№ 15 (208) апрель1999

еженедельное
ПРИЛОЖЕНИЕ К ГАЗЕТЕ
«ПЕРВОЕ СЕНТЯБРЯ»

# ИНФОРМАТИК

Электронные версии газеты “Первое сентября” и приложений http://www.1september.ru


# 50 лет методу Монте-Карло

**В 1999 году исполняется 50 лет методу Монте-Карло. В 1949 году в американском журнале “Journal of American Statistical Association” появилась статья под названием “Метод Монте-Карло”. Возникновение и развитие метода связано с именами известных ученых Дж. фон Неймана[1], С.Улама[2], Н.Метрополиса, а также Г.Кана и Э.Ферми.**

Общепринятого определения этого метода не существует. “Методом Монте-Карло” обычно называют численный метод решения задач при помощи моделирования случайных величин и последующей статистической оценки их характеристик (см., например: *Соболь И.М.* Численные методы Монте-Карло. М.: Наука, 1973). Более популярно методика его использования может быть сформулирована так: чтобы приближенно вычислить некоторую скалярную величину $a$, надо придумать такую случайную величину $\xi$, что, проведя ее многократную генерацию $\xi_1, \xi_2, \ldots, \xi_n$, можно получить величину, которая может быть принята в качестве искомой.

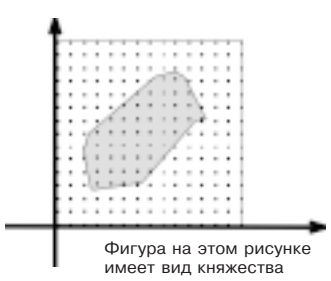
Одним из простейших примеров применения метода Монте-Карло является следующий.

*Требуется определить площадь некоторой ограниченной фигуры.*

Построим квадрат (в общем случае — прямоугольник), содержащий эту фигуру. Выберем $n$ случайных точек, равномерно распределенных в квадрате, и обозначим через $n'$ количество точек, попавших внутрь фигуры. Ясно, что, когда $n$ велико, значение $n'$ пропорционально площади фигуры:

$$\frac{n'}{n} = \frac{s_\phi}{s_к}$$

Здесь $S_\phi$ и $S_к$ — площадь фигуры и площадь квадрата соответственно.


Фигура на этом рисунке имеет вид княжества Монако.

Название метода произошло от города Монте-Карло (княжество Монако), известного своими казино, ибо одним из простейших приборов для генерирования случайных чисел является рулетка.

Необходимо подчеркнуть, что теоретические основы метода Монте-Карло были известны значительно раньше 1949 года. Однако до появления электронно-вычислительных машин (ЭВМ) метод не мог стать универсальным, так как моделирование случайных величин вручную — весьма трудоемкий процесс, и для получения достаточно точной оценки искомой величины требуются выполнение весьма большого количества вычислений и последующая статистическая обработка колоссального объема числовых данных. Именно бурное развитие ЭВМ способствовало широкому распространению метода Монте-Карло. Перечислим лишь некоторые из областей его применения: вычисление определенных интегралов большой кратности, решение систем линейных и нелинейных уравнений, вычисление экстремумов (в том числе глобальных у многоэкстремальных функций многих переменных).


Монте-Карло. Княжество Монако.

[1] Джон (Янош) фон Нейман — американский математик, внесший также большой вклад в создание первых ЭВМ (см., например, “Информатику”, № 35/98).

[2] С именем американского математика Станислава Улама связана статья в “Информатике” (см. № 43/97).

**НАШИ ДЕТИ БУДУТ ЖИТЬ В XXI ВЕКЕ**

## Системы счисления и компьютерная арифметика

*Е.В. АНДРЕЕВА, И.Н. ФАЛИНА*

*Продолжение. Начало в № 14/99.*

В предыдущем номере опубликованы разделы:

Тема 1. Системы счисления как разновидность информационных систем.

Тема 2. История систем счисления.

Тема 3. Позиционные системы счисления.

Теперь представлены следующие темы:

Тема 4. Двоичное кодирование информации.

Тема 5. Двоичная арифметика.

Тема 6. Перевод чисел из двоичной системы счисления в десятичную и обратно.

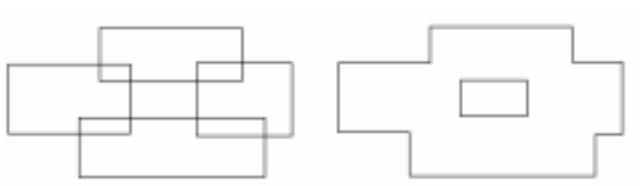
*Продолжение следует*

Задача 11-й Кировской областной олимпиады по информатике

# Прямоугольники

*С.М. ОКУЛОВ*

На плоскости задано $N$ ($1 \le N \le 300$) прямоугольников со сторонами, параллельными координатным осям (см. рисунок). Координаты вершин прямоугольников — вещественные числа, заданные с точностью до двух знаков после запятой. Написать программу определения площади фигуры, получающейся в результате объединения прямоугольников.



