделать более плавный переход, но это не повысит качество, а линии станут более размытыми. При редактировании растрового файла следует всегда оставлять исходное качественное изображение. Исходный файл следует создавать с разрешением, достаточным для его дальнейшей обработки.

В векторной графике изображение формируется с помощью графических примитивов (точка, линия, окружность, прямоугольник и пр.), которые задаются их математическим описанием. Например, для описания окружности нужно указать ее радиус, толщину линии, цвета заливки и координаты центра. В дальнейшем можно без потери качества производить преобразования. Ниже, в практической работе 4.1, приводится пример масштабирования одного и того же изображения в растровом и векторном представлении. При этом видно, что векторное изображение масштабируется без потери качества.

Муаровый узор (от франц. *moire*) — узор, возникающий при наложении двух периодических сетчатых рисунков. Явление обусловлено тем, что повторяющиеся элементы двух рисунков следуют с немного разной частотой и то накладываются друг на друга, то образуют промежутки (рис. 4.8).

Рис. 4.8. Муаровый узор

Муаровый узор наблюдается при наложении друг на друга различных частей тюлевых занавесок. Понятие «муар» происходит от ткани муар, при отделке которой использовалось данное явление.

Муаровый узор возникает при цифровом фотографировании и сканировании сетчатых и других периодических изображений, если их период близок к расстоянию между светочувствительными элементами оборудования. Этот факт используется в одном из механизмов защиты денежных знаков от подделки: на купюры наносится волнообразный рисунок, который при сканировании может покрыться очень заметным узором, отличающим подделку от оригинала.

Вопросы для размышления

1. Чем различаются системы векторной и растровой графики?
2. Когда появляется муаровый узор?

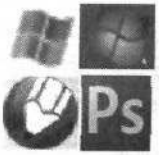
Практическая работа 4.1

Растровая и векторная графика

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows XP/Vista.

Цель работы. Создать изображение в системе векторной графики CorelDraw и изображение в системе растровой графики Adobe Photoshop.

Задание. Создать текст и изменить его масштаб в системах векторной и растровой графики.



Векторные рисунки и растровые изображения

1. В системе векторной графики CorelDraw поместить с использованием инструмента *Текст* на поле рисования текст «текст» и увеличить его в несколько раз с использованием инструмента *Масштаб*:

текст текст

2. В системе растровой графики Adobe Photoshop поместить с использованием инструмента *Горизонтальный текст* на поле рисования текст «текст» и увеличить его в несколько раз с использованием инструмента *Масштаб*:

текст текст

4.4. Устройства ввода графической информации

В основном ввод графических изображений осуществляется с помощью сканера и цифровых фотокамер.

Сканер. Рассмотрим принцип действия планшетных сканеров как наиболее распространенных моделей. Сканируемый объект кладется на стекло планшета сканируемой поверхностью вниз. Под стеклом располагается подвижная каретка с лампой и системой зеркал, движение которой регулируется шаговым двигателем.

Свет, отраженный от объекта, через систему зеркал попадает на светочувствительную матрицу и передается в компьютер. За каждый шаг двигателя сканируется полоска объекта, которая потом объединяется с другими в общее изображение (рис. 4.9).

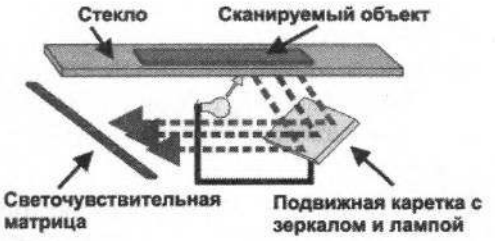


Рис. 4.9. Устройство планшетного сканера

Светочувствительная матрица состоит из фотодиодных элементов, чувствительных к свету. Перед каждым фотодиодом стоит светофильтр, пропускающий лучи только определенного цвета (красный, зеленый или синий). Стоящие три рядом элемента формируют изображение в системе RGB.

В зависимости от способа сканирования объекта существуют следующие виды сканеров:

- планшетные — наиболее распространенный вид сканеров. Представляет собой планшет, внутри которого под прозрачным стеклом расположен механизм сканирования;
- слайд-сканеры — служат для сканирования пленочных слайдов, выпускаются и как самостоятельные устройства, и в виде дополнительных модулей к обычным сканерам;
- сканеры штрих-кода — небольшие, компактные модели для сканирования штрих-кодов товара в магазинах. Сканер штрихкода представляет собой устройство, главная функция которого — чтение изображения штрихкода, представленного в виде совокупности белых и черных полос, и преобразование его в цифровой сигнал. Функцию преобразования выполняет специальный декодер, как правило, встроенный в сканер.

Разрешение сканеров измеряется в точках на дюйм (dpi). Чем больше светочувствительных элементов у сканера, тем больше точек он может снять с каждой горизонтальной полосы изображения. Это и называется оптическим разрешением. Чем мельче шаг каретки, тем больше можно считать полос с изображения. Шаг каретки определяет аппаратное разрешение сканера. Например, разрешение 600×1200 dpi означает что количество светочувствительных элементов на дюйм — 600, шагов на дюйм — 1200.

Глубина цвета измеряется количеством оттенков, которые устройство способно распознать. Современные сканеры выпускают с глубиной цвета 24, 30, 36, 48 битов. Так как сканер работает в системе RGB, то при глубине цвета 24 бита на каждый канал (красный, зеленый, синий) приходится по одному байту.

Цифровая фотокамера состоит из объектива и светочувствительной матрицы (рис. 4.10). В отличие от сканера светочувствительная матрица фотокамеры представляет собой не линейку, а плоскость, на которую через объектив фокусируется изображение. Матрицы сканера и фотокамеры имеют значительные различия. Каждый пиксель матрицы камеры закрыт светофильтром одного из цветов RGB. Зеленых элементов в два раза больше, чем красных и синих. Это связано с особенностью человеческого зрения, которое наиболее чувствительно к зеленому цвету, и при фотографировании камера должна принимать изображение так, как его воспринимает человек.

Рис. 4.10. Устройство цифровой фотокамеры

Вопросы для размышления

1. Каково устройство сканера? Цифровой фотокамеры?

4.5. Устройства вывода графической информации

Точки на экране монитора и бумаге. Находящиеся рядом мелкие точки разных цветов воспринимаются глазом человека как одна, состоящая из смеси красок. Например, две мелкие точки, светящиеся красным и зеленым светом, будут восприниматься как одна желтая. Расположенные рядом две

капли краски на бумаге — желтая и синяя — будут восприниматься как зеленая. Это свойство используется при выводе изображения на экран монитора и при печати на бумаге.

Вывод на экран монитора. Каждая точка экрана монитора состоит из трех рядом расположенных светящихся элементов (красного, зеленого, синего) (рис. 4.11). Так как монитор воспроизводит цвета в системе RGB, то перед выводом на экран информация о цвете точек переводится в эту систему.

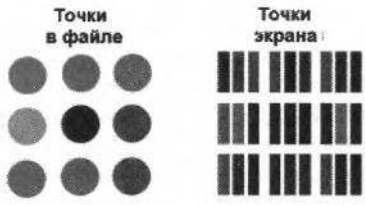


Рис. 4.11. Точки на экране монитора

Элементы экрана могут светиться с разной интенсивностью, которая формирует полутона. Часто встречается представление цвета, в котором каждой компоненте отводится 8 битов (в общей сложности — 24 бита). Так в точке со значением цвета R:0 G:128 B:0 красный и синий элементы будут выключены, а зеленый будет светится «вполнакала». В современных устройствах отображения используется и более высокое разрешение — 32 бита на точку изображения.

Вывод изображения на принтере. Цветное изображение печатается на бумаге с использованием четырех базовых цветов: C — голубой, M — малиновый, Y — желтый, K — черный.

Для передачи полутонов при рисовании кистью художник смешивает в определенном количестве разные краски. Принтер не может смешивать краски в разной пропорции. Для передачи полутонов изменяется количество закрашенных точек на бумаге. Если в файле точка имеет плотность синего 50%, это значит, что на небольшом участке белой бумаги половина площади должна быть закрашена синей краской. Таким образом, одной точке растрового файла соответствует множество точек, распечатанных с помощью принтера (рис. 4.12).

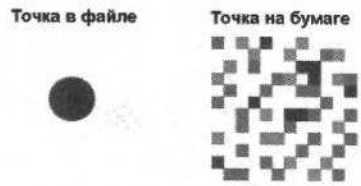


Рис. 4.12. Точки, распечатанные на принтере

Допустим,что разрешение растрового файла 400 dpi, а разрешение принтера — 4800 dpi. Это значит, что каждой

точке файла соответствует напечатанный квадрат со стороной 4800/400 = 12 точек (см. рис. 4.12). Такой квадрат может передать 12×12 = 144 оттенка.

Матричные принтеры. Матричные принтеры — это принтеры ударного действия. В печатающей головке матричного принтера находятся стержни. Под воздействием магнитного поля они выталкиваются из головки и через красящую ленту переносят изображение на бумагу. Перемещаясь, головка формирует строку (рис. 4.13).

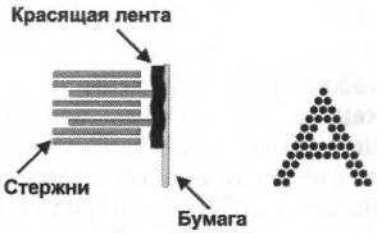


Рис. 4.13. Принцип формирования изображения на матричном принтере

Матричные принтеры характеризует низкая разрешающая способность и малая скорость печати. Из-за возможности печати на рулонном материале и самокопирующейся бумаге такие принтеры наибольшее распространение получили в банковской сфере, кассовых аппаратах, в маркировочных устройствах.

Лазерные принтеры. Лазерные принтеры используют технологию фотокопирования, называемую еще электрофотографической, которая заключается в переносе заряженных частиц тонера (красителя) на бумагу. Подобная технология печати применяется также в копировальных аппаратах. Лазерные принтеры характеризует высокая скорость печати и хорошая разрешающая способность, а также низкая стоимость тонера.

Важнейшим конструктивным элементом лазерного принтера является вращающийся фотобарабан, с помощью которого производится перенос изображения на бумагу. Фотобарабан представляет собой металлический цилиндр, покрытый тонкой пленкой из фотопроводящего полупроводника. На небольшом расстоянии от барабана проходит проволока, на которую подается высокое напряжение. За счет этого образуется ионизированная область, и статический заряд переходит на фотобарабан.

Тонкий лазерный луч, отражаясь от вращающегося зеркала, засвечивает на фотобарабане точки (количество максимально возможных точек на дюйм определяет разрешение лазерного принтера), повторяя выводимое изображение. На

засвеченных участках статический разряд снимается. Частицы тонера имеют заряд, совпадающий по полярности с зарядом фотобарабана, поэтому на тех участках, где сохранился заряд, тонер отталкивается, а где заряд снят лучом лазера, притягивается.

Бумага заряжается зарядом, противоположным по полярности заряду тонера. Проходя под фотобарабаном, она притягивает частицы тонера, и на листе формируется изображение. Это изображение еще не закреплено и легко снимается с бумаги. Проходя через нагретый вал-печку, тонер расплавляется и спекается с бумагой (рис. 4.14).

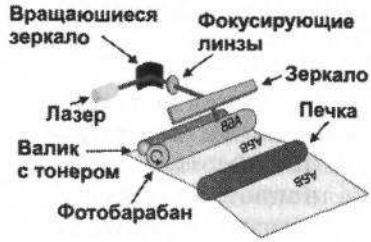


Рис. 4.14. Принцип формирования изображения на лазерном принтере

Цветные лазерные принтеры последовательно наносят на бумагу базовые краски (голубая, пурпурная, желтая, черная). Для каждого цвета может использоваться отдельный фотобарабан (рис. 4.15).

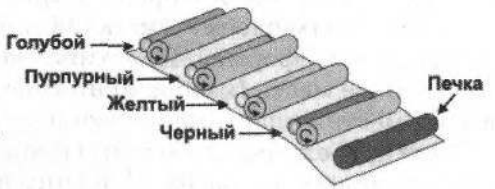


Рис. 4.15. Принцип формирования изображения на цветном лазерном принтере

Струйные принтеры. Струйные принтеры выводят изображение путем разбрызгивания краски. Головка струйного принтера состоит из резервуара с краской и тонкого сопла. Для выброса чернил применяются способы выдавливания или нагрева.

При выдавливании чернил пьезоэлемент, встроенный в сопло, расширяется и выбрасывает каплю краски (рис. 4.16).

Рис. 4.16. Принцип формирования изображения на струйном принтере

Струйные принтеры характеризует высокое качество печати и большой расход чернил. Из-за возможности печати на рулонном материале и использования различных видов краски струйные принтеры применяются для изготовления рекламных плакатов. Для этой цели используются специальные широкоформатные принтеры (струйные плоттеры).

Офсетная печатная машина. В типографиях печать осуществляется в основном на офсетных машинах. Каждая секция офсетной печатной машины наносит одну краску. Полноцветное изображение получается при последовательном проходе бумаги через все печатные секции.

Перед началом печати необходимо подготовить специальное клише — офсетную форму. Отредактированное на компьютере изображение передается на фотовывод. Фотовыводное устройство производит цветоделение и растрирование, а затем выводит цветоделенное изображение на пленку. Каждой краске соответствует своя пленка. При печати полноцветного изображения выводятся четыре пленки (CMYK). Пленка укладывается на незасвеченную алюминиевую пластину, одна сторона которой покрыта специальным светочувствительным слоем. С каждой пленки делается отдельная офсетная форма. Пластина через пленку засвечивается. Темные места (изображения) не пропускают лучи света, в этих местах пластина остается незасвеченной, прозрачные места (пробельные элементы) пропускают свет, засвечивая слой на пластине. Затем пластину промывают в специальном растворе. Эмульсия с засвеченных участков смывается и остается только на незасвеченных. Свойства эмульсии таковы, что на ней не удерживается пленка воды, но при этом прилипает масляная краска.

При печати на пластину постоянно наносится тонкий слой увлажняющего раствора. За счет поверхностного натяжения вода остается только на пробельных элементах, так как там нет эмульсии. Затем краска посредством красочного валика наносится на форму. Слой воды не дает возможности краске перейти на пробельные элементы, и она переносится только на печатные элементы (рис. 4.17).

Сформированное изображение переносится на резиновую поверхность печатного цилиндра и с него на бумагу. Полноцветное изображение получается при последовательном проходе бумаги через все печатные цилиндры.

Рис. 4.17. Офсетная печать

Ризограф. Низкая стоимость оборудования и расходных материалов, простота в работе позволяют использовать ризограф для оперативной печати небольших тиражей. На базе ризографа, подключенного к компьютеру, можно создать школьную мини-типографию.

Внешне ризограф напоминает лазерный копировальный аппарат, но в отличие от последнего изображение наносится на бумагу методом трафаретной печати. **Принцип трафаретной печати** заключается в продавливании краски сквозь печатную форму на печатный материал через открытые элементы трафарета с помощью ракели (устройства, напоминающего швабру). В зависимости от типа станка трафарет или подготавливается фотоспособом, (аналогично подготовке офсетной пластины), или прожигается лазером (рис. 4.18).

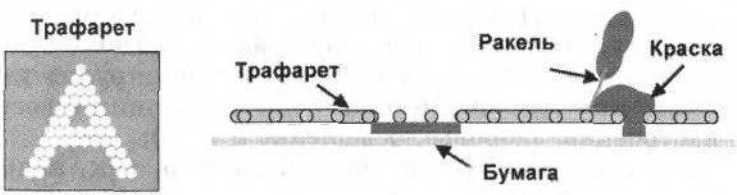


Рис. 4.18. Трафаретная печать

Почти все устройства выводят изображение дискретным способом. Экран монитора состоит из множества мелких светящихся точек, принтер печатает мелкие точки. Глаз человека не может различить мелкие детали, они сливаются и воспринимаются как непрерывные линии и фигуры. Поэтому любое изображение, даже векторное, перед выводом на экран разбивается на точки.

Вопросы для размышления

1. Чем отличается вывод точки на экран монитора и на бумагу с помощью принтера?
2. Каково устройство матричного, струйного и лазерного принтеров?
3. Каково устройство офсетной машины? Ризографа?

4.6. Системы управления цветом

Воспроизвести во всем цветовом диапазоне цветной слайд средствами полиграфической печати — трудновыполнимая задача. Цветное изображение (например, с фотографии или акварели), проходя все этапы обработки, начиная с ввода с помощью сканера или цифровой камеры, обработки и отображения на экране монитора и заканчивая выводом на печатном устройстве, претерпевает разнообразные изменения в связи с неизбежным конвертированием цветовых систем.

Исходных причин этой ситуации — три:

- восприятие цвета — сложный психофизиологический процесс, который, видимо, никогда не удастся моделировать техническими средствами, поскольку на восприятие оказывают влияние тысячи трудно учитываемых условий (возраст, цветовосприимчивость, настроение, здоровье, освещение и т. д.);
- цвет нельзя измерить непосредственно, как, например, длину. Цвет измеряется как спектральная композиция из световых волн различной интенсивности и различной длины;
- передача цветовой информации вынуждает пользоваться неким «языком», а точнее сказать, несколькими языками, поскольку у различных устройств они разные, например устройства ввода и мониторы используют цветовую модель RGB, а устройства вывода — цветовую модель CMYK. Следовательно, для достижения полной цветовой идентичности оригинала и печатного оттиска требуется «перевод» (конвертирование) с одного языка на другой, что никогда не обходится без потерь.

Одним из способов выхода из этой ситуации являются системы управления цветом. Некоторые программы обработки графической информации включают систему управления цветом (Color Management System, CMS), которая позволяет контролировать работу с цветом на экране монитора и при выводе на внешние печатные устройства. Система управления цветом основана на едином цветовом пространстве, которое обеспечивается использованием цветовой модели CIE Lab.

Вторым важным компонентом системы управления цветом являются цветовые профили используемых устройств — файл, описывающий соответствие цветов устройств ввода или вывода в терминах цветовой модели CIE Lab.

И наконец, третьим компонентом, обеспечивающим непосредственно управление, служит так называемая «машина цветового соответствия» (color-matching engine). Она должна получить цветовой профиль устройства ввода, например сканера, цветовой профиль промежуточного устройства (монитора), если требуется редактирование изображения, и цветовой профиль устройства вывода, например струйного принтера, и очень редко параметры «печатного станка», и обеспечить сканирование и отображение на экране монитора таким образом, чтобы пользователь редактировал как бы «конечное» изображение, то есть печатный оттиск. Кроме того, это позволяет хотя бы частично имитировать на экране или с помощью цветных принтеров пробную печать, обеспечивая существенную экономию при подготовке к печати изданий среднего качества.

Вопросы для размышления

1. Для чего необходимы системы управления цветом?

Практическая работа 4.2

Системы управления цветом в CorelDraw и Adobe Photoshop

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows XP/Vista.