**Входные данные**. В первой строке файла INPUT.TXT записано число $N$. В следующих $N$ строках записано по четыре числа — координаты левого нижнего и правого верхнего углов прямоугольника.

**Выходные данные**. В файле OUTPUT.TXT записывается одно число — площадь фигуры с точностью до четырех знаков.

**Пример**

**Входные данные**

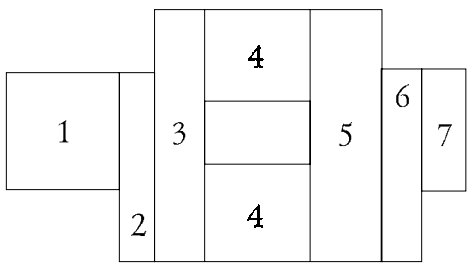
```
4
0 0 10.1 5.2
-3 3 5.36 7
1 6 9 15
8 3 20 8
```

**Выходные данные**

```
195.188
```

**Идея решения**

Продлим все вертикальные стороны прямоугольников до пересечения с осью $X$. Этим мы определим на оси $X$ множество интервалов, внутри каждого из которых длина сечения по оси $Y$ будет постоянной (см. рисунок).




Поэтому для нахождения площади достаточно отсортировать точки на оси $X$, рассмотреть все сечения и добавить к площади <длину интервала> · <длину сечения>. Для поиска длины сечения используем следующий метод. Началу каждого отрезка присвоим значение признака, равное 1, а его концу $-1$ и отсортируем точки по значению координаты $Y$. Подсчитаем сумму значений признаков. Если она не равна нулю, то соответствующий интервал "плюсуем" к длине сечения.

**Структуры данных**

```
Const InputFile='INPUT.TXT';
OutputFile='OUTPUT.TXT';
MaxN=300;
Eps=1e-5;
Tochn=4;
Type  TPoint=Record X,Y:Real;End;
MasPnt=Array[1..MaxN] Of TPoint;
BMasPnt=Array[1..MaxN*2] Of Real;
BMasSSt=Array[1..MaxN*2] Of LongInt;
Var   PrM: Array[1..2] Of MasPnt;
    N: LongInt;
    Res: Real;
    Ox, Oy: BMasPnt;
    FOy: BMasSSt;
```

**Процедура инициализации и ввода данных**

Координаты вершин прямоугольников храним в массиве PrM, причем в PrM[1] (это массив) — координаты первой вершины, а в PrM[2] — второй. Вершина прямоугольника с меньшими значениями (X, Y) записывается в PrM[1]. Кроме того, значения X запоминаются в массиве Ox.

```
Procedure ReadPoint(Var A: TPoint);
 Begin
   Read(A.X, A.Y);
 End;
Procedure More(Const a, b: Real): Boolean;
  Begin
    More:=a-b>Eps;
  End;
Procedure Swap(Var a, b: Real);
  Var t: Real;
   Begin
     t:=a; a:=b; b:=t;
   End;
Procedure Init;
Var i: LongInt;
 Begin
  Res:=0;
  FillChar(Ox, SizeOf(Ox), 0);
  FillChar(Oy, SizeOf(Oy), 0);
  FillChar(Foy, SizeOf(FoY), 0);
  FillChar(PrM, SizeOf(PrM), 0);
  Assign(Input, InputFile);Reset(Input);
  Read(N);
  For i:=1 To N Do Begin
   ReadPoint(PrM[1,i]);ReadPoint(PrM[2,i]);
   If More(PrM[1,i].X, PrM[2,i].X)
   Then Swap(PrM[1,i].X, PrM[2,i].X);
   If More(PrM[1,i].Y, PrM[2,i].Y)
   Then Swap(PrM[1,i].Y, PrM[2,i].Y);
   Ox[i*2-1]:=PrM[1,i].X;Ox[i*2]:=PrM[2,i].X;
  End;
  Close(Input);
 End;
```

Процедура вывода результата и основная программа очевидны.

```
Procedure Done;
  Begin
    Assign(Output, OutputFile);
    Rewrite(Output);
    WriteLn(Res:0:Tochn);
    Close(Output);
  End;
Begin
  Init;
  Solve;
  Done;
End.
```

**Вспомогательные процедуры**

```
Function Eq(Const a, b: Real): Boolean;
 Begin
   Eq:=Abs(a-b)<Eps;
 End;

Procedure SwapInt(Var a, b: LongInt);
Var t: LongInt;
 Begin
   t:=a; a:=b; b:=t;
 End;
```

Для решения задачи потребуется сортировка элементов массива. Используем метод Хоара.

```
Sort(Var Ox:BMasPnt; Const lf,rg:LongInt);
Var i,j:LongInt;
    x:Real;
Begin
  i:=lf;j:=rg;x:=Ox[(lf+rg) Div 2];
  Repeat
    While More(x, Ox[i]) Do Inc(i);
    While More(Ox[j], x) Do Dec(j);
    If i<=j Then
    Begin
      Swap(Ox[i], Ox[j]);Inc(i); Dec(j);
    End;
  Until i>j;
  If lf<j Then Sort(Ox, lf, j);
  If i<rg Then Sort(Ox, i, rg);
 End;
```