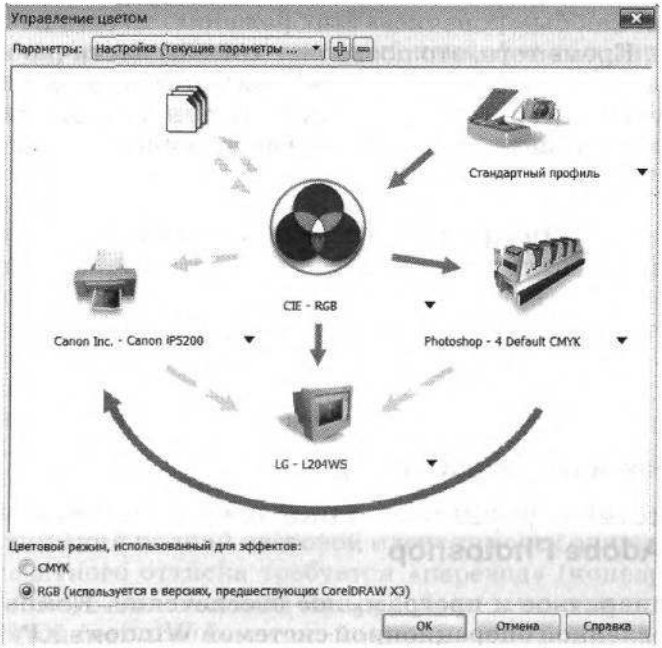
Цель работы. Использовать системы управления цветом в системах векторной графики CorelDraw и растровой графики Adobe Photoshop.

Задание 1. В системе векторной графики CorelDraw для каждого устройства (принтер, монитор, сканер, цифровая камера) выбрать оптимальный цветовой профиль или использовать стандартный.

Задание 2. В системе растровой графики Adobe Photoshop выбрать рабочие пространства и стратегии управления цветом. Конвертировать цвета из цветовой модели RGB в цветовую модель CMYK.

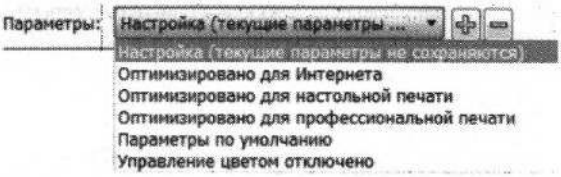
Системы управления цветом в CorelDraw и Adobe Photoshop

1. В системе векторной графики CorelDraw ввести команду [*Инструменты-Управление цветом...*].
   Появится диалоговое окно *Управление цветом*. Для каждого устройства (принтер, монитор, сканер, цифровая камера) можно выбрать оптимальный цветовой профиль или использовать стандартный.

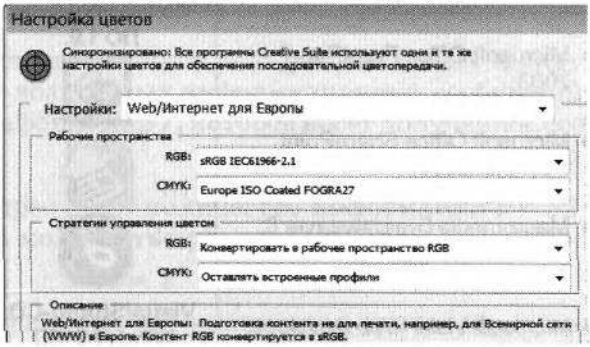


Также, активизируя ту или иную стрелку, можно выбрать приоритет цветопередачи.

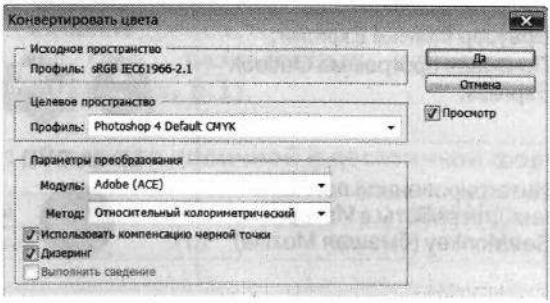
2. С использованием раскрывающегося списка *Параметры:* можно выбрать оптимальную цветовую схему для различных случаев.



1. В системе растровой графики Adobe Photoshop ввести команду [*Редактирование-Настройка цветов...*]. Появится диалоговое окно *Настройка цветов*. В этом окне можно выбрать рабочие пространства и стратегии управления цветом.



2. Ввести команду [*Редактирование-Конвертировать цвета...*]. В диалоговом окне *Конвертировать цвета* можно, например, конвертировать цвета из цветовой модели RGB в цветовую модель CMYK.



Глава 5

Коммуникационные технологии

Глава 3. Коммуникационные технологии — *Информатика и ИКТ-8*

При изучении данной главы рекомендуется установить следующее программное обеспечение для операционных систем Windows и Linux:

 	• Microsoft Office FrontPage 2003; • Microsoft Office SharePoint Designer 2007; • Macromedia Dreamweaver 8;	**Первая помощь ПО 1.0.**   	 **CD-53** **CD-54** **CD-31**
	• Microsoft Visial Web Developer 2005 Express Edition; • Интегрированное приложение для работы в Интернете SeaMonkey (бывшая Mozilla);	**VisualStudio-CD Windows-CD** 	
	• Браузер Internet Explorer; • Почтовая программа Outlook Express;	**Операционная система Windows** 	
	• Интегрированное приложение для работы в Интернете SeaMonkey (бывшая Mozilla)	**Linux-DVD** 	

5.1. Глобальная компьютерная среда Интернет

Глобальная сетевая среда Интернет, по действующему определению, это сообщество сетей, взаимодействующих с помощью протоколов TCP/IP.

5.1.1. Адресация в Интернете

IP-адрес. Для того чтобы в процессе обмена информацией компьютеры могли найти друг друга, в Интернете существует единая система адресации, основанная на использовании IP-адресов.

Каждый компьютер, подключенный к Интернету, имеет свой уникальный двоичный 32-битовый **IP-адрес**.

Вы знаете формулу, которая связывает между собой количество возможных информационных сообщений N и количество информации I, которое несет полученное сообщение:

$$N = 2^I.$$

IP-адрес несет количество информации $I = 32$ бита, тогда общее количество различных IP-адресов N равно:

$$N = 2^I = 2^{32} = 4\ 294\ 967\ 296.$$

На практике это количество значительно меньше, поскольку действуют сложные правила распределения и разделения адресов на диапазоны.

Также IP-адрес можно представить десятиразрядным десятичным числом. Для удобства восприятия двоичный 32-битовый IP-адрес можно разбить на четыре части по 8 бит и каждую часть представить в десятичной форме. Десятичный Интернет-адрес состоит из четырех чисел в диапазоне от 0 до 255, разделенных точками (например, 213.171.37.202) (табл. 5.1).

Таблица 5.1. **IP-адрес в двоичной и десятичной формах**

Двоичный	11010101	10101011	00100101	11001010
Десятичный	213	171	37	202
				3584763338

Все узлы Интернета, постоянно обслуживающие клиентов, имеют постоянные IP-адреса. Однако провайдеры Интернета часто предоставляют пользователям доступ в Интернет не с постоянным, а с временным IP-адресом. IP-адрес может меняться при каждом подключении к Интернету, но в процессе сеанса остается неизменным, и пользователь может его определить.

Практическая работа 5.1

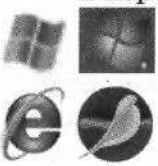
IP-адрес в различных форматах

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux, подключенный к глобальной компьютерной сети Интернет.

Цель работы. Научиться представлять IP-адрес в различных форматах.

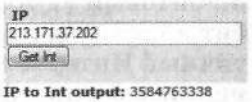
Задание:

- задать IP-адрес стандартным способом четырьмя десятичными числами и перевести его в десятиразрядное десятичное число;
- задать IP-адрес десятиразрядным десятичным числом и перевести его в стандартный формат, состоящий их четырех десятичных чисел;
- задать IP-адрес десятиразрядным десятичным числом и перевести его в двоичное число.



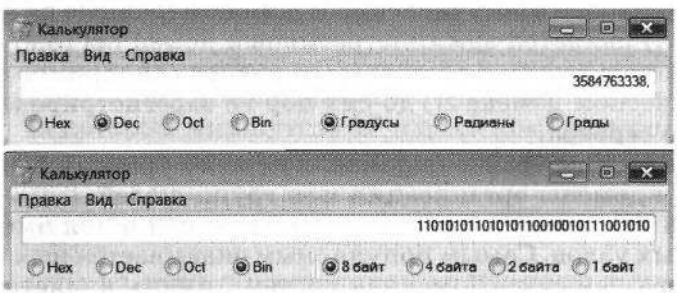
IP-адрес в различных форматах

1. В браузер загрузить сайт по адресу http://www.logbud.com/ip_code
2. В текстовое поле *IP* ввести IP-адрес стандартным способом, четырьмя десятичными числами.
   Щелкнуть по кнопке *Get Int*.
   На выходе мы получим IP-адрес в виде десятиразрядного десятичного числа.



3. В текстовое поле *INT* ввести IP-адрес в виде десятиразрядного десятичного числа.
   Щелкнуть по кнопке *Get IP*.
   На выходе мы получим стандартный IP-адрес в виде четырех десятичных чисел.

4. Ввести в программный калькулятор IP-адрес в виде десятиразрядного десятичного числа.
Выбрать переключатель *BIN*. Получим IP-адрес длиной 32 двоичных бита.



5.1.2. Доменная система имен

Человеку запомнить числовой адрес нелегко, поэтому для удобства пользователей Интернета была введена **доменная система имен**. Доменная система имен ставит в соответствие IP-адресу некоторое уникальное доменное имя.

Каждый компьютер, подключенный к Интернету, имеет IP-адрес, однако он может не иметь доменного имени. Доменные имена имеют серверы Интернета, но обычно не имеют компьютеры, подключающиеся к Интернету периодически.

Единицей именования в этой среде является домен — именованное объединение узлов в дереве DNS.

Доменная система имен имеет иерархическую структуру: корень — домены верхнего уровня — домены второго уровня — домены третьего уровня. Доменное имя сервера Интернета состоит из последовательности (справа налево) имен домена верхнего уровня, домена второго уровня и собственно имени компьютера. Так, основной сервер компании Microsoft имеет имя www.microsoft.com, а сервер Московского института открытого образования имеет имя www.mioo.ru.

Домены верхнего уровня бывают двух типов: географические и административные. Каждой стране мира выделен свой географический домен, обозначаемый двухбуквенным

кодом. Например, России принадлежит географический домен ru, в котором российские организации и граждане имеют право зарегистрировать домен второго уровня.

База данных «Доменная система имен» основана на иерархической модели данных. На верхнем уровне находится реляционная база данных, содержащая перечень доменов верхнего уровня (всего 269 доменов), из которых 12 — административные, а остальные 257 — географические. Наиболее многочисленным доменом (данные на январь 2008 г.) является административный домен net (около 190 миллионов имен), а в некоторых доменах (например, в географическом домене zr) до сих пор не зарегистрировано ни одного сервера.

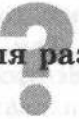
На втором уровне (в доменах первого уровня) регистрируются домены организаций или групп людей.

На третьем уровне чаще всего перечисляются имена отдельных узлов. Самым популярным оказывается имя www.

База данных «Доменная система имен» является распределенной — ее составные части находятся на разных DNS-серверах. Поиск информации в такой иерархической распределенной базе данных ведется следующим образом. Например, мы хотим ознакомиться с содержанием WWW-сервера фирмы Microsoft.

Сначала наш запрос, содержащий доменное имя сервера www.microsoft.com, будет отправлен на DNS-сервер нашего провайдера, который переадресует его на DNS-сервер самого верхнего уровня базы данных. Будет найден интересующий нас домен com, и запрос будет адресован на DNS-сервер второго уровня, который содержит перечень доменов второго уровня, зарегистрированных в домене com.

На DNS-сервере, хранящем записи второго уровня, будет найден домен microsoft, и при необходимости запрос будет переадресован на DNS-сервер третьего уровня. Поскольку имя www запрашивается часто, то, скорее всего, оно будет найдено сразу. Поиск информации в базе данных «Доменная система имен» будет завершен и начнется поиск компьютера в сети по его IP-адресу.



Вопросы для размышления

1. Имеет ли каждый компьютер, подключенный к Интернету, IP-адрес? Доменное имя?
2. Как строится доменная система имен?

5.1.3. Маршрутизация и транспортировка данных по компьютерным сетям

Среда Интернет, являющаяся сетью сетей и объединяющая огромное количество различных локальных, региональных и корпоративных сетей, функционирует и развивается благодаря использованию единого принципа маршрутизации и транспортировки данных.

Маршрутизация данных. Маршрутизация данных обеспечивает передачу информации между компьютерами сети. Передаваемая по сети информация «упаковывается в конверт», на котором «пишутся» IP-адреса компьютеров получателя и отправителя, например: «Кому: 198.78.213.185», «От кого: 193.124.5.33». Содержимое конверта на компьютерном языке называется IP-пакетом и представляет собой набор байтов.

Пакеты на пути к компьютеру-получателю также проходят через многочисленные промежуточные серверы Интернета, на которых производится операция маршрутизации. В результате маршрутизации IP-пакеты направляются от одного сервера Интернета к другому, постепенно приближаясь к компьютеру-получателю.

Маршрутизация IP-пакетов, обеспечивает доставку информации от компьютера-отправителя к компьютеру-получателю.

Маршруты доставки IP-пакетов могут быть совершенно разными, и поэтому они могут приходить не в том порядке, в котором их отправили. «География» Интернета существенно отличается от привычного нам пространственного распределения. Скорость получения информации зависит не от удаленности сервера Интернета, а от маршрута прохождения информации, т. е. количества промежуточных серверов и качества линий связи (их пропускной способности), по которым передается информация от сервера к серверу.

Транспортировка данных. В Интернете часто случается ситуация, когда компьютеры обмениваются большими по объему сообщениями. Попытка послать такое сообщение целиком приведет к затруднениям во время его отправки — например, из-за резко возрастающей верятности повреждений.

Для того чтобы этого не происходило, на компьютере-отправителе необходимо разбить большое сообщение на мелкие части, пронумеровать их и транспортировать в виде отдельных пакетов до компьютера-получателя.

На компьютере-получателе необходимо собрать исходное сообщение из отдельных частей в правильной последовательности, поэтому сообщение не может быть собрано до тех пор, пока не придут все IP-пакеты.

Транспортировка данных производится путем разбиения сообщений на IP-пакеты на компьютере-отправителе, индивидуальной маршрутизации каждого пакета и сборки сообщений из пакетов в первоначальном порядке на компьютере-получателе.

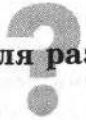
Время транспортировки отдельных IP-пакетов между локальным компьютером и сервером Интернета можно определить с помощью специальных программ.

Обмен данными между прикладными программами в среде Интернет выполняется с помощью набора протоколов TCP/IP.

Термин TCP/IP включает название двух протоколов передачи данных:

- TCP (Transmission Control Protocol — транспортный протокол, обеспечивающий обмен между программами);
- IP (Internet Protocol — протокол обмена пакетами между узлами разных сетей).

Помимо этой пары основных протоколов, набор включает в себя много отдельных протоколов для выполнения служебных и прикладных задач.



Вопросы для размышления

1. Объясните, каким образом производится доставка данных по указанному Интернет-адресу.
2. В каких целях при передаче сообщений по компьютерным сетям производится их разбиение на IP-пакеты?

Практическая работа 5.2
«География» Интернета

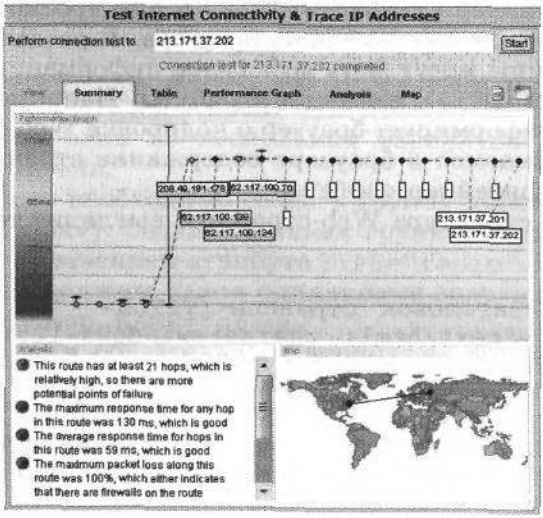
Аппаратное и программное обеспечение. Компьютер с установленной операционной системой Windows или Linux, подключенный к Интернету.

Цель работы. При работе в Интернете научиться получать информацию о маршруте прохождения данных между локальным компьютером и удаленным сервером Интернета.

Задание. Определить «удаленность» сервера Интернета от локального компьютера, т. е. провести трассировку маршрута прохождения данных от локального компьютера к удаленному Интернет-серверу.

Определение маршрута прохождения данных от локального компьютера до удаленного Интернет-сервера

1. В операционной системе Windows или Linux загрузить в браузер сайт по адресу http://visualroute.visualware.com/
2. Получить маршрут прохождения информации до сайта с доменным именем или Интернет-адресом (например, 213.171.37.202).
   На карте мира получить отображение маршрута прохождения данных от локального компьютера до указанного выше удаленного Интернет-сервера.



5.2. Интерактивные формы на Web-страницах

5.2.1. Структура HTML-кода Web-страницы

Для создания Web-страниц служит **язык разметки гипертекстовых документов HTML** (Hyper Text Markup Language).

С точки зрения HTML, весь объем информации представляется в виде страниц. Каждая страница — специально размеченный текст (подготовленный человеком или специализированной программой). Основным элементом разметки в языке HTML является тэг, т. е. указание, как обрабатывать часть текста. Тэги выделяются угловыми скобками и могут быть одиночными или парными. Парные тэги содержат открывающий и закрывающий тэги (такая пара тэгов называется **контейнером**). Закрывающий тэг содержит прямой слэш (/) перед обозначением. Тэги могут записываться как прописными, так и строчными буквами. Большинство тэгов имеют атрибуты — параметры обработки или применения тэгов. Атрибуты записываются в угловых скобках открывающего тэга в виде пар «имя-значение».

HTML-код Web-страницы помещается в контейнер <HTML></HTML> и состоит из двух частей: заголовка и отображаемого в браузере содержания. Заголовок страницы помещается в контейнер <HEAD></HEAD>.

Заголовок содержит название страницы, которое помещается в контейнер <TITLE></TITLE> и при просмотре отображается в верхней строке окна браузера. В раздел заголовка Web-страницы могут быть добавлены информационные одиночные тэги <META>, имеющие атрибуты. Например, атрибут charset информирует браузер о кодировке Web-страницы.

Отображаемое в браузере содержание страницы помещается в контейнер <BODY></BODY>.

Общая структура Web-страницы выглядит так:

```
<HTML>
<HEAD>
<TITLE>Заголовок страницы</TITLE>
<META ="text/html; charset=windows-1251"
 http-equiv="content-type">
</HEAD>
<BODY>
...
</BODY>
</HTML>
```

5.2.2. Создание интерактивных Web-страниц

Для того чтобы посетители сайта могли не только просматривать информацию, но и отправлять сведения для обработки и использования на сервер, на страницах сайта размещают интерактивные формы. Формы включают в себя элементы управления различных типов: текстовые поля, раскрывающиеся списки, флажки, переключатели и т. д.

Пусть мы хотим разместить на странице «Анкета» анкету для посетителей, чтобы выяснить, кто из наших посетителей, с какими целями и с помощью каких программ получает и использует информацию из сети Интернет, а также узнать, какую информацию они хотели бы видеть на нашем сайте.

Вся форма заключается в контейнер `<FORM></FORM>`.

Текстовые поля. В первую очередь выясним имя посетителя нашего сайта и его электронный адрес, чтобы иметь возможность ответить ему на замечания и поблагодарить за посещение сайта. Для получения этих данных разместим в форме два однострочных текстовых поля для ввода информации.

Текстовые поля создаются с помощью тэга `<INPUT>` со значением атрибута `TYPE="text"`. Атрибут `NAME` является обязательным и служит для идентификации полученной информации. Значением атрибута `SIZE` является число, задающее длину поля ввода в символах.

Для того чтобы анкета «читалась», необходимо разделить строки с помощью тэга перевода строки `<BR>`.

Переключатели. Далее, мы хотим выяснить, к какой группе пользователей относит себя посетитель. Предложим выбрать ему один из нескольких вариантов: учащийся, студент, учитель.

Для этого необходимо создать группу переключателей («радиокнопок»). Создается такая группа с помощью тэга `<INPUT>` со значением атрибута `TYPE="radio"`. Все элементы в группе должны иметь одинаковые значения атрибута `NAME`. Например, `NAME="group"`.

Еще одним обязательным атрибутом является `VALUE`, которому присвоим значения `"учащийся"`, `"студент"` и `"учитель"`. Значение атрибута `VALUE` должно быть уникальным для каждой «радиокнопки», так как при ее выборе именно оно передается серверу.

Флажки. Далее, мы хотим узнать, какими сервисами Интернета наш посетитель пользуется наиболее часто. Здесь из предложенного перечня он может выбрать одновременно несколько вариантов, пометив их флажками.

Флажки создаются в тэге <INPUT> со значением атрибута TYPE="checkbox". Флажки, объединенные в группу, могут иметь различные значения атрибута NAME. Например, NAME="box1", NAME="box2" и т. д.

Еще одним обязательным атрибутом является VALUE, которому присвоим значения "WWW", "e-mail" и "FTP". Значение атрибута VALUE должно быть уникальным для каждого флажка, так как при его выборе именно оно передается серверу.

Поля списков. Теперь выясним, какой из браузеров предпочитает посетитель сайта. Перечень браузеров представим в виде раскрывающегося списка, из которого можно выбрать только один вариант.

Для реализации **раскрывающегося списка** используется контейнер <SELECT></SELECT>, в котором каждый элемент списка определяется тэгом <OPTION>.

В переключателях, флажках и списках выбранный по умолчанию элемент задается с помощью атрибута SELECTED.

Текстовая область. В заключение поинтересуемся, что хотел бы видеть посетитель на наших страницах, какую информацию следовало бы в них добавить. Так как мы не можем знать заранее, насколько обширным будет ответ читателя, отведем для него текстовую область с линейкой прокрутки. В такое поле можно ввести достаточно длинный текст.

Создается текстовая область с помощью тэга <TEXTAREA> с обязательными атрибутами: NAME, задающим имя области, ROWS, определяющим число строк, и COLS — число столбцов области.

Отправка данных из формы. Отправка введенной в форму информации осуществляется с помощью щелчка по кнопке.

Кнопка создается с помощью тэга <INPUT>. Атрибуту TYPE необходимо присвоить значение "submit", а атрибуту VALUE, который задает надпись на кнопке, присвоить значение "Отправить".

Щелчком по кнопке *Отправить* можно отправить данные из формы на определенный адрес электронной почты. Для этого атрибуту ACTION контейнера <FORM> надо присвоить значение адреса электронной почты. Кроме того, в атрибутах METOD и ENCTYPE необходимо указать метод и форму передачи данных:

```
<FORM ACTION="mailto:test@metodist.ru"
METHOD="POST" ENCTYPE="text/plain">
```

Создание Web-страниц в Web-редакторах. HTML-код Web-страниц можно создавать не «вручную», а с использованием специальных программ — Web-редакторов. В них код элементов управления интерактивной формы создается автоматически.

Вопросы для размышления

1. Какие тэги используются для создания на форме текстовых полей? Переключателей? Флажков? Раскрывающихся списков? Текстовых областей?
2. Какие значения необходимо присвоить атрибутам тэга <FORM> для отправки введенной в форму информации?

Практическая работа 5.3

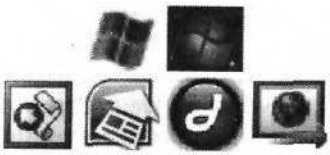
Разработка интерактивной Web-страницы с использованием Web-редакторов

Аппаратное и программное обеспечение. Компьютер с установленной операционной системой WindowsXP/Vista, подключенный к Интернету.

Цель работы. Научиться создавать интерактивные Web-страницы с использованием различных Web-редакторов.

Задание. В операционной системе Windows создать интерактивную Web-страницу с использованием следующих Web-редакторов:

Microsoft Office FrontPage 2003;
Microsoft Office SharePoint Decigner 2007;
Macromedia Dreamweaver 8;
Microsoft Visial Web Developer 2005 Express Edition.

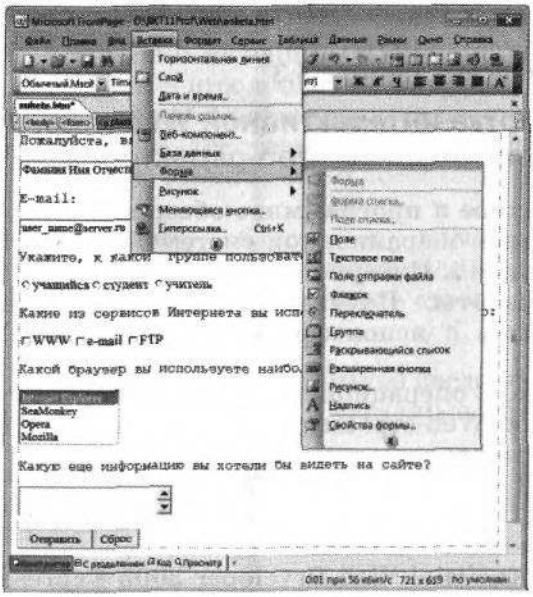


Разработка интерактивной Web-страницы с использованием Web-редакторов

Интерактивная Web-страница «Анкета». Интерактивная Web-страница «Анкета» содержит форму, которая заключается в контейнер <FORM></FORM>. В первую очередь выясним имя посетителя нашего сайта и его электронный адрес, чтобы иметь возможность ответить ему на замечания и поблагодарить за посещение сайта.

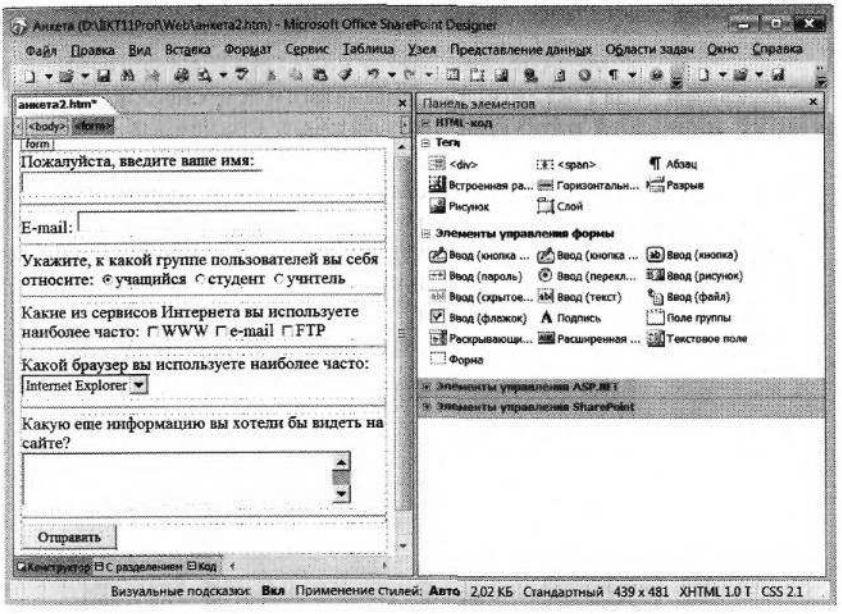
Web-редактор FrontPage часто вставляет лишние тэги, которые можно удалить в ручном режиме.

1. Запустить Web-редактор FrontPage.
   В режиме *Конструктор* ввести команду [*Вставка-Форма*]. Ввести все необходимые элементы управления. В частности, вставить два текстовых поля и поясняющие надписи к ним.
   В контекстном меню текстовых полей выбрать пункт *Свойство поля формы...* и в появившемся диалоговом окне ввести значения в текстовые поля.
   Перейти в режим *Код* и при желании изменить HTML-код.



Вставим в HTML-код группу переключателей, в которой устанавливается, к какой группе пользователей относится посетитель.

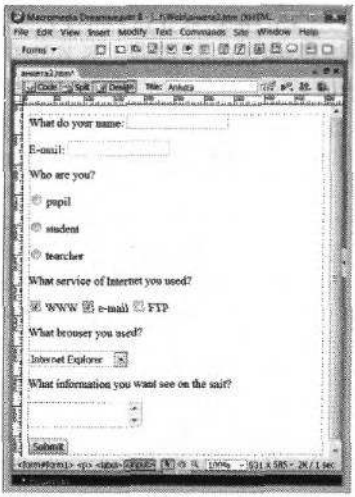
2. Запустить Web-редактор SharePoint Decigner.
   В режиме *Конструктор* на *Панели управления* выбрать и ввести все необходимые элементы управления.
   В частности, вставить группу переключателей и поясняющие надписи к ним.
   В контекстном меню переключателя выбрать пункт *Свойство поля формы...* и в появившемся диалоговом окне ввести значение *Выбран*.
   Перейти в режим *Код* и при желании изменить HTML-код.



Вставим в HTML-код группу флажков, которые выявляют наиболее популярные сервисы Интернета.

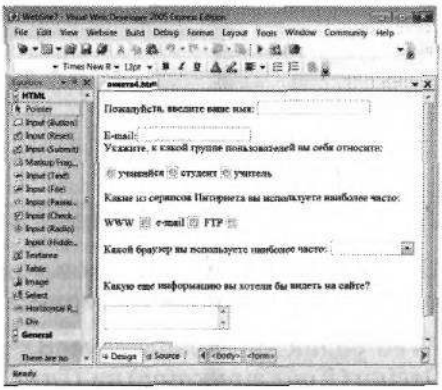
3. Запустить Web-редактор Dreamweaver 8.
   В режиме *Design* (*Конструктор*) на *Панели управления* выбрать и ввести все необходимые элементы управления.
   В частности, вставить группу флажков и поясняющие надписи к ним.

В контекстном меню флажков выбрать пункт *Свойство поля формы...* и в появившемся диалоговом окне ввести *Checked* (*Выбран*). Перейти в режим *Code* (*Код*) и при желании изменить HTML-код.

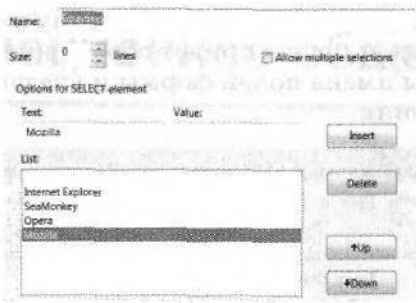


Вставим в HTML-код раскрывающийся список, содержащий названия наиболее популярных браузеров.

4. Запустить Web-редактор Visial Web Developer.
Ввести команду [*File-New Web Site...*]. В появившемся диалоговом окне выбрать пункт *Empty Web Site*.
В режиме *Design* (*Конструктор*) на *Toolbox* (*Панели элементов управления*) выбрать и ввести все необходимые элементы управления.
В частности, вставить раскрывающийся список.



5. В контекстном меню списка выбрать пункт *Properties* (*Свойства*) и в появившемся диалоговом окне ввести значения списка, а также установить значение по умолчанию.



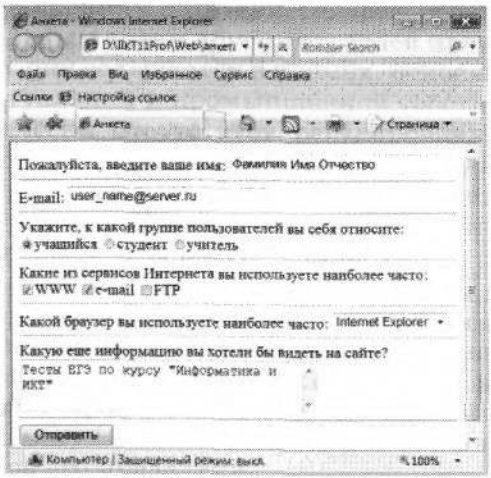
6. Перейти в режим *Source* (*Код*) и при желании изменить HTML-код.

Вставим в HTML-код текстовую область, в которой посетитель сайта может высказать свои замечания и предложения.

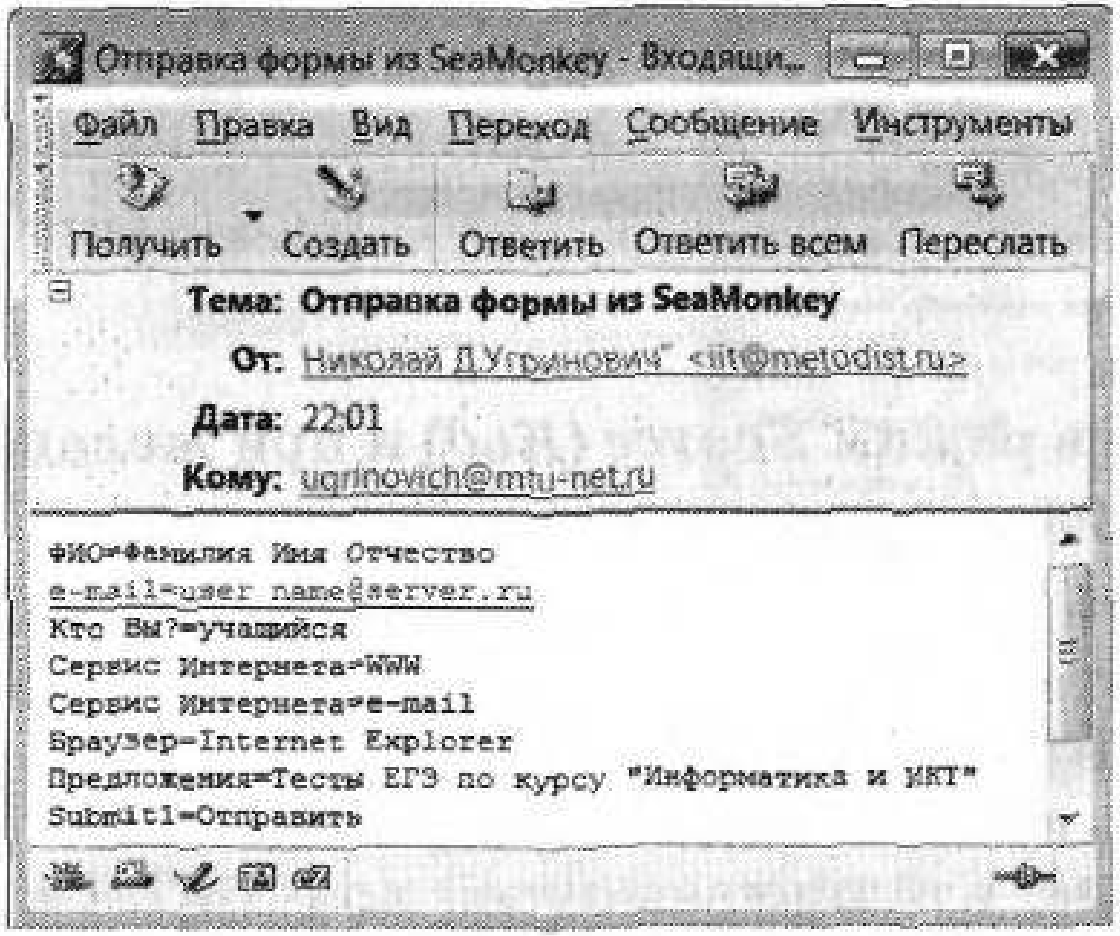
7. Вставить в HTML-код текстовую область.

Чтобы данные из интерактивной формы были отправлены по указанному адресу электронной почты или на сервер, необходимо указать этот адрес и создать кнопку *Отправить*.

8. В любом Web-редакторе вставить кнопку *Submit* (*Отправить*).
9. После открытия в браузере Web-страницы «Анкета» и внесения данных в поля формы необходимо щелкнуть по кнопке *Отправить*. Данные будут отправлены по указанному адресу электронной почты.



10. Через несколько секунд по указанному в «Анкете» адресу электронной почты придет сообщение, в котором будут указаны имена полей формы и введенные пользователем значения.



«Анкеты», выполненные в различных Web-редакторах, хранятся в папке ..\IIKT11prof\Web **Windows-CD** ☞

Глава 6

Информационное общество

6.1. Право в Интернете

Глобальный характер Интернета создает значительные проблемы в определении того, какие органы должны рассматривать споры по правовым вопросам. Интернет, по крайней мере в настоящий момент, представляет собой пример того, насколько удачно и эффективно может развиваться сложная техническая система практически в отсутствие единого управляющего центра и правового регулирования.

Имеющиеся нормативные правовые акты по отношению к Интернету можно охарактеризовать следующим образом.

1. Ни в одной стране мира нет всеобъемлющего законодательства по Интернету. Существующие нормативные (подзаконные) акты регулируют частные аспекты функционирования сети.
2. Нормы, которые можно было бы применить к правовым отношениям в Интернете, «разбросаны» по законодательным актам других отраслей права. В первую очередь они содержатся в нормах, относящихся к интеллектуальной и промышленной собственности.
3. Практически отсутствует регулирование правовых отношений в Интернете на международном (межгосударственном) уровне. Основная проблема заключается в том, что Интернет не имеет территориальных границ своего распространения.

Подходы к решению одинаковых правовых проблем могут сильно различаться в разных странах. Признания определенных действий в Интернете уголовным преступлением в одной стране может считаться вполне легальной деятельностью в другой стране. Во многих случаях интересы государства легко могут быть затронуты и действиями лиц, находящимися за его пределами.

Оценить наличие связи лица с конкретным государством весьма трудно, поскольку любой материал, размещенный в Интернете, становится одинаково доступным для лиц, находящихся в любой точке земного шара. Если лицо занимается продажей товаров или оказанием услуг через Интернет, его клиентами могут оказаться представители разных стран. Поэтому для такого лица важно иметь представление о законодательстве этих стран — всех, где находятся его клиенты, ведь возможные споры будут рассматриваться в местных судах.

Государства устанавливают свою юрисдикцию над лицом, если существует определенная связь между этим лицом и территорией такого государства. Наиболее очевидна связь с территорией при размещении информации на определенном сервере, делающем доступным эту информацию для пользователей Интернета.

Однако страна размещения сервера может не совпадать со страной регистрации доменного имени, причем для владельца доменного имени не составит особого труда заменить один компьютер, использующий в Интернете это доменное имя, другим, находящимся за тысячи километров от первого.

Оптимальным решением указанных проблем стала бы как можно более полная унификация национальных и региональных законодательств в области Интернета.

Контрольные вопросы

1. В чем заключаются основные правовые проблемы в Интернете?

6.2. Этика в Интернете

Достоинство Интернета в том, что он расширяет среду общения до размеров земного шара. Написал несколько слов, нажал на кнопку — письмо ушло, хоть соседу по парте, хоть президенту США. В этой простоте кроются опасности: возникает иллюзия доступности и вседозволенности, которая еще усугубляется тем, что при электронном общении мы не видим лица собеседника и не всегда можем правильно понять его реакцию.

Сетевой этикет — понятие, возникшее с появлением электронной почты. Интернет развивается, форумы и телеконференции позволяют общаться, не только обмениваясь почтовыми сообщениями, рассылаемыми подписчикам, но и просто оставляя сообщения на специальных сайтах в Интернете, которые можно просматривать стандартными браузерами.