Та же самая сортировка, но с одновременной перестановкой элементов другого массива.

```
Procedure FstSort(Var Ox: BMasPnt;
                  Var SOx: BMasSSt;
                  Const lf, rg: LongInt);
Var i, j: LongInt;
    x: Real;
Begin
  i:=lf;j:=rg;x:=Ox[(lf+rg) Div 2];
  Repeat
    While More(x, Ox[i]) Do Inc(i);
    While More(Ox[j], x) Do Dec(j);
    If i<=j Then
    Begin
      Swap(Ox[i],Ox[j]);
      SwapInt(SOx[i],SOx[j]);
      Inc(i);Dec(j);
    End;
  Until i>j;
  If lf<j Then FstSort(Ox, Sox, lf, j);
  If i<rg Then FstSort(Ox, Sox, i, rg);
 End;
```

Перейдем к основной процедуре — вычислению площади объединения прямоугольников.

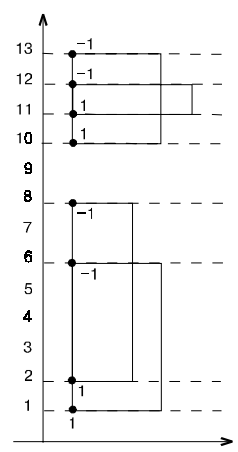
```
Procedure Solve;
 Var i: LongInt; m: Real;
Begin
  Sort(Ox, 1, N*2);{сортируем по неубыванию
      значения координаты X прямоугольников}
  m:=0;Res:=0;{m - длина сечения, Res -
    значение площади объединения прямоугольников}
  For i:=1 To N*2 Do Begin
   If i<>1 Then Res:=Res+Abs((Ox[i]-Ox[i-1])*m);
     {прибавляем площадь очередного сечения}
    If (i=1) Or Not(Eq(Ox[i], Ox[i-1]))
    Then Change(Ox[i], m);
   {определяем новое значение длины сечения}
  End;
End;
```

Продолжим уточнение.

```
Function Peres(Const k: LongInt;
               Const x: Real): Boolean;
Begin
   Peres:=Not More(PrM[1,k].X,x) And
          More(PrM[2,k].X,x);
End;

Procedure Change(Const x: Real; Var rs: Real);
Var i, M: LongInt;
Begin
  M:=0;FillChar(Oy,SizeOf(Oy),0);
  FillChar(FOy,SizeOf(Foy),0);
   For i:=1 To N Do
    If Peres(i, x) Then Begin
      {если есть пересечение?}
     Oy[M+1]:=PrM[1,i].Y;Oy[M+2]:=PrM[2,i].Y;
       {формируем массив ординат для данной
        координаты X}
     FOy[M+1]:=1;FOy[M+2]:=-1;
      {признаки начала и конца отрезка}
     Inc(M, 2);
   End;
   If M=0 Then rs:=0
   Else Begin FstSort(Oy,FOy,1,M);
              rs:=Abs(Calc(Oy,FOy,M));
         End;
{сортируем Oy, переставляя одновременно
соответствующие элементы массива FOy; вычисляем
новое значение длины сечения - функция Calc}
End;
```

И, наконец, последнее уточнение: функция вычисления длины сечения. Для примера на рисунке длина сечения будет равна 10.

```
Function Calc(Const Ox: BMasPnt;
              Const SOx: BMasSSt;
              Const N: LongInt): Real;
Var  sc: Real;
i, bb: LongInt;
Begin
  sc:=0;bb:=0;
  If SOx[1]>0 Then Inc(bb);
  For i:=2 To N Do Begin
   If bb>0 Then sc:=sc + Ox[i] - Ox[i-1];
   Inc(bb, SOx[i]);{изменяем значение признака
пересечения интервалов,
например, для значения 8 на
рисунке оно будет равно нулю и
длина от 8 до 10 не войдет в
результат}
  End;
  Calc:=sc;
End;
```




Разработка тестирующих систем для проведения олимпиады была выполнена студентами МГУ Матюхиным Виктором и Беровым Виталием. Они, а также студент СПбИТМО Пестов Олег являются авторами ряда задач.

# Первое знакомство с объектно-ориентированным программированием

*А.А. СЕМЕНОВ, А.Г. ЮДИНА*

*Продолжение. Начало см. в № 14/99*

## Занятие № 3

Определим нашу ближайшую цель.

Создадим некий родительский объектный тип (на Си он назывался бы классом).

Это будет прародитель целого семейства разнообразных по сложности графических образов, обладающих некоторыми общими характеристиками, а именно: `x`, `y` — координаты левого верхнего угла прямоугольника, описанного около объекта; `color` — его цвет.

Кроме этого, наши объекты будут обладать следующими свойствами:

каждый из них может быть

- создан вызовом специальной процедуры-конструктора `Init`;
- нарисован цветом `color` вызовом процедуры `Show`;
- спрятан вызовом процедуры `Hide` (т.е. нарисован, например, цветом фона);
- перемещен вызовом процедуры `Move` (сперва спрятан, затем, после увеличения координат объекта на величину перемещения по каждой из осей, нарисован вновь).

По синтаксису объекты похожи на записи в Паскале или структуры в Си (т.е. представляют собой объединение под одним именем данных различных типов). В набор процедур добавим также процедуру `Draw`, имеющую служебное назначение. В нашем абстрактном объекте она ровным счетом ничего не делает, так как мы не заключали никаких договоренностей о рисунке (форме объекта). Этот метод будет переопределяться в потомках нашего абстрактного объекта.

Для того чтобы конкретный тип потомка можно было бы задавать на этапе выполнения программы, процедура `Draw` объявлена виртуальной. Это позволяет менять вид объектов в нашем скрин-сейвере "на лету".

```
type
GOb=object
  x,y: integer;
  color: byte;
  constructor Init(ax,ay:integer;acolor: byte);
  procedure Draw(acolor: byte); virtual;
  procedure Show;
  procedure Hide;
  procedure Move(dx,dy: integer);
end;
```

Характеристики (`x,y,color`) называются ПОЛЯМИ объекта, а процедуры/функции (`Init`, `Draw`, `Show`, `Hide`), определяющие его свойства, — МЕТОДАМИ объекта.

В объектно-ориентированном программировании считается дурным тоном менять напрямую поля объекта (хотя Турбо Паскаль это допускает). Лучше использовать для этого специально разработанные методы.

Итак, опишем все методы.

```
uses graph13h;
type
GOb=object
  x,y: integer;
  color: byte;
  constructor Init(ax,ay: integer;acolor: byte);
  procedure Draw(acolor: byte); virtual;
  procedure Show;
  procedure Hide;
  procedure Move(dx,dy: integer);
end;
constructor GOb.Init(ax,ay: integer;acolor: byte);
begin
  x:=ax;  y:=ay;
  color:=acolor;
end;
procedure GOb.Draw(acolor: byte);
begin
end;
procedure GOb.Show;
begin
  Draw(color);
end;
procedure GOb.Hide;
begin
  Draw(0);
end;
procedure GOb.Move(dx,dy: integer);
begin
  Hide;
  x:=x+dx;
  y:=y+dy;
  Show;
end;
var
  a: GOb;
begin
 Screen13h;
 a.init(160,100,14);
 a.show;
 readln;
end.
```

Запускаем программу, и... видим на экране августовскую ночь. Ничего удивительного — ведь наш объектный тип-прародитель имеет пустой метод `Draw`. Кстати, в Паскале даже есть процедура `Abstract`, генерирующая ошибку при обращении к экземплярам абстрактных объектов. Следовательно, надо породить от нашего `GOb`'а жизнеспособных потомков.

Простейшим потомком абстрактного объекта `GOb` будет, по всей видимости, просто точка.

Тип `Point` объекта-потомка (в Си и Java-класс) `GOb` опишем следующим образом после описания объекта-родителя:

```
type
  {... здесь будет описание родительского типа GOb}
  Point=object(GOb);
    procedure Draw; virtual;
  end;
```

Нас вполне устраивает поведение предка (его методы) и его данные (поля).

Нужно всего лишь перекрыть (т.е. заменить одноименным) единственный метод `Draw`. В предке метод `Draw` ничего не делает. Новый `Draw` виртуальный — так же, как и в объекте-предке. Напомним, что объявление метода виртуальным дает возможность потомкам различных типов обращаться к методам общего предка на этапе выполнения программы. В нашем случае различные методы `Draw` потомков будут вызываться методами `Show` и `Hide`, как предписано в объекте `GOb`.

Все остальные методы и поля наш объект `Point` наследует от предка.

Содержание метода `Draw` элементарно:

```
procedure Point.Draw;
begin
  PutPixel(x,y,color)
end;
```

Теперь можно наконец получить работающую пробную программу. Для большей красоты мы создадим сотню-другую объектов-точек.

```
{$R-,Q-}
uses graph13h;
type
GOb=object
  x,y: integer;
  color: byte;
  constructor Init(ax,ay: integer;acolor: byte);
  procedure Draw(acolor: byte); virtual;
  procedure Show;
  procedure Hide;
  procedure Move(dx,dy:integer);
end;
Point=object(GOb)
  procedure Draw(acolor: byte); virtual;
end;
constructor GOb.Init(ax,ay:integer;acolor: byte);
begin
  x:=ax;  y:=ay;
  color:=acolor;
end;
procedure GOb.Draw(acolor: byte);
begin
end;
procedure GOb.Show;
begin
  Draw(color);
end;
procedure GOb.Hide;
begin
  Draw(0);
end;
procedure GOb.Move(dx,dy: integer);
begin
  Hide;
  x:=x+dx;  y:=y+dy;
  Show;
end;
procedure Point.Draw(acolor: byte);
begin
  PutPixel(x,y,acolor)
end;
const
  max=100;
var
  a: array [1..max] of Point;
  i: integer;
begin
 Screen13h;
 for i:=1 to max do
  a[i].Init(Random(320),Random(200),Random(256));
   for i:=1 to max do a[i].Show;
   readln;
end.
```

Запускаем программу, и... видим на экране августовскую ночь, но уже звездную.