В дополнение к электронной почте появились и стали популярными и другие средства общения: чаты, обеспечивающие возможность беседовать в режиме реального времени или, как говорят, в режиме он-лайн (от англ. on-line) людям, удаленным друг от друга на тысячи километров. Некоторые из чатов позволяют слышать и видеть друг друга.

Для того чтобы, общаясь в Интернете с использованием различных его сервисов, не доставлять неприятностей собеседникам и не иметь их самому, полезно следовать некоторым несложным правилам, называемым сетевым этикетом. Правила сетевого этикета просты и похожи на правила поведения в реальной жизни.

Правила этикета для электронной почты. Всегда заполняйте поле «Тема» своего письма. Ориентируясь по темам, проще выделить нужные письма в большом списке поступающей корреспонденции, а также отфильтровать спам (от *англ.* spam) — навязчивую рекламу. Отвечая на чье-либо письмо, в поле темы принято вписывать Re: Исходная_тема. Большинство программ для работы с почтой эту фразу вписывают автоматически.

Отвечая на пришедшее письмо, принято цитировать некоторые его отрывки. Цитируемые фразы нужно выделять каким-нибудь символом, обычно это «>», и отделять их от ваших фраз пустой строкой. Большинство программ для работы с электронной почтой при написании ответа сами выделяют текст исходного письма подобными знаками, и пользователю достаточно лишь удалить ненужные фразы.

В Интернете считается, что фраза, написанная ЗАГЛАВНЫМИ БУКВАМИ, означает, что автор громко кричит. К тому же предложения, написанные буквами верхнего регистра, плохо читаются, что создает неудобства вашему корреспонденту. В связи с этим используйте заглавные буквы, только если это является действительно необходимым.

В официальной переписке принято использовать подпись, содержащую некоторую информацию об авторе пись-

ма: полное имя, должность или другие регалии, контактную информацию. Поскольку во многих случаях программы для работы с электронной почтой автоматически добавляют к письму однажды подготовленную подпись, будьте внимательны и не помещайте в подпись информацию, которую вы не хотели бы показывать всем своим корреспондентам.

Старайтесь не допускать грамматических ошибок. Многие современные текстовые редакторы и программы для работы с электронной почтой имеют встроенные системы проверки правописания. В любом случае полезно перед отправкой еще раз перечитать письмо.

Не вставляйте в электронное письмо файлы большого объема, не узнав у вашего корреспондента, принимает ли такие вложения его почтовый сервер. Если есть проблема с объемом и количеством приложенных файлов, большие файлы разбейте на несколько меньших и разошлите их отдельными письмами.

В обычной (не деловой) переписке часто используются смайлики — комбинации текстовых символов, например :-), которые напоминают лицо, если смотреть на них, повернув голову набок. Использование смайликов способно придать письму живой характер и даже заменить жестикуляцию. Но не следует этим злоупотреблять — это будет уже плохим тоном. Смайликов придумано очень много, наиболее часто используются такие:

- :-) или :) — улыбка; обычно используется для выражения радости;
- :-(или :(— несчастное лицо; выражает сожаление или разочарование;
- ;-) или ;) — подмигивающее лицо; обычно выражает иронию и означает, что слова не следует понимать слишком буквально.

Правила этикета для общения в чате, форуме, телеконференции. В чате, форуме, гостевой книге общается большое количество разных людей, с разными мнениями и интересами. Следует быть тактичным и корректным в своих высказываниях. Не нападайте на человека только из-за того, что его мнение не совпало с вашим.

Выбирайте себе псевдоним, или ник (от англ. nickname — прозвище, кличка), не оскорбляющий других участников чата. Избегайте нейтральных имен, не дающих возможности представить кто вы: женщина или мужчина.

Обращаясь к кому-либо, пишите его ник в начале вашей фразы.

Не повторяйте многократно одну и ту же фразу, не забивайте «эфир».

Как и в электронной почте, текст, написанный заглавными буквами или с большим количеством восклицательных знаков (например, ПОМОГИТЕ!!!!), интерпретируется как громкий крик, поэтому не нужно злоупотреблять этим средством выделения своих сообщений.

Используйте смайлики, но не злоупотребляйте ими. Не факт, что экзотический смайлик знаком участникам общения.

Уходя из чата, не забудьте попрощаться с вашими собеседниками и, возможно, договориться о времени следующей беседы.

Контрольные вопросы

1. Каковы основные этические правила при общении по электронной почте?
2. Каковы основные этические правила при общении чатах и форумах?

6.3. Перспективы развития информационных и коммуникационных технологий

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 (рис. 6.8). По оси абсцисс отложены этапы. 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 отло-

жен уровень оценки технологии обществом, что носит довольно субъективный характер.

Первый этап. «Восход надежд», время теоретических разработок и первых экспериментальных реализаций новой информационной или коммуникационной технологии. Разработчикам и экспертам кажется, что данная новая технология разрешит многие проблемы развития информационных технологий.

Примерами таких технологий являются разработки транзисторов молекулярных и атомных размеров. Действительно, современные транзисторы уже имеют размеры в несколько десятков атомов, и дальнейшая миниатюризация должна строится на новой основе.

Основой квантового компьютера может стать любая квантовая частица, обладающая двумя состояниями (логические 0 и 1). Например, это может быть спин электрона, имеющий два состояния (вниз и вверх), основное и возбужденное состояния атома и другие объекты, подчиняющиеся законам квантовой механики (рис. 6.1). Первые теоретические разработки квантовых компьютеров начались около 20 лет назад, проведены успешные лабораторные опыты по созданию элементов таких компьютеров. Однако предсказать сроки появления квантовых компьютеров сейчас невозможно.

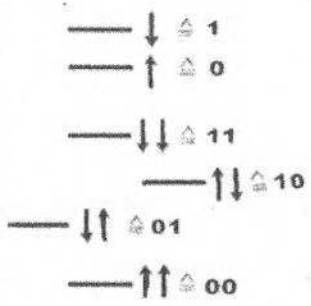


Рис. 6.1. Квантовый компьютер (квантовый бит — это спин электрона или ядра атома)

ДНК-вычисления предполагают создание новых алгоритмов вычислений на основе знаний о строении и функциях молекулы ДНК. Так же, как и любой другой процессор, ДНК-процессор характеризуется структурой и набором команд. В нашем случае структура процессора — это структура молекулы ДНК (рис. 6.2). А набор команд — это перечень биохимических операций с молекулами.

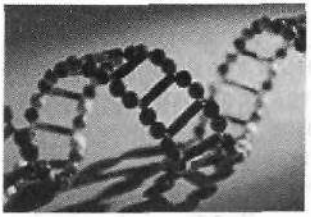


Рис. 6.2. Модель ДНК-процессора

На базе ДНК-вычислений ведется разработка нанокомпьютера, который можно будет вживлять в клетку организма и производительность которого будет исчисляться миллиардами операций в секунду при энергопотреблении не более одной миллиардной ватта. В настоящее время ДНК-вычисления находятся на стадии лабораторных исследований, поэтому создание биологического компьютера прогнозируется только через несколько десятков лет.

Молекулярный транзистор (рис. 6.3) — это молекула, которая может существовать в двух устойчивых состояниях, обладающих разными свойствами (логические 0 и 1). Транзистор на одной молекуле в десятки раз меньше современных транзисторов. Переводить молекулу из одного состояния в другое можно с помощью света, тепла, магнитного поля и других физических воздействий. Уже в настоящее время существуют логические схемы на молекулярных транзисторах и, планируется, что уже в ближайшее десятилетие начнется их промышленное производство.

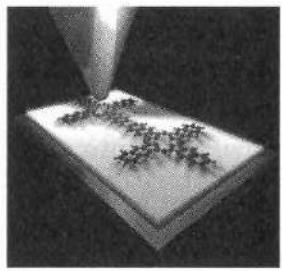


Рис. 6.3. Молекулярный транзистор

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 являются «электронные чернила» (рис. 6.4). В ходе многолетних исследований удалось создать тип устройств визуализации информации, которые обладают механическими свойствами обычной бумаги (например, их можно свертывать в рулон).

Базовыми элементами таких устройств являются микрокапсулы (пиксели), заполненные микрочастицами двух цветов: белого и черного. Слой микрокапсул расположен между двумя прозрачными и гибкими электродами. При подаче напряжения определенной полярности, микрочастицы белого цвета собираются в верхней части капсулы, а микрочастицы черного цвета — в нижней части. При перемене полярности напряжения все происходит наоборот. Так формируется черно-белое изображение. Однако существенным недостатком таких устройств является большое время

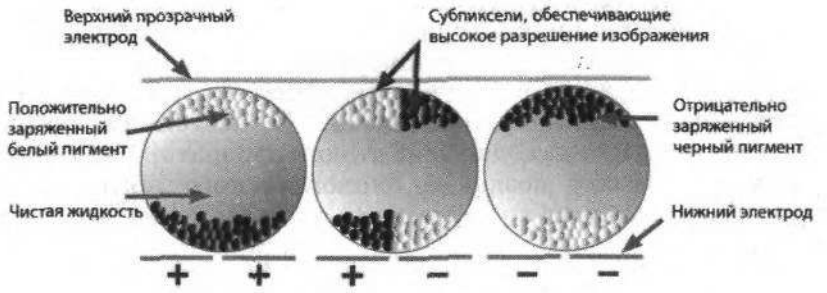


Рис. 6.4. Электронные чернила

переключения пикселей (около 1 с), что препятствует их широкому промышленному производству.

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.

Компактные топливные элементы (рис. 6.5) предназначены для прямого преобразования энергии, высвобождающейся в ходе реакции окисления топлива, в электрическую энергию. В отличие от аккумуляторов, заряд которых возобновляется при подключении к внешнему источнику тока, восстановление работоспособности топливных элементов осуществляется путем пополнения запаса топлива.

Рис. 6.5. Компактный топливный элемент

Однако у топливных элементов обнаружились серьезные недостатки: проблема зарядки топливом, высокая температура топливного элемента при работе. Все это откладывает массовое промышленное производство топливных элементов.

Четвертый этап. «Подъем жизнестойкости», когда на основе новых исследований оптимизируется технологический процесс и начинается массовое серийное производство.

Примером такой технологии является машинный перевод. Системы машинного перевода (рис. 6.6) получили ши-

рокое распространение и дают приемлемое качество перевода. С помощью систем машинного перевода можно переводить тексты как off-line, так и on-line (Web-страницы и письма электронной почты). Кроме того, расширился набор языков и направлений перевода.

Рис. 6.6. Система машинного перевода

Пятый этап. «Плато продуктивности», когда массовое серийное производство изделий по новой технологии находит массовый устойчивый спрос потребителей и приносит стабильную прибыль производителям.

Определение местоположения на поверхности земли стало широко применяться в спутниковых системах (GPS — США или ГЛОНАСС — Россия). Для этого запущено требуемое количество спутников и развернуто массовое промышленное производство приемников спутникового сигнала (с нескольких спутников). На экране такого приемника (рис. 6.7) отображаются карты местности с указанием местоположения. Точность такого определения местоположения в открытом гражданском секторе составляет несколько десятков метров, а в закрытом военном — несколько метров.

Рис. 6.7. Приемник определения местоположения

Определение местоположения предоставляют и операторы мобильной связи, правда, точность определения местоположения составляет обычно несколько сот метров и зависит от количества и расположения базовых станций.

Рис. 6.8. Развитие информационных и коммуникационных технологий

Контрольные вопросы

1. Назовите информационные и коммуникационные технологии, соответствующие различным этапам развития технологии.

Глава 7

Подготовка к ЕГЭ. Тесты по темам курса «Информатика и ИКТ»

1. На выполнение экзаменационной работы в форме тестов по курсу «Информатика и ИКТ» отводится 4 часа (240 минут), включая работу на компьютере.
2. Тесты включают 160 заданий, которые делятся на 4 части (A, B, C и D), аналогично Единому государственному экзамену (ЕГЭ):
   - Часть A (ВО — выборочный ответ). Задание считается выполненным, если учащийся записал номер верного варианта ответа.
   - Часть B (КО — краткий ответ). Задание с кратким ответом считается выполненным, если учащийся дал ответ (число, значение переменной, путь к файлу или логическое значение выражения), соответствующий верному варианту ответа.
   - Часть C (РО — развернутый ответ). Задание с развернутым ответом считается выполненным, если учащийся правильно записал последовательность преобразований логического выражения или программу.
   - Часть D (ПЗ — практическое задание). Практическое задание считается выполненным, если файл задания, сохраненный учащимся, соответствует заданному эталону.
3. Тесты распределены по темам курса «Информатика и ИКТ».
4. Установлены следующие уровни сложности заданий:
   Б — базовый,
   П — повышенный,
   В — высокий.
5. Практические задания должны выполняться на базе классов современных компьютеров с установленными операционной системой Windows 95/98/Me/2000/XP/Vista, интегрированными офисными приложениями Microsoft Office или OpenOffice, системами программиро-

вания (например, VisualStudio 2005 Express Edition и Delphi), графическим редактором (например, GIMP) и системой компьютерного черчения КОМПАС.

6. Файлы-задания, необходимые для выполнения практических заданий, размещены на диске Windows-CD в папке ..\Test\.

Тема 1. Информация. Кодирование информации

1.1. Единицы измерения количества информации

1.1.1. За минимальную единицу измерения количества информации принят:
1) 1 бод; 2) 1 пиксель; 3) 1 байт;
4) 1 бит.

1.1.2. Чему равен 1 байт?
1) 2^3 битов; 2) 10^3 битов; 3) 2^{10} битов; 4) 10^{10} битов.

1.1.3. Сколько битов в 1 килобайте?
1) 1000 битов;
2) $8 \cdot 2^{10}$ битов;
3) 1024 бита;
4) $8 \cdot 10^3$ битов.

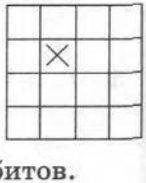
1.1.4. Чему равен 1 мегабайт?
1) 10^6 битов;
2) 10^6 байтов;
3) 2^{10} Кбайт;
4) 2^{10} байтов.

1.2. Определение количества информации (вероятностный подход)

1.2.1. В рулетке общее количество лунок равно 32. Какое количество информации (с точки зрения вероятностного подхода) мы получаем в зрительном сообщении об остановке шарика в одной из лунок?
1) 8 битов; 2) 5 битов; 3) 2 бита; 4) 1 бит.

1.2.2. Производится бросание симметричной четырехгранной пирамидки. Какое количество информации (с точки зрения вероятностного подхода) мы получаем в зрительном сообщении о ее падении на одну из граней?
1) 1 бит; 2) 2 бита; 3) 4 бита; 4) 8 битов.

1.2.3. Какое количество информации (с точки зрения вероятностного подхода) получит второй игрок при игре в крестики-нолики на поле 4×4 после первого хода первого игрока, играющего крестиками?



1) 1 бит; 2) 2 бита; 3) 4 бита; 4) 8 битов.

1.2.4. Какое количество информации (с точки зрения вероятностного подхода) получит при игре в шахматы играющий черными после первого хода белых (при условии, что ходить конями запрещено)?

1) 1 бит; 2) 2 бита; 3) 4 бита; 4) 1 байт.

1.3. Определение количества информации (алфавитный подход)

1.3.1. Какое количество информации (с точки зрения алфавитного подхода) содержит двоичное число 101_2?
1) 3 байта; 2) 2 байта; 3) 3 бита; 4) 2 бита.

1.3.2. Какое количество информации (с точки зрения алфавитного подхода) содержит восьмеричное число 55_8?
1) 10 битов; 2) 8 битов; 3) 6 битов; 4) 5 битов.

1.3.3. Какое количество информации (с точки зрения алфавитного подхода) содержит шестнадцатеричное число AB_{16}?
1) 16 битов; 2) 8 битов; 3) 4 бита; 4) 2 бита.

1.3.4. Какое количество информации (с точки зрения алфавитного подхода) содержит слово «информатика», если считать, что алфавит состоит из 32 букв?
1) 55 битов; 2) 55 байтов; 3) 11 битов; 4) 11 байтов.

1.4. Кодирование текстовой информации

1.4.1. Во сколько раз увеличится информационный объем страницы текста (текст не содержит управляющих символов форматирования) при его преобразовании из кодировки Windows (таблица кодировки содержит 256 символов) в кодировку Unicode (таблица кодировки содержит 65 536 символов)?
1) в 2 раза; 2) в 8 раз; 3) в 16 раз; 4) в 256 раз.

1.4.2. Во сколько раз уменьшится информационный объем страницы текста (текст не содержит управляющих символов форматирования) при его преобразовании

из кодировки Unicode (таблица кодировки содержит 65 536 символов) в кодировку Windows (таблица кодировки содержит 256 символов)?

1) в 256 раз; 3) в 4 раза;
2) в 8 раз; 4) в 2 раза.

1.4.3. Какое количество информации необходимо для кодирования каждого из 256 символов алфавита?

1) 256 битов; 3) 8 битов;
2) 16 битов; 4) 4 бита.

1.4.4. Какое количество информации необходимо для кодирования каждого из 65 536 символов алфавита?

1) 1 байт; 2) 2 байта; 3) 8 битов; 4) 32 бита.

1.5. Кодирование графической информации

1.5.1. Черно-белое (без градаций серого цвета) растровое графическое изображение имеет размер 10×10 точек. Какой объем памяти займет это изображение?

1) 100 битов; 3) 1000 битов;
2) 100 байтов; 4) 1000 байтов.

1.5.2. Цветное (с палитрой из 256 цветов) растровое графическое изображение имеет размер 10×10 точек. Какой объем памяти займет это изображение?

1) 100 битов; 3) 100 байтов;
2) 800 битов; 4) 800 байтов.

1.5.3. В процессе преобразования растрового графического изображения количество цветов уменьшилось с 65 536 до 16. Во сколько раз уменьшился информационный объем графического файла?

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

1.5.4. В процессе преобразования растрового графического изображения количество цветов увеличилось с 256 до 65 536. Во сколько раз увеличился информационный объем графического файла?

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

1.6. Кодирование звуковой информации

1.6.1. Аналоговый звуковой сигнал был дискретизирован сначала с использованием 65 536 уровней интенсивности сигнала (качество звучания аудио-CD), а затем — с использованием 256 уровней интенсивности сигнала (качество звучания радиотрансляции). Во

сколько раз различаются информационные объемы оцифрованных звуковых сигналов?
1) в 256 раз; 2) в 16 раз; 3) в 8 раз; 4) в 2 раза.

1.6.2. Звуковая плата реализует 16-битовое двоичное кодирование аналогового звукового сигнала. Это позволяет воспроизводить звук с:
1) 8 уровнями интенсивности;
2) 16 уровнями интенсивности;
3) 256 уровнями интенсивности;
4) 65 536 уровнями интенсивности.

1.6.3. Звуковая плата производит двоичное кодирование аналогового звукового сигнала. Какое количество информации необходимо для кодирования каждого из 65 536 возможных уровней интенсивности сигнала?
1) 256 битов; 2) 16 битов; 3) 8 битов; 4) 1 бит.

1.6.4. Звуковая плата реализует 8-битовое двоичное кодирование аналогового звукового сигнала. Это позволяет воспроизводить звук с:
1) 8 уровнями интенсивности;
2) 16 уровнями интенсивности;
3) 256 уровнями интенсивности;
4) 65 536 уровнями интенсивности.