Раздел описаний (намного больший, чем раздел операторов) включает:

- описание типов (одного предка и одного потомка);
- описание процедур (методов наших объектов);
- описание констант;
- описание переменных.

**Примечание**

**Вопрос**: *Как использовать процедуру* `Abstract`*?*

**Ответ**: *Нужно подключить модуль* `Objects` *библиотеки Turbo Vision, входящей в комплект поставки Турбо Паскаля, начиная с версии 6 (файл OBJECTS.TPU). После этого достаточно вызвать процедуру* `Abstract` *из абстрактного метода. При прямом вызове абстрактного метода процедура генерирует ошибку времени выполнения номер 211, напоминая, что такое действие некорректно.*





Окончание на с. 4

# Первое знакомство с объектно-ориентированным программированием

**Окончание. Начало на с. 3**

*Задание на дом*

**1.** Сделать так, чтобы звезды двигались с одной скоростью паралелльно диагонали экрана или случайным образом (как броуновские частицы).

**2.** Дополнить листинг объектным типом `Cross`, экземпляры которого на экране выглядят как крестики 3×3 точки. Если новый тип создан, то простой заменой описания

```
var
  a: array [1..max] of Point;
```

на

```
var
  a: array [1..max] of Cross;
```

точки превращаются... точки превращаются... превращаются точки... в элегантные крестики! А поведение крестиков останется тем же, что и у точек. Это и неудивительно, ведь у объектов `Point` и `Cross` общий предок `GOb`.

## Занятие № 4

Наверное, читатель уже оценил удобство объектного подхода, выполняя домашнее задание. Добавляя аналогичным образом все новых потомков, можно легко разнообразить "ассортимент" живущих на экране объектов. Попробуйте поселить одновременно на экране и точки, и крестики. И пусть, например, точки изображают падающие снежинки, а крестики (или другие придуманные вами объекты) движутся случайным образом.

Но сначала избавимся от необходимости переносить из программы в программу описание Адама нашей иерархии объектов — создадим еще один модуль с описанием родительского объекта `GOb`:

```
{$R-,Q-}
unit MyObj;

Interface
type
  GOb=object
  x,y:word;
  color:byte;
   constructor Init(ax,ay:integer;acolor:byte);
   procedure Draw(acolor:byte);virtual;
   procedure Show;
   procedure Hide;
   procedure Move(dx,dy:integer);
end;
Implementation
constructor GOb.Init(ax,ay:integer;acolor:byte);
begin
 x:=ax;y:=ay;color:=acolor
end;
procedure GOb.Draw(acolor:byte);
begin
end;
procedure GOb.Show;
begin
  Draw(color)
end;
procedure GOb.Hide;
begin
  Draw(0)
end;
procedure GOb.Move(dx,dy:integer);
begin
  Hide;
  x:=x+dx;  y:=y+dy;
  Show
end;
end.
```

Файл надо сохранить как MYOBJ.PAS и затем откомпилировать. Получим еще один модуль — MYOBJ.TPU.

Теперь программы будут выглядеть не так "тяжеловесно".

```
{$R-,Q-}
uses graph13h,MyObj,crt;
type
Point=object(GOb)
procedure Draw(acolor:byte);virtual;
end;
Cross=object(GOb)
procedure Draw(acolor:byte);virtual;
end;
procedure Point.Draw(acolor:byte);
begin
  PutPixel(x,y,acolor)
end;
procedure Cross.Draw(acolor:byte);
begin
  HLine(x,y+2,x+4,acolor);VLine(x+2,y,y+4,acolor)
end;
const max=100;
var a:array[1..max] of Point;b:array[1..max] of Cross;
        i:integer;
    begin
     Screen13h;
for i:=1 to max do
  begin
       a[i].Init(random(320),random(200),14);
       {это 100 желтых точек}
       b[i].Init(random(320),random(200),random(256));
      {a это 100 разноцветных крестиков}
  end;
for i:=1 to max do begin  a[i].Show;b[i].Show end;
readln;
while not keypressed do
for i:=1 to max do
  begin
     a[i].Move(-2+random(5),-2+random(5));
    {точки изображают броуновское движение}
     b[i].Move(1,1); delay(1)
    {a крестики дружно плывут на юго-восток}
  end;
readln
end.
```

А если мы захотим существенно изменить поведение наших объектов? Пусть нам необходимо, чтобы объекты летали внутри некоторой области и отражались от ее границ. Тогда нашим объектам понадобятся дополнительные характеристики — скорости по осям `vx` и `vy`. При достижении вертикальной границы скорость `vx` объекта должна менять знак, а при "столкновении" с горизонтальной границей меняет знак вертикальная составляющая `vy`.

В ООП не принято "тревожить" первичный объект в иерархии без крайней необходимости. Нужно порождать потомков, имеющих все необходимые поля и методы, определяющие их поведение. Нам требуется, чтобы точка и крестик отражались от стенок.

Давайте попробуем создать потомка объекта `Gob` по имени `Ball`, который "знает", как обеспечивать зеркальное отражение, а от него уже породим `Point` и `Cross`.

Итак, изменим схему наследования.

```
{$R-,Q-}
uses graph13h,MyObj,crt;
type
Ball=object(GOb)
vx,vy:integer;
constructor Init(ax,ay,avx,avy:integer;acolor:byte);
procedure Move;
end;
Cross=object(Ball)
procedure Draw(acolor:byte);virtual;
end;
constructor Ball.Init(ax,ay,avx,avy:integer;acolor:byte);
begin
  inherited Init(ax,ay,acolor);
  {унаследованная процедура инициализации}
  vx:=avx;
  vy:=avy;
 {плюс еще два поля (две характеристики объекта): скорости по осям}
end;
procedure Ball.Move;
begin
  inherited Move(vx,vy);
  if (x<20) or (x>299) then vx:=-vx;
  if (y<20) or (y>179) then vy:=-vy;
  {добавление к унаследованной процедуре движения}
  {отражения от вертикальных и горизонтальных "стенок"}
end;
procedure Cross.Draw(acolor:byte);
begin
   HLine(x,y+2,x+4,acolor);VLine(x+2,y,y+4,acolor)
end;
const max=100;
var a:array[1..max] of Cross;
    i:integer;
begin
  Randomize;
  Screen13h;
  for i:=1 to max do
  begin
   a[i].Init(20+random(280),20+random(160),random(5),random(3), random(255));
     a[i].Show;
     {каждый крестик при "рождении" в случайном месте наделяется
     случайными скоростями по осям}
  end;
  readln;
  while not keypressed do
  for i:=1 to max do
    begin
       a[i].Move;delay(1)
    end;
  readln
end.
```

Уф! Ну, достаточно на этот раз.

На следующем занятии мы создадим более сложный тип-потомок под освежающим именем `Sprite`.

**Примечание**

**Вопрос**: ***Почему меню и стандартные диалоги (открыть/сохранить файл, получить подтверждение на серьезное действие) в Windows-программах все "на одно лицо"?***

**Ответ**: *Дело в том, что все они — экземпляры или прямые потомки стандартных объектов Windows. Так проще жить и программистам, и пользователям. Достаточно вспомнить муки бухгалтеров начала 90-х, привыкших вызывать меню в электронной таблице Supercalc клавишей* [/]*, затем вынужденных переучиться на* [F10] *в QuattroPro, а затем — на* [Alt] *в Excel, если вдруг вышла из строя мышка. Дела давно минувших дней...*

***Домашнее задание в стиле "ретро".***

Попробуйте заставить звездочки и крестики бежать по радиусам из центра экрана в духе скрин-сейвера "Летящие звезды".

# Постижение теории через решение задач

А.Ю. УВАРОВ

## Рецензия на учебное пособие для общеобразовательной школы "Практикум по информатике в среде LogoWriter"

Рецензируемое пособие содержит задачи и упражнения по основным темам курса информатики: основы алгоритмизации и программирования, компьютерное моделирование. Пособие состоит из двух частей. Первая часть предназначена для учащихся 8—9-х классов. Вторая часть включает материал, который, по мнению автора, может использоваться в старших классах физико-математических школ.

Первая отличительная особенность пособия — его направленность на старшие классы школы. Трудно не согласиться с автором, что до последнего времени язык ЛОГО по большей части применялся при работе с учащимися младших классов, где потенциал ЛОГО используется в минимальной мере. ЛОГО — мощное инструментальное средство, в основе которого лежит основной язык функционального программирования — Лисп. Естественно, что на ЛОГО можно решать самые разные задачи. Достоинство пособия в том, что его автор, А.Г. Юдина, попыталась воспользоваться этим обстоятельством и подготовить материал для курса информатики, который ориентирован на использование ЛОГО. Если учесть, что ЛОГО — распространенный инструмент изучения информатики в начальной школе, то такая ориентировка выглядит весьма привлекательной.

Другая отличительная особенность пособия — движение к теории от решения конкретных задач, а не наоборот. По мнению рецензента, это в полной мере соответствует духу того подхода к обучению, который развил и проповедует создатель ЛОГО Сеймур Пейперт, на которого неоднократно ссылается автор. Последовательно выдерживая конструктивистский стиль предъявления материала, автор выстраивает фактически две системы задач. Их последовательное решение способствует формированию умения разбивать задачу на подзадачи, выстраивать общий план решения как структуру взаимосвязанных модулей, выявлять аналогичные части и создавать соответствующие им функциональные элементы, которые могут использоваться для решения широкого класса аналогичных задач.

Третья особенность пособия — двойной характер решаемых учащимися задач, их ориентация на конкретные предметные области, изучаемые в общеобразовательной школе. Межпредметные связи объединяют пособие с физикой (задачи по изучению движения) и математикой (построение и исследование графиков). Важно и то, что пособие предлагает рассматривать соответствующие темы в курсе информатики в связи с их изучением в курсе физики или математики. Далеко не каждый материал по курсу информатики обладает этим бесспорным достоинством.