1.7. Представление числовой информации. Сложение чисел в двоичной и десятичной системах счисления

1.7.1. Вычислить сумму двоичного и десятичного чисел $10_2 + 10_{10}$. Представить результат в десятичной системе счисления.
1) 11_{10}; 2) 12_{10}; 3) 13_{10}; 4) 14_{10}.

1.7.2. Вычислить сумму двоичного и десятичного чисел $10_2 + 10_{10}$. Представить результат в двоичной системе счисления.
1) 1000_2; 2) 1100_2; 3) 1110_2; 4) 1111_2.

1.7.3. Вычислить сумму двоичного и десятичного чисел $11_2 + 11_{10}$. Представить результат в двоичной системе счисления.
1) 1000_2; 2) 1100_2; 3) 1110_2; 4) 1111_2.

1.7.4. Вычислить сумму двоичного и десятичного чисел $11_2 + 11_{10}$. Представить результат в десятичной системе счисления.
1) 12_{10}; 2) 13_{10}; 3) 14_{10}; 4) 15_{10}.

1.8. Представление числовой информации. Сложение чисел в двоичной, восьмеричной, десятичной и шестнадцатеричной системах счисления.

1.8.1. Вычислить сумму чисел $11_2 + 11_8 + 11_{10} + 11_{16} = \_\_\_\_2$. Представить результат в двоичной системе счисления.

1.8.2. Вычислить сумму чисел $11_2 + 11_8 + 11_{10} + 11_{16} = \_\_\_\_10$. Представить результат в десятичной системе счисления.

1.8.3. Вычислить сумму чисел $11_2 + 11_8 + 11_{10} + 11_{16} = \_\_\_\_8$. Представить результат в восьмеричной системе счисления.

1.8.4. Вычислить сумму чисел $11_2 + 11_8 + 11_{10} + 11_{16} = \_\_\_\_16$. Представить результат в шестнадцатеричной системе счисления.

Тема 2. Устройство компьютера и программное обеспечение

2.1. Устройство компьютера

2.1.1. Драйвер — это:
1) устройство компьютера;
2) компьютерный вирус;
3) программа, обеспечивающая работу устройства компьютера;
4) антивирусная программа.

2.1.2. При выключении компьютера вся информация теряется:
1) на гибком диске;
2) на жестком диске;
3) на диске CD-ROM;
4) в оперативной памяти.

2.1.3. Программа может управлять работой компьютера, если она находится:
1) на гибком диске;
2) на жестком диске;
3) на диске CD-ROM;
4) в оперативной памяти.

2.1.4. Процессор обрабатывает информацию, представленную:
1) в десятичной системе счисления;
2) на языке программирования высокого уровня;
3) на алгоритмическом языке;
4) на машинном языке (в двоичном коде).

2.2. Безопасность и технические условия эксплуатации

2.2.1. В целях сохранения информации жесткие магнитные диски необходимо оберегать от:
1) пониженной температуры; 3) света;
2) царапин; 4) ударов при установке.

2.2.2. В целях сохранения информации гибкие магнитные диски необходимо оберегать от:
1) пониженной температуры;
2) магнитных полей;
3) света;
4) перепадов атмосферного давления.

2.2.3. В целях сохранения информации оптические CD- и DVD-диски необходимо оберегать от:
1) пониженной температуры;
2) магнитных полей;
3) света;
4) загрязнений.

2.2.4. В целях сохранения нормальной работоспособности модули оперативной памяти необходимо оберегать от:
1) электростатических зарядов при установке;
2) магнитных полей;
3) света;
4) загрязнений.

2.3. Операционная система: назначение и функциональные возможности

2.3.1. Операционная система — это:
1) программа, обеспечивающая управление базами данных;
2) антивирусная программа;
3) программа, управляющая работой компьютера;
4) система программирования.

2.3.2. Процесс загрузки операционной системы представляет собой:
1) копирование файлов операционной системы с гибкого диска на жесткий диск;
2) копирование файлов операционной системы с CD-диска на жесткий диск;
3) последовательную загрузку файлов операционной системы в оперативную память ;
4) копирование содержимого оперативной памяти на жесткий диск.

2.3.3. Системный диск необходим для:
1) загрузки операционной системы;
2) хранения важных файлов;
3) систематизации файлов;
4) лечения компьютера от вирусов.

2.3.4. В логический раздел диска одновременно может быть установлено:
1) несколько различных операционных систем;
2) несколько копий одной операционной системы;
3) только одна операционная система;
4) фрагменты различных операционных систем.

2.4. Архитектура компьютера

2.4.1. Какова пропускная способность системной шины (с точностью до целых), если ее разрядность составляет 64 бита, а частота — 1066 МГц?

2.4.2. Какова пропускная способность шины памяти (с точностью до целых), если ее разрядность составляет 64 бита, а частота — 533 МГц?

2.4.3. Какова пропускная способность шины AGP (с точностью до целых), если ее разрядность составляет 32 бита, а частота — 528 МГц?

2.4.4. Какова пропускная способность шины PCI (с точностью до целых), если ее разрядность составляет 64 бита, а частота — 66 МГц?

2.5. Файлы и файловые системы

2.5.1. Файл — это:
1) единица измерения количества информации;
2) программа или данные на диске, имеющие имя;
3) программа в оперативной памяти;
4) текст, распечатанный на принтере.

2.5.2. При полном форматировании гибкого диска:
1) стираются все данные;
2) производится только очистка каталога диска;
3) диск становится системным;
4) производится дефрагментация размещения файлов на диске.

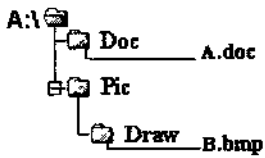
2.5.3. Разные файлы могут иметь одинаковые имена, если они:
1) имеют разные объемы;
2) созданы в различные дни;

3) созданы в различное время суток;
4) хранятся в разных папках.

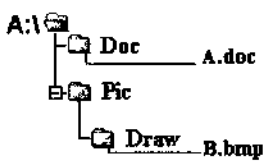
2.5.4. Информационный объем файла на гибком диске не может быть меньше, чем:
1) размер сектора диска; 3) 1 байт;
2) 1 бит; 4) 1 Кбайт.

2.6. Путь к файлу

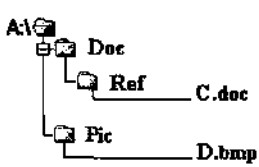
2.6.1. Записать полное имя файла B.bmp (включая путь к файлу) в иерархической файловой системе, изображенной на рисунке.



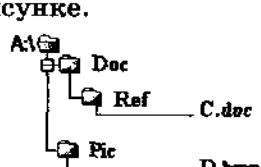
2.6.2. Записать полное имя файла A.doc (включая путь к файлу) в иерархической файловой системе, изображенной на рисунке.



2.6.3. Записать полное имя файла C.doc (включая путь к файлу) в иерархической файловой системе, изображенной на рисунке.



2.6.4. Записать полное имя файла D.bmp (включая путь к файлу) в иерархической файловой системе, изображенной на рисунке.



2.7. Защита информации

2.7.1. Характерным отличием компьютерных вирусов от других вредоносных программ является :
1) проникновение на компьютер по компьютерным сетям;
2) способность к размножению (самокопированию);
3) воровство информации;
4) сетевые атаки.

2.7.2. Характерным отличием сетевых червей от других вредоносных программ является:
1) проникновение на компьютер по компьютерным сетям;
2) способность к размножению (самокопированию);
3) воровство информации;
4) сетевые атаки.

2.7.3. Характерным отличием троянских программ от других вредоносных программ является:
1) проникновение на компьютер по компьютерным сетям;
2) способность к размножению (самокопированию);
3) воровство информации;
4) сетевые атаки.

2.7.4. Характерным отличием хакерских утилит от других вредоносных программ является:
1) проникновение на компьютер по компьютерным сетям;
2) способность к размножению (самокопированию);
3) воровство информации;
4) сетевые атаки.

Тема 3. Алгоритмизация и программирование

3.1. Основные алгоритмические структуры

3.1.1. Нарисовать блок-схему алгоритмической структуры «ветвление».

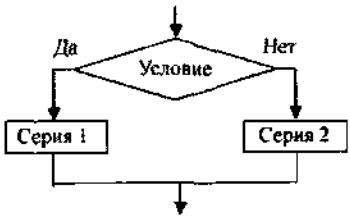
3.1.2. Нарисовать блок-схему алгоритмической структуры «выбор».

3.1.3. Нарисовать блок-схему алгоритмической структуры «цикл со счетчиком».

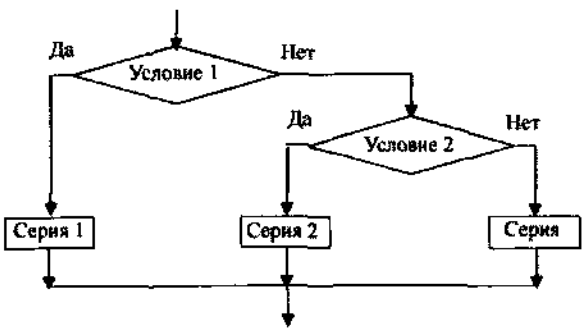
3.1.4. Нарисовать блок-схему алгоритмической структуры «цикл с предусловием».

3.2. Кодирование алгоритмических структур на языках программирования

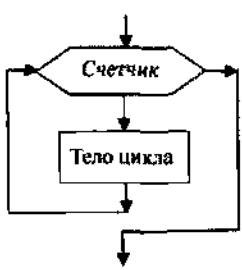
3.2.1. Записать алгоритмическую структуру «ветвление» на одном из языков программирования:
Visual Basic .NET, Visual C#, Visual J# или Delphi.



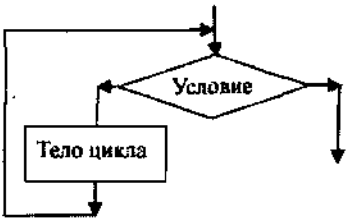
3.2.2. Записать алгоритмическую структуру «выбор» на одном из языков программирования: Visual Basic .NET, Visual C#, Visual J# или Delphi.



3.2.3. Записать алгоритмическую структуру «цикл со счетчиком» на одном из языков программирования: Visual Basic .NET, Visual C#, Visual J# или Delphi.

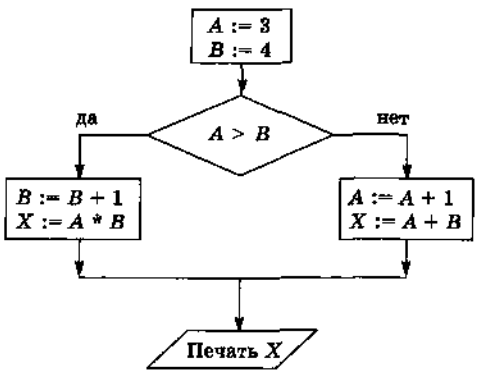


3.2.4. Записать алгоритмическую структуру «цикл с предусловием» на одном из языков программирования: Visual Basic .NET, Visual C#, Visual J# или Delphi.



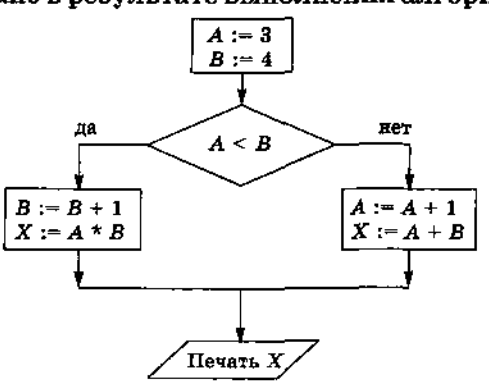
3.3. Формальное исполнение простого алгоритма

3.3.1. Фрагмент алгоритма изображен в виде блок-схемы. Определить, какое значение переменной X будет напечатано в результате выполнения алгоритма.



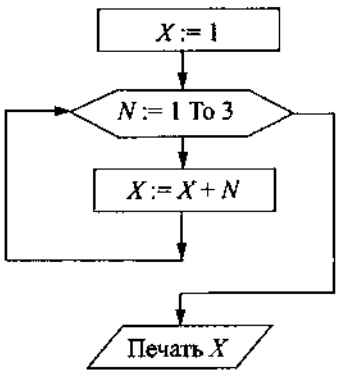
1) 8; 2) 10; 3) 15; 4) 18.

3.3.2. Фрагмент алгоритма изображен в виде блок-схемы. Определить, какое значение переменной X будет напечатано в результате выполнения алгоритма.



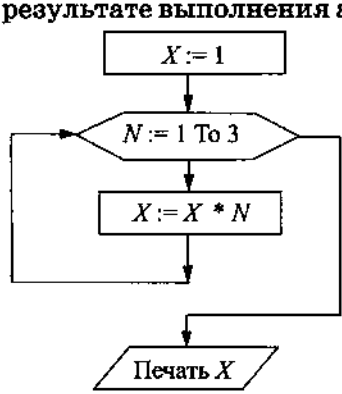
1) 8; 2) 10; 3) 15; 4) 18.

3.3.3. Фрагмент алгоритма изображен в виде блок-схемы. Определить, какое значение переменной X будет напечатано в результате выполнения алгоритма.



1) 5; 2) 6; 3) 7; 4) 8.

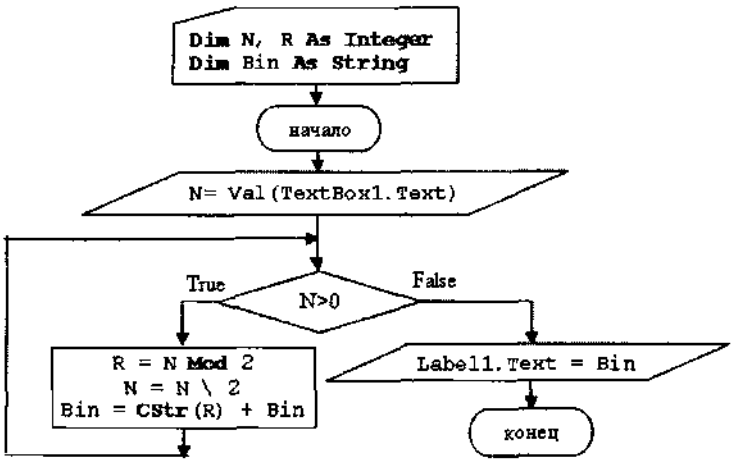
3.3.4. Фрагмент алгоритма изображен в виде блок-схемы. Определить, какое значение переменной X будет напечатано в результате выполнения алгоритма.



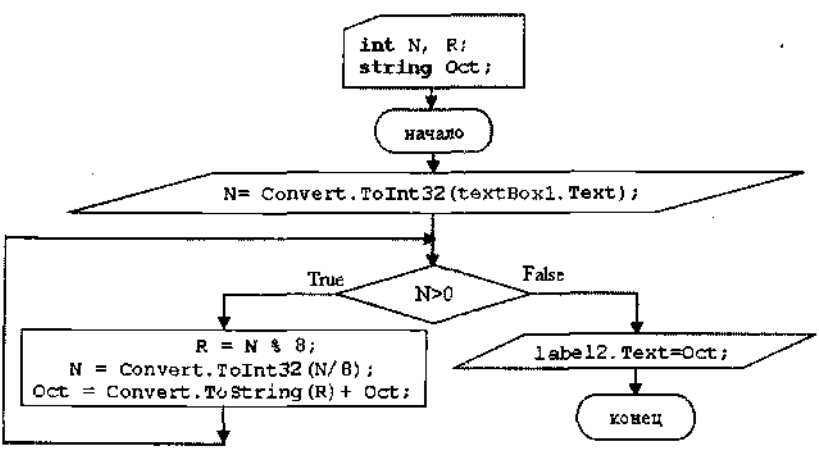
1) 3; 2) 4; 3) 5; 4) 6.

3.4. Формальное исполнение сложного алгоритма

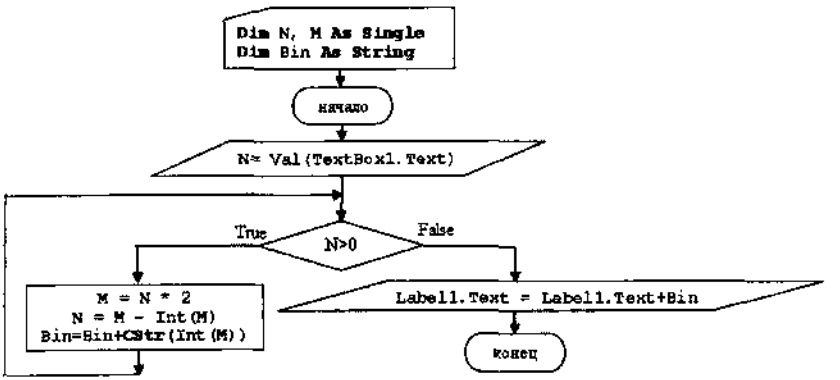
3.4.1. Алгоритм изображен в виде блок-схемы, в графических элементах которой приведен программный код на языке Visual Basic .NET. В текстовое поле введено целое десятичное число 10. Определить, какое целое двоичное число будет выведено на надпись в результате выполнения алгоритма.



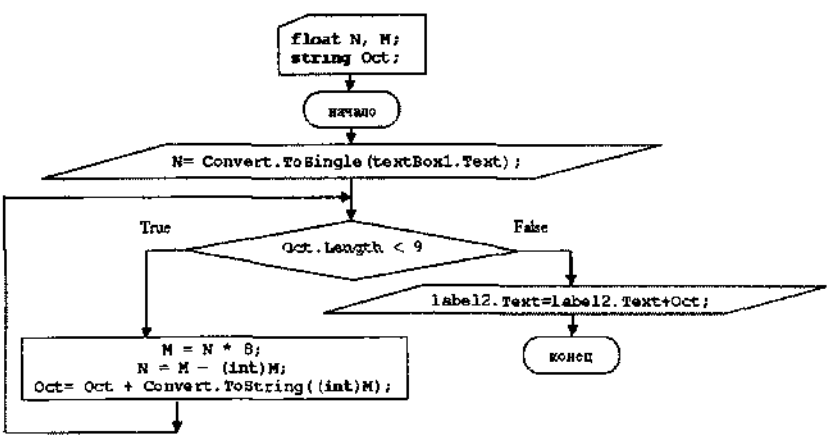
3.4.2. Алгоритм изображен в виде блок-схемы, в графических элементах которой приведен программный код на языке Visual C#. В текстовое поле введено целое десятичное число 20. Определить, какое целое восьмеричное число будет выведено на надпись в результате выполнения алгоритма.



3.4.3. Алгоритм изображен в виде блок-схемы, в графических элементах которой приведен программный код на языке Visual Basic .NET. В текстовое поле введено дробное десятичное число 0.25. Определить, какое дробное двоичное число будет выведено на надпись в результате выполнения алгоритма.



3.4.4. Алгоритм изображен в виде блок-схемы, в графических элементах которой приведен программный код на языке Visual C#. В текстовое поле введено дробное десятичное число 0.25. Определить, какое дробное восьмеричное число будет выведено на надпись в результате выполнения алгоритма.



3.5. Переменные

3.5.1. Имя переменной определяет:

1) данные, хранящиеся в выделенной области оперативной памяти;
2) выделенную область оперативной памяти;
3) количество выделяемых ячеек оперативной памяти;
4) диапазон значений переменной.

3.5.2. Значение переменной определяет:
1) данные, хранящиеся в выделенной области оперативной памяти;
2) выделенную область оперативной памяти;
3) количество выделяемых ячеек оперативной памяти;
4) диапазон значений переменной.