Материал пособия написан хорошим языком, а все содержащиеся в нем задания выверены на компьютере в среде LogoWriter. В заключительной части приведены решения всех рассматриваемых задач, что будет с большим удовлетворением встречено учителями. В целом пособие представляет собой законченный добротный материал.

Можно сделать несколько замечаний, которые могли бы способствовать повышению привлекательности рецензируемого пособия.

К пособию приложен материал "Краткое описание концепции пособия". Причина появления этого материала очевидна: включенное в текст предисловие формально и явно недостаточно для учителя, впервые открывающего пособие. Поместить в предисловии описание целей, которые преследовали авторы, "оправдание" принятой структуры пособия, указание о возможных способах использования пособия учителем, методистом, учениками — значит проявить уважение и доверие к читателю, облегчить ему знакомство с новой книгой. Рецензент рекомендует включить материал пояснительной записки в текст пособия. Имеющееся предисловие неоправданно сжато, и текст пояснительной записки его только украсит.

Второе замечание относится к "Приложению для учителя". Интересный текст "Среда LogoWriter — средство формирования и развития познавательной самостоятельности" написан излишне сжато. Для большинства потенциальных читателей пособия этой информации будет недостаточно. Совершенно справедливо указывая на потенциальные возможности предложенной системы задач, автор слишком мало говорит о методике (способе) их использования. Автор молчаливо полагает, что все читатели свободно владеют "сократовским диалогом", а это не всегда верно. Детальная методическая проработка нескольких занятий, несомненно, украсила бы материал. Включение "Приложения для учителя" в структуру пособия сравнительно просто решило бы и эту задачу. Раздел, где рассматриваются решения задач, — самое удобное место, чтобы разместить соответствующие описания.

# Практикум по информатике в среде LogoWriter

А.Г. ЮДИНА

## Избранные главы

**Полтора года назад (см. "Информатика", № 44/97) Национальный фонд подготовки кадров объявил о проведении конкурса на создание школьных учебников. Настоящее учебное пособие было признано победителем этого конкурса по направлению: "Задачник по информатике (практическое руководство) по основным разделам программы с теоретическим введением".**

### От автора

При подготовке к изданию замечания рецензента будут учтены. Методические рекомендации будут расширены и представлены отдельной книжкой.

Разрабатывая это учебное пособие, автор ставил перед собой следующие задачи:

- преодолеть некоторую оторванность, обособленность школьной информатики от основных школьных дисциплин;
- в большей степени подчинить ее преподавание общим педагогическим, образовательным целям.

Отличительными особенностями курса информатики, основанного на данном учебном пособии, являются:

- частичная интеграция со школьными курсами геометрии, алгебры и физики;
- направленность на общее развитие мышления и исследовательских навыков, использование специфических возможностей предмета для организации проблемного и поискового обучения.

Пособие содержит около 200 задач и заданий, предполагающих в основном самостоятельную практическую работу учащихся. Методика изложения материала — не от теории к задачам, а, наоборот, от практических результатов к обобщениям; положенный в основу педагогический принцип — предоставить школьникам возможность познавать и учиться в самостоятельной деятельности.

Первая часть (базовая, для 8—9-х классов) доступна среднему ученику обычной школы (возможно, троечнику) и школьному учителю физики или математики, не программисту, взявшемуся за преподавание информатики.

Более точным для пособия (но слишком громоздким) было бы название "Основы алгоритмизации и программирования. Компьютерное моделирование и исследование при изучении ряда тем геометрии, алгебры и физики".

Во второй части значительное внимание уделено работе со структурами данных и развитию навыков абстрагирования и обобщения. От задания к заданию постепенно усложняются как структуры программ, так и структуры данных, причем последовательность задач составлена так, чтобы можно было использовать предыдущие задачи в качестве готовых модулей.

Учебное содержание большей части задач двойное — с одной стороны, приемы алгоритмизации и программирования, а с другой — геометрия, либо физика, либо построение и исследование графиков функций.

Особое внимание при подборе задач уделялось тому, чтобы результаты были привлекательны для подростков (графические эффекты, имитация движения, диалог).

По мнению автора, программирование — область, в которой благодаря самой природе предмета достаточно легко организовать поисковое, проблемное обучение, предельно индивидуализированное, предоставить учащемуся возможность осваивать знания через достижение осознаваемых личностных целей. Таким образом, занятия программированием могут дать результаты не частные, а весьма ценные в общеобразовательном смысле. Проблемный, развивающий характер таких компьютерных занятий позволяет пробудить в школьниках желание экспериментировать, формулировать и проверять гипотезы, учиться на своих ошибках.

К сожалению, отечественная школьная информатика, в которой тема "Алгоритмизация и программирование" занимала с самого начала центральное место, все больше поворачивается к утилитарно-прагматическому использованию компьютеров, к обучению работе с конторскими программными средствами. А блестящие развивающие возможности программирования используют в основном математические школы. Этому есть ряд объективных причин. Перед началом содержательной работы в профессиональных языках программирования ученикам (и учителю) нужно преодолевать первоначальный барьер сложности (в основном синтаксис). В результате распространенным стало представление о том, что овладение программированием — удел избранных.