3.5.3. Тип переменной определяет:
1) данные, хранящиеся в выделенной области оперативной памяти;
2) выделенную область оперативной памяти;
3) количество выделяемых ячеек оперативной памяти;
4) присваивание переменной значения.

3.5.4. Присваивание переменной значения приводит:
1) к изменению данных, хранящихся в выделенной области оперативной памяти;
2) к изменению выделенной области оперативной памяти;
3) к изменению количества выделяемых ячеек оперативной памяти;
4) к изменению диапазона значений переменной.

3.6. Результаты выполнения программы на языках программирования

3.6.1. Приведены программы на языках объектно-ориентированного программирования Visual Basic .NET и Delphi. В текстовые поля введены десятичные числа 5 и 3. Определить число, которое будет выведено на надпись в результате выполнения алгоритма.

Visual Basic .NET

```
Dim A, B, Max As Single
Private Sub
Button1_Click(…)
A=Val(TextBox1.Text)
B=Val(TextBox2.Text)
If A>=B Then
Max=A
Else
Max=B
End If
Label1.Text=Max
End Sub
```

Delphi

```
var
A, B: string;
Max: string;
procedure TForm1.Button1Click
 (Sender: TObject);
begin
A:=EditA.Text;
B:=EditB.Text;
If A>=B Then Max:=A
Else Max:=B;
Label1.Caption:=Max;
end;
```

3.6.2. Приведены программы на языках объектно-ориентированного программирования Visual Basic .NET и Delphi. В текстовые поля введены слово "информатика" и символ "а". Определить число, которое будет выведено на надпись в результате выполнения алгоритма.

Visual Basic .NET	Delphi

`Dim N, K As Byte`
`Private Sub`
`Button1_Click(…)`
`N=1`
`K=0`
`Do While`
`N<=Len(TextBox1.Text)`
`S=Mid(TextBox1.Text,`
`N, 1)`
`If S=TextBox2.Text`
`Then K=K+1`
`N=N+1`
`Loop`
`Label1.Text=K`
`End Sub`
	`var`
	`N, K: integer;`
	`A, B, M: string;`
	`procedure TForm1.Button1Click`
	`(Sender: TObject);`
	`begin`
	`N:=0;`
	`K:=0;`
	`A:=Edit1.Text;`
	`B:=Edit2.Text;`
	`While N<Length(A) Do`
	`begin`
	`N:=N+1;`
	`M:=Copy(Edit1.Text,N,1);`
	`If M=B Then K:=K+1;`
	`end;`
	`Label1.Caption:=IntToStr(K);`
	`end;`

3.6.3. Приведены программы на языках объектно-ориентированного программирования Visual Basic .NET и Delphi. В текстовое поле введено число 5. Определить, какое число будет выведено на надпись в результате выполнения алгоритма.

Visual Basic .NET	Delphi

`Dim N As Byte, I As`
`Integer, F As Decimal`
`Private Sub`
`TextBox1_TextChanged(…)`
`N=Val(TextBox1.Text)`
`F=1`
`I=1`
`Do`
`F=F*I`
`I=I+1`
`Loop While I<=N`
`Label1.Text= F`
`End Sub`
	`var`
	`N, I: integer;`
	`F: int64;`
	`procedure TForm1.Edit1Change`
	`(Sender: TObject);`
	`begin`
	`N:=StrToInt(Edit1.Text);`
	`F:=1;`
	`I:=1;`
	`Repeat`
	`F:=F*I;`
	`I:=I+1;`
	`Until I>N;`
	`Label1.Caption:=IntToStr(F);`
	`end;`

3.6.4. Приведены программы на языках программирования Visual Basic .NET и Delphi. Определить, что будет выведено на надпись в результате выполнения алгоритма.

Visual Basic .NET

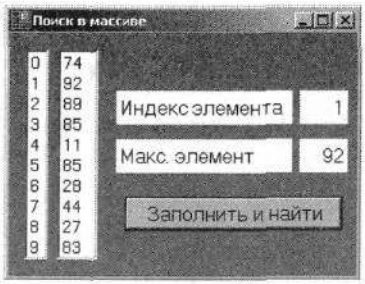
```
Dim L1, L2, L3
  As Boolean
Private Sub
    Button1_Click(…)
L1=5>3
L2=2*2=5
L3=L1 And L2
Label1.Text=L3
End Sub
```

Delphi

```
var
A, B, C: boolean;
procedure TForm1.Button1Click
  (Sender: TObject);
begin
A:=5>3;
B:=2*2=5;
C:=A and B;
Label1.Caption:=BoolToStr(C,True);
end;
```

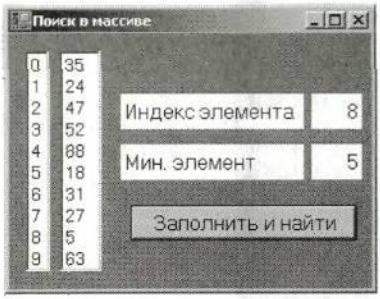
3.7. Составление программ на языках программирования

3.7.1. Составить программу заполнения массива случайными числами и поиска в массиве максимального элемента на одном из языков программирования Visual Basic .NET, Visual C#, Visual J# или Delphi. Индексы элементов массива и сами массивы выводятся в элементы управления списки, а индекс максимального элемента и сам максимальный элемент выводятся на надписи.

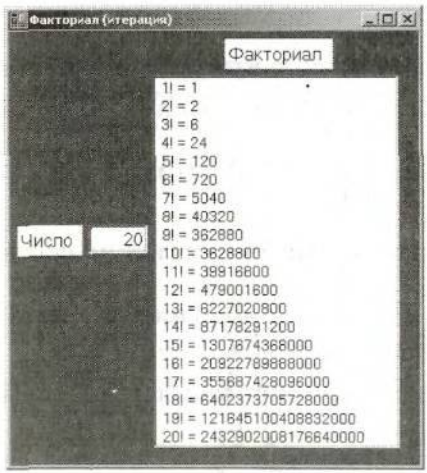


3.7.2. Составить программу заполнения массива случайными числами и поиска в массиве минимального элемента на одном из языков программирования Visual Basic .NET, Visual C#, Visual J# или Delphi. Индексы элементов массива и сами массивы выводятся в элементы

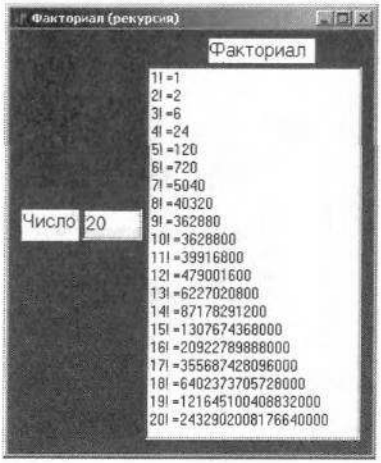
управления списки, а индекс минимального элемента и сам минимальный элемент выводятся на надписи.



3.7.3. Составить программу вычисления факториала числа итерационным методом с использованием цикла со счетчиком на одном из языков программирования Visual Basic .NET, Visual C#, Visual J# или Delphi. Число вводить в элемент управления текстовое поле, а каждый шаг вычисления факториала выводить в элемент управления список.



3.7.4. Составить программу вычисления факториала числа методом рекурсии на одном из языков программирования Visual Basic .NET, Visual C#, Visual J# или Delphi. Число вводить в элемент управления текстовое поле, а каждый шаг вычисления факториала выводить в элемент управления список.



Тема 4. Основы логики и логические основы компьютера

4.1. Определение истинности высказывания

4.1.1. Определить, истинно или ложно составное высказывание:

$A = \{(2 \times 2 = 4$ **или** $3 \times 3 = 10)$ **и** $(2 \times 2 = 5$ **или** $3 \times 3 = 9)\}$

4.1.2. Определить, истинно или ложно составное высказывание:

$A = \{(2 \times 2 = 4$ **и** $3 \times 3 = 10)$ **или** $(2 \times 2 = 5$ **и** $3 \times 3 = 9)\}$

4.1.3. Определить, истинно или ложно составное высказывание:

$A = \{(2 \times 2 = 4$ **или** $3 \times 3 = 10)$ **или** $(2 \times 2 = 5$ **и** $3 \times 3 = 9)\}$

4.1.4. Определить, истинно или ложно составное высказывание:

$A = \{(2 \times 2 = 4$ **и** $3 \times 3 = 10)$ **или** $(2 \times 2 = 5$ **или** $3 \times 3 = 9)\}$

4.2. Построение таблиц истинности логических выражений

4.2.1. Укажите таблицу истинности, которая соответствует логической функции $F = A \,\&\, \overline{B}$.

1)

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

2)

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

3)

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

4)

A	B	F
0	0	0
0	1	0
1	0	1
1	1	0

4.2.2. Укажите таблицу истинности, которая соответствует логической функции $F = \overline{A} \,\&\, B$.

1)

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

2)

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

3)

A	B	F
0	0	0
0	1	1
1	0	0
1	1	0

4)

A	B	F
0	0	0
0	1	0
1	0	1
1	1	0

4.2.3. Укажите таблицу истинности, которая соответствует логической функции $F = A \vee \overline{B}$.

1)

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

2)

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

3)

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

4)

A	B	F
0	0	1
0	1	0
1	0	1
1	1	1

4.2.4. Укажите таблицу истинности, которая соответствует логической функции $F = \overline{A} \vee B$.

1)

A	B	F
0	0	1
0	1	1
1	0	0
1	1	1

2)

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

3)

A	B	F
0	0	0
0	1	1
1	0	0
1	1	0

4)

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

4.3. Логические выражения и их преобразование

4.3.1. Упростить логическое выражение $(A \ \& \ B) \vee (A \ \& \ \overline{B})$.

4.3.2. Упростить логическое выражение $(\overline{A} \ \& \ B) \vee (A \ \& \ B)$.

4.3.3. Упростить логическое выражение $(A \vee B) \ \& \ (\overline{A} \vee B)$.

4.3.4. Упростить логическое выражение $(A \vee B) \ \& \ (A \vee \overline{B})$.

4.4. Построение таблиц истинности логических функций в электронных таблицах

4.4.1. Построить в электронных таблицах таблицу истинности функции логического сложения (дизъюнкции).

A	B	A ∨ B
0	0	0
0	1	1
1	0	1
1	1	1

4.4.2. Построить в электронных таблицах таблицу истинности функции логического умножения (конъюнкции).

A	B	A & B
0	0	0
0	1	0
1	0	0
1	1	1

4.4.3. Построить в электронных таблицах таблицу истинности функции логического следования (импликации), которая равносильна логическому выражению $\overline{A} \vee B$.

A	B	F = A → B
0	0	1
0	1	1
1	0	0
1	1	1

4.4.4. Построить в электронных таблицах таблицу истинности функции логического равенства (эквивалентности), которая равносильна логическому выражению $(A \ \& \ \overline{B}) \vee (\overline{A} \ \& \ B)$.

A	B	F = A ~ B
0	0	1
0	1	0
1	0	0
1	1	1

4.5. Построение логической схемы по логической функции

4.5.1. Построить логическую схему для логической функции $F(A, B) = A \vee B$.

4.5.2. Построить логическую схему для логической функции $F(A, B) = A \ \& \ B$.

4.5.3. Построить логическую схему для логической функции $F(A, B) = A \ \& \ \overline{B}$.

4.5.4. Построить логическую схему для логической функции $F(A, B) = \overline{A} \vee B$.

Тема 5. Моделирование и формализация

5.1. Информационные модели

5.1.1. Информационной (знаковой) моделью является:
1) анатомический муляж;
2) макет здания;
3) модель корабля;
4) химическая формула.

5.1.2. Материальной моделью является:
1) анатомический муляж;
2) техническое описание компьютера;
3) рисунок функциональной схемы компьютера;
4) программа на языке программирования.

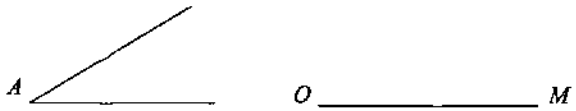
5.1.3. Какие пары объектов находятся в отношении «объект – модель»?
1) компьютер – данные;
2) компьютер – его функциональная схема;

3) компьютер – программа;
4) компьютер – алгоритм.

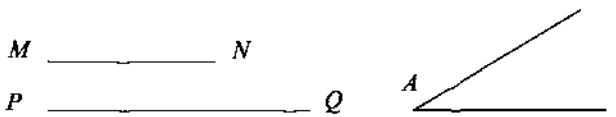
5.1.4. Какая модель является статической (описывающей состояние объекта)?
1) формула равноускоренного движения;
2) формула химической реакции;
3) формула химического соединения;
4) второй закон Ньютона.

5.2. Создание геометрических моделей

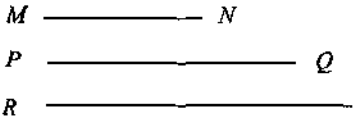
5.2.1. Выполнить в системе компьютерного черчения КОМПАС геометрическое построение «с помощью циркуля и линейки». Отложить от луча OM угол, равный заданному углу A.



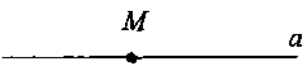
5.2.2. Выполнить в системе компьютерного черчения КОМПАС геометрическое построение «с помощью циркуля и линейки». Построить треугольник по двум сторонам и углу между ними.



5.2.3. Выполнить в системе компьютерного черчения КОМПАС геометрическое построение «с помощью циркуля и линейки». Построить треугольник по трем сторонам.



5.2.4. Выполнить в системе компьютерного черчения КОМПАС геометрическое построение «с помощью циркуля и линейки». Даны прямая и точка на ней. Построить прямую через данную точку и перпендикулярную к данной прямой.



Тема 6. Информационные технологии

6.1. Технология обработки текстовой информации. Объекты в текстовом документе

6.1.1. 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).

6.1.2. В маркированном списке для обозначения элемента списка используются:
1) латинские буквы;
2) русские буквы;
3) римские цифры;
4) графические значки.

6.1.3. В каком случае изменится иерархическая структура оглавления документа:
1) изменятся стили форматирования заголовков;
2) изменятся тексты заголовков;
3) изменятся параметры форматирования абзацев;
4) изменятся уровни заголовков?

6.1.4. Какой текст является гипертекстом:
1) текст с большим размером шрифта;
2) текст, содержащий гиперссылки;
3) текст, содержащий много страниц;
4) текст, напечатанный на большом принтере?

6.2. Технология обработки текстовой информации. Форматирование абзацев и символов по заданному образцу

6.2.1. В текстовом редакторе открыть файл-задание Text1.txt, хранящийся на диске Windows-CD в папке ..\Test. Отформатировать текст, хранящийся в файле-задании, по указанному образцу. Сохранить документ в формате, сохраняющем форматирование.

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.

6.2.2. В текстовом редакторе открыть файл-задание Text2.txt, хранящийся на диске Windows-CD в папке ..\Test. Преобразовать текст, хранящийся в файле-задании, в многоуровневый список по указанному образцу. Сохранить документ в формате, сохраняющем форматирование.

1. Первый нумерованный элемент списка (первый уровень)
   - Первый маркированный элемент списка (второй уровень)
   - Второй маркированный элемент списка (второй уровень)
2. Второй нумерованный элемент списка (первый уровень)
   - ✓ Первый маркированный элемент списка (второй уровень)
   - ✓ Второй маркированный элемент списка (второй уровень)
3. Третий нумерованный элемент списка (первый уровень)

6.2.3. В текстовом редакторе открыть файл-задание Text3.txt, хранящийся на диске Windows-CD в папке ..\Test. Преобразовать текст, хранящийся в файле-задании, в таблицу по указанному образцу. Вставить в таблицу столбец с номерами строк. Вычислить суммарную цену всех устройств. Оформить внешний вид таблицы в соответствии с образцом. Сохранить документ в формате, сохраняющем форматирование.

№	Наименование устройства	Цена (в у.е.)
1.	Системная плата	80
2.	Процессор	70
3.	Оперативная память	15
4.	Жесткий диск	100
5.	Монитор	200
6.	Дисковод 3,5"	12
7.	Дисковод CD-ROM	30
8.	Корпус	25
9.	Клавиатура	10
10.	Мышь	5
	Итого:	**547**

6.2.4. В текстовом редакторе открыть файл-задание Text4.txt, хранящийся на диске Windows-CD в папке ..\Test. Отформатировать символы строк текста по указанному образцу. Сохранить документ в формате, сохраняющем форматирование.

Times New Roman, 14, полужирный, зеленый: $N = 2^I$;
Arial, 12, курсив, синий: $10_2 + 10_{10} = 12_{10}$;
Courier New, 16, подчеркнутый, красный: A & 1 = A.

6.3. Технология обработки графической информации. Формирование цветов в различных системах цветопередачи

6.3.1. Определить цвета и внести их в таблицу, если заданы интенсивности базовых цветов в системе цветопередачи RGB.

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

6.3.2. Определить цвета и внести их в таблицу, если заданы интенсивности базовых цветов в системе цветопередачи RGB.

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

6.3.3. Определить цвета, если на бумагу нанесены краски в системе цветопередачи CMYK (C — голубой, M — пурпурный, Y — желтый).

Y + M
Y + C
M + C

6.3.4. Определить цвета, если на бумагу нанесены краски в системе цветопередачи CMYK (C — голубой, M — пурпурный, Y — желтый).

C + Y
M + Y
C + M

6.4. Технология обработки графической информации. Создание и редактирование рисунка по заданному образцу

6.4.1. В векторном графическом редакторе нарисовать модель часов (круг и две стрелки). Получить уменьшенную и увеличенную копии без потери качества изображения. Сохранить графический файл.

6.4.2. Скопировать *Рабочий стол* компьютера в растровый графический редактор, вырезать значок *Мой компьютер* и создать изображение, состоящее из пяти значков

Сохранить графический в формате, обеспечивающем минимальный информационный объем.

6.4.3. В графическом редакторе создать надпись «Информатика», растянуть ее в два раза, наклонить по вертикали на 45° и повернуть на 180°. Сохранить графический файл.

6.4.4. В графическом редакторе нарисовать функциональную схему компьютера. Сохранить графический файл.

6.5. Технология обработки числовой информации. *Абсолютные, относительные и смешанные ссылки*

6.5.1. Какой вид приобретут формулы, хранящиеся в диапазоне ячеек C1:C3 при их копировании в диапазон ячеек D2:D4?

	A	B	C	D	E	F	G
1			=A1+B1				
2			=$A$1*$B$1				
3			=$A1*B$1				
4							

6.5.2. Какой вид приобретут формулы, хранящиеся в диапазоне ячеек C1:C3 при их копировании в диапазон ячеек E2:E4?

	A	B	C	D	E	F	G
1			=A1+B1				
2			=$A$1*$B$1				
3			=$A1*B$1				
4							

6.5.3. Какой вид приобретут формулы, хранящиеся в диапазоне ячеек C1:C3 при их копировании в диапазон ячеек F2:F4?

	A	B	C	D	E	F	G
1			=A1+B1				
2			=$A$1*$B$1				
3			=$A1*B$1				
4							

6.5.4. Какой вид приобретут формулы, хранящиеся в диапазоне ячеек C1:C3 при их копировании в диапазон ячеек G2:G4?

	A	B	C	D	E	F	G
1			=A1+B1				
2			=$A$1*$B$1				
3			=$A1*B$1				
4							

6.6. Технология обработки числовой информации. Визуализация данных с помощью диаграмм и графиков

6.6.1. С помощью круговой диаграммы визуализировать данные о стоимости комплектующих компьютера, хранящиеся на диске Windows-CD в папке ..\Test в файле Test.xls на листе *Круговая диаграмма*. Сохранить файл электронных таблиц с круговой диаграммой в файле Calc.xls.

Наименование устройства	Цена (в у.е.)
Системная плата	80
Процессор	70
Оперативная память	15
Жесткий диск	100
Дисковод 3,5"	14
Монитор	200
Дисковод CD-ROM	30
Корпус	25
Клавиатура	10
Мышь	5

6.6.2. С помощью линейчатой диаграммы визуализировать данные о численности населения в некоторых странах мира. Данные хранятся на диске Windows-CD в папке ..\Test в файле Test.xls на листе *Гистограмма*. Сохранить файл электронных таблиц с круговой диаграммой в файле Calc.xls.

Страна	Население
Китай	1273
Индия	1030
США	279
Индонезия	228
Бразилия	175
Россия	146
Бангладеш	131

6.6.3. Построить график функции $y = 5 \cdot x^2$ по ее числовому представлению, хранящийся на диске Windows-CD в папке ..\Test\ в файле Test.xls на листе *График*. Сохранить файл электронных таблиц с круговой диаграммой в файле Calc.xls.

x	–5	–4	–3	–2	–1	0	1	2	3	4	5
y=5*x^2	125	80	45	20	5	0	5	20	45	80	125

6.6.4. С помощью линейчатой диаграммы наглядно представить сравнительную длину некоторых единиц измерения длины. Данные хранятся на диске Windows-CD в папке ..\Test в файле Test.xls на листе *Линейчатая диаграмма*. Сохранить файл электронных таблиц с круговой диаграммой в файле Calc.xls.

Единица длины	Значение в миллиметрах
1 сантиметр	10
1 аршин	710
1 вершок	44,4
1 фут	304,8
1 дюйм	25,4

6.7. Базы данных

6.7.1. Записи в базе данных размещаются в:
1) ячейках; 2) строках; 3) столбцах; 4) таблицах.

6.7.2. Просмотр всех записей базы данных удобнее производить в:
1) отчете; 2) запросе; 3) форме; 4) таблице.

6.7.3. Просмотр отдельной записи базы данных удобнее производить в:
1) отчете; 2) таблице; 3) форме; 4) запросе.

6.7.4. Отбор записей базы данных, удовлетворяющих заданным условиям, удобнее производить в:
1) отчете; 2) таблице; 3) форме; 4) запросе.

6.8. Системы управления базами данных. Создание простой табличной базы данных

6.8.1. С помощью системы управления базами данных (Microsoft Access или OpenOffice.org Base) создать базу данных *«Записная книжка»*, включающую столбцы (Фамилия — текстовый тип, Год рождения — дата, Возраст — числовой тип) и состоящую не менее чем из пяти записей. Сохранить в папке ..\Result файл базы данных. Произвести сортировку записей по первому столбцу.

6.8.2. С помощью системы управления базами данных (Microsoft Access или OpenOffice.org Base) создать базу данных «Процессоры», включающую столбцы (Название — текстовый тип, Год выпуска — дата и Разряд-

ность — числовой тип) и состоящую не менее чем из пяти записей. Сохранить в папке ..\Result файл базы данных. Произвести сортировку записей по второму столбцу.

6.8.3. С помощью системы управления базами данных (Microsoft Access или OpenOffice.org Base) создать базу данных «Переменные», включающую столбцы (Название — текстовый тип, Тип переменной — текстовый тип, и Количество занимаемых ячеек в оперативной памяти — числовой тип) и состоящую не менее чем из пяти записей. Сохранить в папке ..\Result файл базы данных. Произвести сортировку записей по третьему столбцу.

6.8.4. С помощью системы управления базами данных (Microsoft Access или OpenOffice.org Base) создать базу данных «Поисковые системы в Интернете», включающую столбцы (Название — текстовый тип, Адрес в Интернете — текстовый тип, и Наличие системы каталогов — логический тип) и состоящую не менее чем из пяти записей. Сохранить в папке ..\Result файл базы данных. Произвести сортировку записей по первому столбцу.

Тема 7. Коммуникационные технологии

7.1. Способы подключения к Интернету

7.1.1. Для подключения к Интернету домашнего настольного компьютера целесообразно использовать:
1) спутниковый канал; 3) GPRS;
2) ADSL; 4) оптоволокно.

7.1.2. Для подключения к Интернету ноутбука в поездке целесообразно использовать:
1) спутниковый канал; 3) GPRS;
2) ADSL; 4) оптоволокно.

7.1.3. Для подключения к Интернету компьютерного класса целесообразно использовать:
1) спутниковый канал; 3) GPRS;
2) ADSL; 4) оптоволокно.

7.1.4. Для подключения к Интернету географически удаленного сервера целесообразно использовать:
1) спутниковый канал; 3) GPRS;
2) ADSL; 4) оптоволокно.

7.2. Адресация в Интернете

7.2.1. Как правильно записывается доменное имя сервера в Интернете?
1) ru.iit.metodist;
2) ru.metodist.iit;
3) iit.metodist.ru;
4) iit.ru.metodist.

7.2.2. Как правильно записывается IP-адрес компьютера в Интернете?
1) 83.237.199.60;
2) 8323719960;
3) 83.237.199;
4) 237.199.60.

7.2.3. При подключении к Интернету любой компьютер обязательно получает:
1) доменное имя;
2) IP-адрес;
3) доменное имя и IP-адрес;
4) IP-адрес и доменное имя.

7.2.4. База данных доменных имен хранится:
1) на центральном компьютере Интернета;
2) на каждом сервере Интернета;
3) на серверах Интернет-провайдеров;
4) иерархически распределена по серверам доменов.

7.3. Создание Web-страницы

7.3.1. С помощью Web-редактора (например, Компоновщика, входящего в SeaMonkey — интегрированное приложение для работы в Интернете) создать Web-страницу. На странице должны быть: заголовок, разделительная линия, текст, рисунок и гиперссылка. Сохранить Web-страницу в папке ..\Result, файле index1.htm.

7.3.2. С помощью Web-редактора (например, Компоновщика, входящего в SeaMonkey — интегрированное приложение для работы в Интернете) создать Web-страницу. На странице должны быть: заголовок, разделительная линия, таблица и гиперссылка. Сохранить Web-страницу в папке ..\Result, файле index2.htm.

7.3.3. С помощью Web-редактора (например, Компоновщика, входящего в SeaMonkey — интегрированное приложение для работы в Интернете) создать Web-страницу. На странице должны быть: заголовок, разделительная линия, нумерованный список и гиперссылка. Сохранить Web-страницу в папке ..\Result, файле index3.htm.

7.3.4. С помощью Web-редактора (например, Компоновщика, входящего в SeaMonkey — интегрированное приложение для работы в Интернете) создать Web-страницу. На странице должны быть: заголовок, разделительная линия, маркированный список и гиперссылка. Сохранить Web-страницу в папке ..\Result, файле index4.htm.

Ответы на тесты

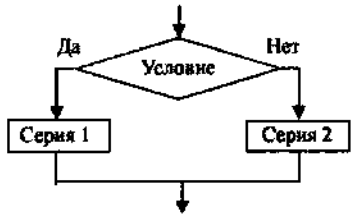
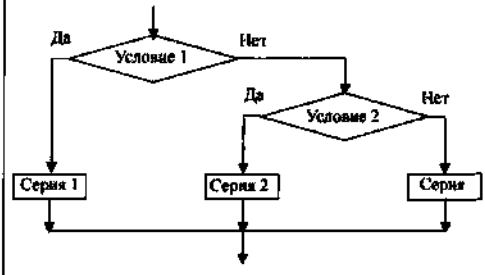
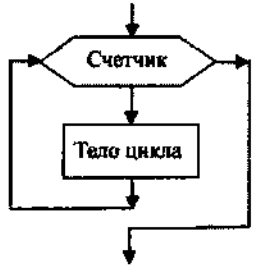
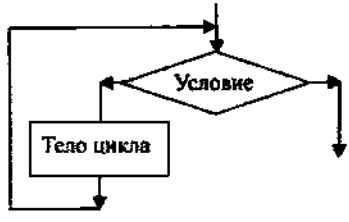
Тема 1. Информация. Кодирование информации

№ задания	Тип задания	Уровень сложности	Ответ
1.1.1	ВО	Б	4
1.1.2	ВО	Б	1
1.1.3	ВО	Б	2
1.1.4	ВО	Б	3
1.2.1	ВО	Б	2
1.2.2	ВО	Б	2
1.2.3	ВО	Б	3
1.2.4	ВО	Б	3
1.3.1	ВО	Б	3
1.3.2	ВО	Б	3
1.3.3	ВО	Б	2
1.3.4	ВО	Б	1
1.4.1	ВО	Б	1
1.4.2	ВО	Б	4
1.4.3	ВО	Б	3
1.4.4	ВО	Б	2
1.5.1	*ВО*	*Б*	*1*
1.5.2	ВО	Б	2
1.5.3	ВО	Б	2
1.5.4	ВО	Б	1
1.6.1	ВО	Б	4
1.6.2	ВО	Б	4
1.6.3	ВО	Б	2
1.6.4	ВО	Б	3
1.7.1	ВО	Б	2
1.7.2	ВО	Б	2
1.7.3	ВО	Б	3
1.7.4	ВО	Б	3
1.8.1	КО	П	101000
1.8.2	КО	П	40
1.8.3	КО	П	50
1.8.4	КО	П	28

Тема 2. Устройство компьютера и программное обеспечение

№ задания	Тип задания	Уровень сложности	Ответ
2.1.1	ВО	Б	3
2.1.2	ВО	Б	4
2.1.3	ВО	Б	4
2.1.4	ВО	Б	4
2.2.1	ВО	Б	4
2.2.2	ВО	Б	2
2.2.3	ВО	Б	4
2.2.4	ВО	Б	1
2.3.1	ВО	Б	3
2.3.2	ВО	Б	3
2.3.3	ВО	Б	1
2.3.4	ВО	Б	3
2.4.1	КО	П	8 Гбайт/с
2.4.2	КО	П	4 Гбайт/с
2.4.3	КО	П	2 Гбайт/с
2.4.4	КО	П	528 Мбайт/с
2.5.1	ВО	Б	2
2.5.2	ВО	Б	1
2.5.3	*ВО*	*Б*	*4*
2.5.4	ВО	Б	1
2.6.1	КО	Б	A:\Pic\Draw\B.bmp
2.6.2	КО	Б	A:\Doc\A.doc
2.6.3	КО	Б	A:\Doc\Ref\C.doc
2.6.4	КО	Б	A:\Pic\D.bmp
2.7.1	ВО	Б	2
2.7.2	ВО	Б	1
2.7.3	ВО	Б	3
2.7.4	ВО	Б	4

Тема 3. Алгоритмизация и программирование

№ задания	Тип задания	Уровень сложности	Ответ
3.1.1	РО	Б	
3.1.2	РО	Б	
3.1.3	РО	Б	
3.1.4	РО	Б	

№ задания	Тип задания	Уровень сложности
3.2.1	РО	П

Ответ:

Visual Basic .NET	Visual C#	Visual J#	Delphi

`If Условие Then`
`  Серия 1`
`[Else`
`  Серия 2]`
`End If`
	`if (Условие)`
	`{`
	`  Серия 1;`
	`}`
	`[else`
	`{`
	`  Серия 2;`
	`}]`
		`if (Условие)`
		`{`
		`  Серия 1;`
		`}`
		`[else`
		`{`
		`  Серия 2;`
		`}]`
			`If Условие Then`
			`  begin`
			`  Серия 1`
			`  end`
			`[Else`
			`  begin`
			`  Серия 2`
			`  end];`

№ задания	Тип задания	Уровень сложности
3.2.2	РО	П

Ответ:

Visual Basic .NET

```
Select Case Выражение
 Case Условие1
 Серия 1
 Case Условие2
 Серия 2
 [Case Else
 Серия]
End Select
```

Visual C#

```
switch (Выражение)
 [case список1_констант:
 Серия 1;
 break;
 case список2_констант:
 Серия 2;
 break;
 [default:
 Серия;
 break;
]
```

Visual J#

```
switch (Выражение)
 {case список1_констант:
 Серия 1;
 break;
 case список2_констант:
 Серия 2;
 break;
 [default:
 Серия;
 break;
}
```

Delphi

```
Case Выражение Of
 список1_констант:
 begin
     Серия 1
 end;
 список2_констант:
 begin
     Серия 2
 end;
 [Else
 begin
     Серия
 end;]
end;
```

№ задания	Тип задания	Уровень сложности
3.2.3	РО	П

Ответ:

Visual Basic .NET	Visual C#	Visual J#	Delphi
For Счетчик= НачЗнач To КонЗнач [Step] Тело цикла Next [Счетчик]	for (int Счетчик=1; Счетчик<НачЗнач; Счетчик++) { Тело цикла; }	for (int Счетчик=1; Счетчик<НачЗнач; Счетчик++) { Тело цикла; }	For Счетчик:= НачЗнач to КонЗнач Do begin Тело цикла end;

№ задания	Тип задания	Уровень сложности
3.2.4	РО	П

Ответ:

Visual Basic .NET	Visual C#	Visual J#	Delphi

Do While
Условие
Тело цикла
Loop
 	
Do Until
Условие
Тело цикла
Loop
	While
	(Условие)
	{
	Тело
	цикла;
	}
		While
		(Условие)
		{
		Тело
		цикла;
		}
			while
			Условие **do**
			begin
			Тело
			цикла;
			end;

№ задания	Тип задания	Уровень сложности	Ответ
3.3.1	ВО	Б	1
3.3.2	ВО	Б	3
3.3.3	ВО	Б	3
3.3.4	ВО	Б	4
3.4.1	КО	В	1010
3.4.2	КО	В	24
3.4.3	КО	В	0.01
3.4.4	КО	В	0.2
3.5.1	ВО	Б	2
3.5.2	ВО	Б	1
3.5.3	ВО	Б	3
3.5.4	ВО	Б	1
3.6.1	КО	П	5
3.6.2	КО	П	2
3.6.3	КО	П	120
3.6.4	КО	П	False

№ задания	Тип задания	Уровень сложности	Ответ
3.7.1	ПЗ	В	..\Result\VB2003\arrayMax ..\Result\Visual C#\arrayMax ..\Result\Visual J#\arrayMax ..\Result\Delphi\arrayMax

Visual Basic .NET

```
Dim A(9), I, Max As Byte
Private Sub Button1_Click(…)
Max=0
A(Max)=A(0)
For I=1 To 9
If A(I)>A(Max) Then A(Max)=A(I) : Max=I
Next I
Label1.Text=Max
Label2.Text=A(Max)
End Sub
```

Visual C#

```
Random rnd=new Random();
int[] A=new int[10];
int I, Max;
private void button1_Click(…)
{
Max=0;
A[Max]=A[0];
for (I=1; I<A.Length; I++)
{if (A[I]>A[Max])
  {  A[Max]=A[I];
Max=I;
  }
}
label1.Text=Convert.ToString(Max);
label2.Text=Convert.ToString(A[Max]);
}
```

Visual J#

```
Random rnd=new Random();
int[] A=new int[10];
int I, Max;
private void button1_Click (…)
{
Max=0;
A[Max]=A[0];
for (I=1; I<10; I++)
{
if (A[I]>A[Max])
{ A[Max]=A[I];
  Max=I;
  }
}
label1.set_Text(System.Convert.ToString(Max));
label2.set_Text(System.Convert.ToString(A[Max]));
}
```

Delphi

```
var
A:array[1..10] of integer;
I, Max, N: integer;
procedure Tform1.Button1Click(…);
begin
Max:=1;
A[Max]:=A[1];
For I:=2 To 10 Do
begin
If A[I]>A[Max] Then
begin
A[Max]:=A[I];
Max:=I;
end;
end;
Label1.Caption:=IntToStr(Max);
Label2.Caption:=IntToStr(A[Max]);
end;
```

№ задания	Тип задания	Уровень сложности	Ответ
3.7.3	ПЗ	В	..\Result\VB2003\factorial1 ..\Result\Visual C#\factorial1 ..\Result\Visual J#\factorial1 ..\Result\Delphi\factorial1

Visual Basic .NET

```
Dim N, I As Byte, F As Long
Private Sub TextBox1_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
TextBox1.TextChanged
ListBox1.Items.Clear()
N=Val(TextBox1.Text)
F=1
For I=1 To N
F=F*I
ListBox1.Items.Add(Str(I) & "! =" & Str(F))
Next I
End Sub
```

Visual C#

```
int N, I;
ulong F;
private void textBox1_TextChanged_1(object sender,
System.EventArgs e)
{
listBox1.Items.Clear();
N=Convert.ToInt16(textBox1.Text);
F=1;
for (I=1; I<=N; I++)
{
F=F*Convert.ToUInt64(I);
listBox1.Items.Add(Convert.ToString(I) + "! =" +
Convert.ToString(F));
}
}
```

Visual J#

```
int N, I;
long F;

private void textBox1_TextChanged (Object sender,
System.EventArgs e)
{
listBox1.get_Items().Clear();
N=System.Convert.ToInt16(textBox1.get_Text());
F=1;
for (I=1; I<=N; I++)
{
F=F*System.Convert.ToInt64(I);
listBox1.get_Items().Add(System.Convert.ToString(I) + "! =" +
System.Convert.ToString(F));
}
}
```

Delphi

```
var
N, I: byte;
F: int64;
procedure Tform1.Edit1Change(Sender: Tobject);
begin
N:=StrToInt(Edit1.Text);
F:=1;
ListBox1.Items.Clear;
For I:=1 To N Do
begin
F:=F*I;
ListBox1.Items.Add(IntToStr(I) + '! =' + IntToStr(F));
end;
end;
```

№ задания	Тип задания	Уровень сложности	Ответ
3.7.4	ПЗ	В	..\Result\VB2003\factorial2 ..\Result\Visual C#\factorial2 ..\Result\Visual J#\factorial2 ..\Result\Delphi\factorial2

Visual Basic .NET

```
'Рекурсивная функция
Function F(ByVal N As Byte) As Long
If N=1 Then
F=1.0
Else
F=N*F(N-1)
End If
End Function

Dim N, I As Byte
Private Sub TextBox1_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
TextBox1.TextChanged
ListBox1.Items.Clear()
N=Val(TextBox1.Text)
For I=1 To N
ListBox1.Items.Add(Str(I) & "! =" & Str(F(I)))
Next I
End Sub
```

Visual C#

```
int N, I;

public long F(int N)
{
long result;
if (N!=1)
{
result=F(N-1)*Convert.ToInt64(N);
return result;
}
else return 1;
}
private void textBox1_TextChanged(object sender,
System.EventArgs e)
{
listBox1.Items.Clear();
N=Convert.ToInt16(textBox1.Text);
for (I=1;I<=N;I++)
{
listBox1.Items.Add(Convert.ToString(I) + "! =" +
Convert.ToString(F(I)));
}
}
```

Visual J#

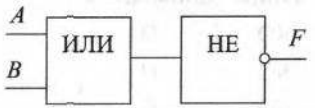
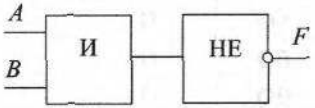
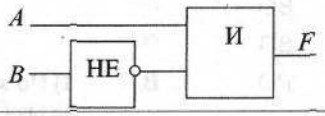
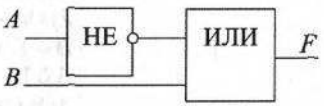
```
int N, I;
public long F(int N)
{
long result;
if (N!=1)
{
result=F(N-1)*System.Convert.ToInt64(N);
return result;
}
else return 1;
}

private void textBox1_TextChanged (Object sender,
System.EventArgs e)
{
listBox1.get_Items().Clear();
N=System.Convert.ToInt16(textBox1.get_Text());
for (I=1; I<=N; I++)
{
listBox1.get_Items().Add(System.Convert.ToString(I) + "! =" +
System.Convert.ToString(F(I)));
}
}
```

Delphi

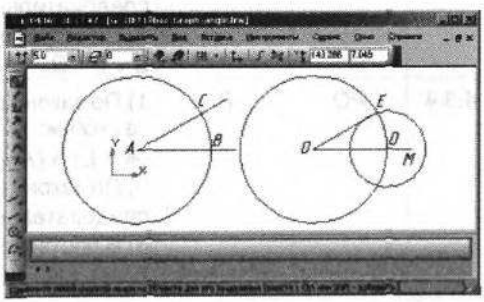
```
function Factorial(N:byte): int64;
begin
if N=1
then Factorial:=1
else Factorial:=N*Factorial(N-1);
end;

var
I: byte;
N: byte;

procedure TForm1.Edit1Change(Sender: TObject);
begin
N:=StrToInt(Edit1.Text);
ListBox1.Items.Clear;
For I:=1 To N Do
begin
ListBox1.Items.Add(IntToStr(I) + '! =' +
IntToStr(Factorial(I)));
end;
end;
```

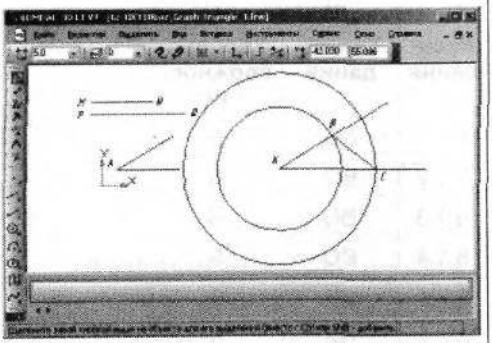
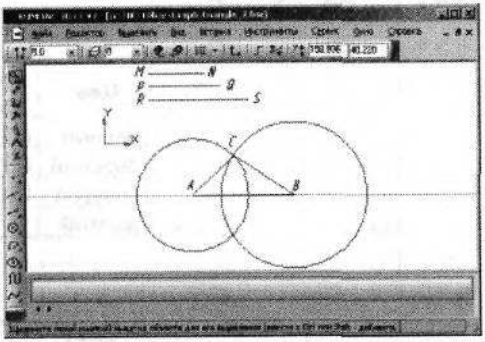
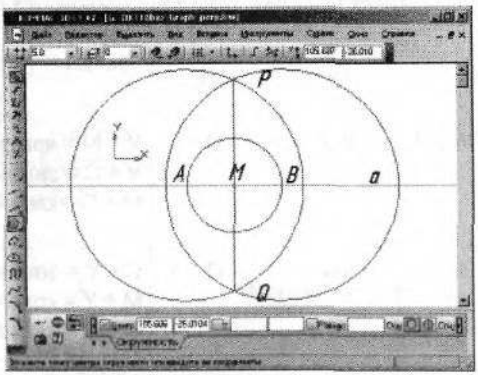
Тема 4. Основы логики и логические основы компьютера

№ задания	Тип задания	Уровень сложности	Ответ
4.1.1	КО	П	истинно
4.1.2	КО	П	ложно
4.1.3	КО	П	истинно
4.1.4	КО	П	истинно
4.2.1	ВО	П	4
4.2.2	ВО	П	3
4.2.3	ВО	П	4
4.2.4	ВО	П	1
4.3.1	РО	В	1) По закону дистрибутивности вынести A за скобки: $(A \& B) \vee (A \& \overline{B}) = A \& (B \vee \overline{B})$. 2) По закону исключенного третьего: $B \vee \overline{B} = 1$, следовательно, $A \& (B \vee \overline{B}) = A \& 1$. 3) В соответствии со свойствами констант: $A \& 1 = A$.
4.3.2	РО	В	1) По закону дистрибутивности вынести B за скобки: $(\overline{A} \& B) \vee (A \& B) = B \& (A \vee \overline{A})$. 2) По закону исключенного третьего: $\overline{A} \vee A = 1$, следовательно, $B \& (\overline{A} \vee A) = B \& 1$. 3) В соответствии со свойствами констант: $B \& 1 = B$.
4.3.3	РО	В	1) По закону дистрибутивности вынести B за скобки: $(A \vee B) \& (\overline{A} \vee B) = B \vee (A \& \overline{A})$. 2) По закону непротиворечия: $A \& \overline{A} = 0$, следовательно, $B \vee (A \& \overline{A}) = B \vee 0$. 3) В соответствии со свойствами констант: $B \vee 0 = B$.
4.3.4	РО	В	1) По закону дистрибутивности вынести В за скобки: $(A \vee B) \& (A \vee \overline{B}) = A \vee (B \& \overline{B})$. 2) По закону непротиворечия: $B \& \overline{B} = 0$, следовательно, $A \vee (B \& \overline{B}) = A \vee 0$. 3) В соответствии со свойствами констант: $A \vee 0 = A$.

4.4.1	ПЗ	П	.\Result\log.xls
4.4.2	ПЗ	П	.\Result\log.xls
4.4.3	ПЗ	П	.\Result\log.xls
4.4.4	ПЗ	П	.\Result\log.xls
4.5.1	РО	В	
4.5.2	РО	В	
4.5.3	РО	В	
4.5.4	РО	В	

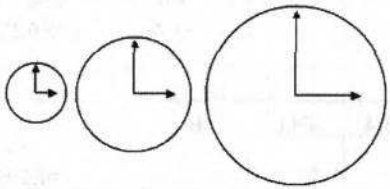
Тема 5. Моделирование и формализация

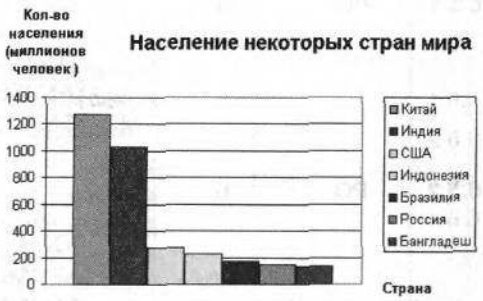
№ задания	Тип задания	Уровень сложности	Ответ
5.1.1	ВО	Б	4
5.1.2	ВО	Б	1
5.1.3	ВО	Б	2
5.1.4	ВО	Б	3
5.2.1	ПЗ	П	..\Result\Model\angle.frw 

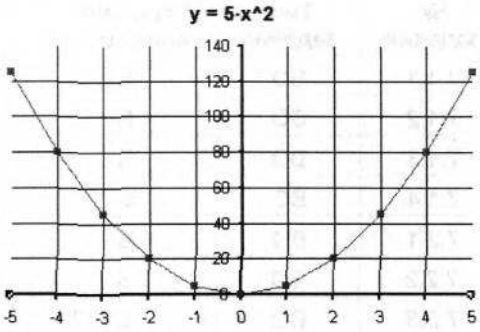
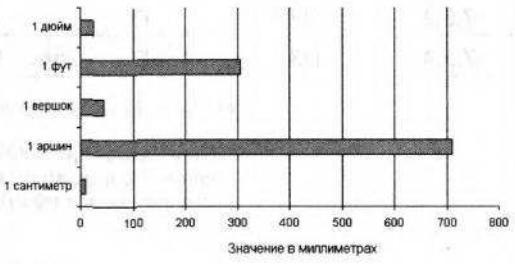
5.2.2	ПЗ	П	..\Result\Model\triangle1.frw 
5.2.3	ПЗ	П	..\Result\Model\triangle1.frw 
5.2.4	ПЗ	П	..\Result\Model\perp.frw 

Тема 6. Информационные технологии

<table>
<tr><th>№ задания</th><th>Тип задания</th><th>Уровень сложности</th><th>Ответ</th></tr>
<tr><td>6.1.1</td><td>ВО</td><td>Б</td><td>1</td></tr>
<tr><td>6.1.2</td><td>ВО</td><td>Б</td><td>4</td></tr>
<tr><td>6.1.3</td><td>ВО</td><td>Б</td><td>4</td></tr>
<tr><td>6.1.4</td><td>ВО</td><td>Б</td><td>2</td></tr>
<tr><td>6.2.1</td><td>ПЗ</td><td>Б</td><td>..\Result\IT\Text1.doc</td></tr>
<tr><td>6.2.2</td><td>ПЗ</td><td>Б</td><td>..\Result\IT\Text2.doc</td></tr>
<tr><td>6.2.3</td><td>ПЗ</td><td>Б</td><td>..\Result\IT\Text3.doc</td></tr>
<tr><td>6.2.4</td><td>ПЗ</td><td>Б</td><td>..\Result\IT\Text4.doc</td></tr>
<tr><td>6.3.1</td><td>КО</td><td>П</td><td>
<table>
<tr><th rowspan="2">Цвет</th><th colspan="3">Интенсивность базовых цветов</th></tr>
<tr><th>Красный</th><th>Зеленый</th><th>Синий</th></tr>
<tr><td>черный</td><td>00000000</td><td>00000000</td><td>00000000</td></tr>
<tr><td>зеленый</td><td>00000000</td><td>11111111</td><td>00000000</td></tr>
<tr><td>голубой</td><td>00000000</td><td>11111111</td><td>11111111</td></tr>
<tr><td>желтый</td><td>11111111</td><td>11111111</td><td>00000000</td></tr>
</table>
</td></tr>
<tr><td>6.3.2</td><td>КО</td><td>П</td><td>
<table>
<tr><th rowspan="2">Цвет</th><th colspan="3">Интенсивность базовых цветов</th></tr>
<tr><th>Красный</th><th>Зеленый</th><th>Синий</th></tr>
<tr><td>красный</td><td>11111111</td><td>00000000</td><td>00000000</td></tr>
<tr><td>синий</td><td>00000000</td><td>00000000</td><td>11111111</td></tr>
<tr><td>пурпурный</td><td>11111111</td><td>00000000</td><td>11111111</td></tr>
<tr><td>белый</td><td>11111111</td><td>11111111</td><td>11111111</td></tr>
</table>
</td></tr>
<tr><td>6.3.3</td><td>КО</td><td>П</td><td>Y + M = красный
Y + C = зеленый
M + C = синий</td></tr>
<tr><td>6.3.4</td><td>КО</td><td>П</td><td>C + Y = зеленый
M + Y = красный
C + M = синий</td></tr>
</table>

№ задания	Тип задания	Уровень сложности	Ответ
6.4.1	ПЗ	Б	
6.4.2	ПЗ	Б	Файл в формате JPEG
6.4.3	ПЗ	Б	
6.4.4	ПЗ	Б	
6.5.1	РО	Б	..\Result\Calc.xls D =B2+C2 =$A$1*$B$1 =$A2*C$1
6.5.2	РО	Б	E =C2+D2 =$A$1*$B$1 =$A2*D$1 ..\Result\Calc.xls

6.5.3	РО	Б	..\Result\Calc.xls F =D2+E2 =$A$1*$B$1 =$A2*E$1
6.5.4	РО	Б	..\Result\Calc.xls G =E2+F2 =$A$1*$B$1 =$A2*F$1
6.6.1	ПЗ	Б	..\Result\Calc.xls 
6.6.2	ПЗ	Б	..\Result\Calc.xls 

6.6.3	ПЗ	Б	..\Result\Calc.xls 
6.6.4	ПЗ	Б	..\Result\Calc.xls 
6.7.1	ВО	Б	2
6.7.2	ВО	Б	4
6.7.3	ВО	Б	3
6.7.4	ВО	Б	4
6.8.1	ПЗ	П	База данных «Записная книжка»
6.8.2	ПЗ	П	База данных «Процессоры»
6.8.3	ПЗ	П	База данных «Переменные»
6.8.4	ПЗ	П	База данных «Поисковые системы в Интернете»

Тема 7. Коммуникационные технологии

№ задания	Тип задания	Уровень сложности	Ответ
7.1.1	ВО	Б	2
7.1.2	ВО	Б	3
7.1.3	ВО	Б	4
7.1.4	ВО	Б	1
7.2.1	ВО	Б	3
7.2.2	ВО	Б	1
7.2.3	ВО	Б	2
7.2.4	ВО	Б	4
7.3.1	ПЗ	П	..\Result\index1.htm
7.3.2	ПЗ	П	..\Result\index2.htm
7.3.3	ПЗ	П	..\Result\index3.htm
7.3.4	ПЗ	П	..\Result\index4.htm

Учебное издание

Угринович Николай Дмитриевич

ИНФОРМАТИКА и ИКТ
Учебник для 11 класса
Профильный уровень

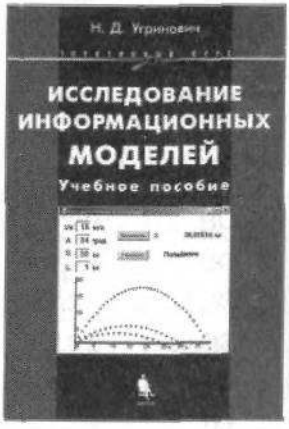
Ведущий редактор *О. Полежаева*
Художник *С. Инфантэ*
Художественный редактор *О. Лапко*
Компьютерная верстка: *В. Носенко*

Подписано в печать 17.03.09. Формат 60×90 1/16
Бумага офсетная. Усл. печ. л. 19,5. Тираж 5000 экз. Заказ 5241.

Издательство «БИНОМ. Лаборатория знаний»
Адрес для переписки: 125167, Москва, проезд Аэропорта, 3
Телефон: (499) 157-5272. E-mail: binom@Lbz.ru
http://www.Lbz.ru

Отпечатано ООО ПФ «Полиграфист»,
Вологда, Челюскинцев, 3,
тел. (8172) 72-61-75,
E-mail: forma@pfpoligrafist.com

ИМЕЕТСЯ В ПРОДАЖЕ



Угринович Н. Д. ***Исследование информационных моделей. Элективный курс*** *: учебное пособие / Н. Д. Угринович. — 2-е изд., испр. — 2006. — 200 с. : ил.*

Данное учебное пособие является частью УМК наряду с компьютерным практикумом на CD-ROM. Курс учит создавать и исследовать информационные модели из различных предметных областей с использованием систем объектно-ориентированного программирования и электронных таблиц.

На CD-ROM представлены: практикум по построению и исследованию моделей в форме проектов на языках Visual Basic и Delphi, а также компьютерных моделей в электронных таблицах; дистрибутивы систем объектно-ориентированного программирования Visual Basic и Delphi, а также электронных таблиц StarOffice Calc и OpenOffice Calc.

Для учащихся старших классов информационно-технологического, физико-математического и естественно-научного профилей.

Угринович Н. Д. ***Исследование информационных моделей. Элективный курс*** *: компьютерный практикум на CD-ROM / Н. Д. Угринович. — 2-е изд., испр. — 2006. — 200 с. : ил.*

CD-ROM включает интерактивный практикум, содержащий указания по выполнению практических заданий и ответы на них, т. е. готовые проекты на языках Visual Basic и Delphi. На CD-ROM размещено программное обеспечение, необходимое для реализации компьютерного практикума, а именно свободно распространяемые версии систем программирования Visual Basic и Delphi, а также файлы электронных таблиц.

ИЗДАТЕЛЬСТВО
«БИНОМ
Лаборатория знаний»

125167, Москва, проезд Аэропорта, д. 3
Телефон: (499) 157-5272
e-mail: Lbz@aha.ru, http://www.Lbz.ru
Оптовые поставки:
(499) 174-7616, 171-1954, 170-6674

ИМЕЕТСЯ В ПРОДАЖЕ

Монахов М. Ю. ***Учимся проектировать на компьютере. Элективный курс*** *: практикум / М. Ю. Монахов, С. Л. Солодов, Г. Е. Монахова. — 2-е изд., испр. — 2006. — 172 с. : ил.*

Практикум обеспечивает преподавание курса компьютерного проектирования в старших классах.

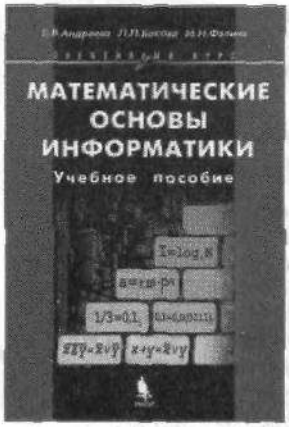
Практикум позволяет освоить основы современных компьютерных технологий проектирования и дизайна. Рассмотрены компьютерные системы проектирования AutoCAD и 3D Studio MAX. Главы практикума представляют собой законченные учебные модули, каждый из которых включает в себя краткую теорию по теме, типовые практические работы, вопросы для самоконтроля и проверочные задания.

Для учащихся старших классов физико-математического, естественно-научного, технологического профилей и универсального обучения.

ИЗДАТЕЛЬСТВО
«БИНОМ
Лаборатория знаний»

125167, Москва, проезд Аэропорта, д. 3
Телефон: (499) 157-5272
e-mail: Lbz@aha.ru, http://www.Lbz.ru
Оптовые поставки:
(499) 174-7616, 171-1954, 170-6674

ИМЕЕТСЯ В ПРОДАЖЕ



Андреева Е. В. ***Математические основы информатики. Элективный курс*** *: учебное пособие / Е. В. Андреева, Л. Л. Босова, И. Н. Фалина. — 2-е изд., испр. — 2007. — 328 с. : ил.*

Учебное пособие входит в УМК для старших классов наряду с методическим пособием и хрестоматией.

Материал раскрывает взаимосвязь математики и информатики, показывает, как развитие одной из этих научных областей стимулировало развитие другой. Дается углубленное представление о математическом аппарате, используемом в информатике, демонстрируется, как результаты, полученные в математике, послужили источником новых идей и результатов в теории алгоритмов, программировании и в других разделах информатики.

Для учащихся старших классов информационно-технологического, физико-математического и естественно-научного профилей, желающих расширить свои теоретические представления о математике в информатике и информатике в математике.

ИЗДАТЕЛЬСТВО
«БИНОМ
Лаборатория знаний»

125167, Москва, проезд Аэропорта, д. 3
Телефон: (499) 157-5272
e-mail: Lbz@aha.ru, http://www.Lbz.ru
Оптовые поставки:
(499) 174-7616, 171-1954, 170-6674

Николай Дмитриевич Угринович — ученый и методист, заведующий лабораторией информатики Московского института открытого образования, кандидат педагогических наук, автор учебного и программно-методического комплекса по курсу «Информатика и ИКТ», который включает учебники для 7–11 классов, элективный курс, методическое пособие для учителей, электронные материалы.

Данный учебник для 11 класса предназначен для обучения информатике и ИКТ на профильном уровне.

- Вы приобретете навыки построения компьютерных моделей в предметных областях физики, математики, биологии, экономики, химии, информатики на языках программирования Visual Basic, Turbo Delphi и в электронных таблицах
- Вы познакомитесь с основами макетирования и верстки в настольных издательских системах, получите навыки работы с системами распознавания символов
- Вы закрепите свои знания технологии работы с графической информацией на компьютере
- Вы получите основы знаний по базам данных и СУБД

Учебник рекомендуется для изучения в 11 классах информационно-технологического и физико-математического профилей

ISBN 978-5-9963-0047-1