Использование в качестве основы специальных наборов исполнителей избавляет от ненужных сложностей и подробностей, но вместе с тем и ограничивает возможности. Эти наборы не предоставляют достаточно богатую среду, микромир (с этим связано и сужение цели курса — только развитие алгоритмического мышления).

Однако существует особый язык программирования, специально разработанный для развития мышления, творческих и исследовательских потенций. Это язык ЛОГО и целое семейство ЛОГО-сред. Несколько упрощая, можно сказать, что язык ЛОГО значительно ближе к человеческому образу мышления, чем к устройству компьютеров. Простой и естественный синтаксис плюс наглядность объектов и процессов позволяют работать с ЛОГО всем, даже слабым ученикам. В то же время, используя его, можно целенаправленно формировать культуру мышления, способности к анализу и синтезу, абстрагированию и обобщению. А ЛОГО-среды — это специально организованные для активного, поискового учения микромиры, предоставляющие широкие возможности для разнообразного моделирования.

При выборе того или иного направления в своей работе учитель информатики учитывает целый ряд объективных обстоятельств. Если для освоения компьютерных технологий действительно нужны современные, дорогие машины, то работать с ЛОГО можно практически на любой технике.

Освоение начал алгоритмизации и программирования в 8-м классе позволяет перейти к подготовке собственных программ как инструментов для построения и исследования графиков функций и моделирования различных видов движения. В учебном плане для 9-го класса этот материал проходит параллельно с соответствующими темами по алгебре и физике.

Продолжение на с. 6

# УЧЕБНИКИ

Кроме того, всем, кто желает освоить работу на современных вычислительных машинах, полезно знать вид первых 16 двоичных чисел наизусть:

| | | | |
|---|---|---|---|
| **0** | 0 | **8** | 1000 |
| **1** | 1 | **9** | 1001 |
| **2** | 10 | **10** | 1010 |
| **3** | 11 | **11** | 1011 |
| **4** | 100 | **12** | 1100 |
| **5** | 101 | **13** | 1101 |
| **6** | 110 | **14** | 1110 |
| **7** | 111 | **15** | 1111 |

### 6.2. Перевод целых чисел из десятичной системы счисления в двоичную

Пусть $a$ — десятичное целое число. Известно, что это число можно единственным образом представить в двоичной системе в виде суммы положительных степеней двойки:

$$a=a_n2^n+a_{n-1}2^{n-1}+ \ldots +a_12+a_0,$$

где $a_i$ — цифры двоичной системы.

Разделим число $a$ на 2 нацело. Частное от деления будет равно $b_n2^{n-1}+ b_{n-1}2^{n-2}+ \ldots + b_1$, а вот остаток от деления будет равен $a_0$ — последней цифре двоичной записи числа $a$.

Полученное частное опять разделим на 2, остаток от деления будет равен $a_1$ — предпоследней цифре двоичной записи $a$.

Если продолжить этот процесс деления, то на $n+1$-м шаге получим последовательность цифр $a_0$, $a_1$, $a_2$, ..., $a_n$, которые являются цифрами числа $a$ в двоичной системе, но записанными в обратном порядке.

Полученное правило перевода можно записать в виде следующего алгоритма.

**Алгоритм перевода целого числа из десятичной системы счисления в двоичную:**

1) *делим исходное число a на 2 нацело в десятичной системе и записываем в качестве нового значения десятичного числа a целую часть результата от деления;*
2) *повторяем деление до тех пор, пока число a не станет равным 0, выписывая остатки от деления (это будут 0 или 1) справа налево и получая при этом двоичную запись исходного числа a.*

**Пример 28**

Переведем число 123 в двоичную систему счисления:

123:2 = 61(**1**). Здесь и далее в скобках указано значение остатка от деления, в данном случае 1, следовательно, последняя цифра в двоичном представлении числа 123 равна 1.

61:2 =30(**1**)
30:2 =15(**0**)
15:2 =7(**1**)
7:2 =3(**1**)
3:2 =1(**1**)
1:2 =0(**1**)

Окончательный результат: $123=111101_2$.

Типичные ошибки при реализации такого алгоритма следующие:

- нарушение порядка записи получаемых цифр (часто их ошибочно записывают слева направо начиная с самой нижней, а не справа налево);
- неправильное выписывание крайней слева цифры (про нее либо забывают, т.к. в результате последнего деления получается ноль целых, либо получают неверно при выполнении операции деления с остатком меньшего числа на большее).

***Вопросы и задания***

**1.** Переведите в десятичную систему счисления числа, записанные в двоичной системе счисления (двоичные числа): 1; 101; 10000; 1000101010; 11001011.

**2.** Какое максимальное число можно записать в двоичной системе счисления пятью цифрами?

**3.** Перевести из двоичной системы в десятичную числа $0{,}00011_2$, $0{,}001_2$.

**4.** Фокусник отгадывает задуманное число по спичкам. Загадавший должен в уме делить задуманное число пополам, полученную половину опять пополам и т.д. (для нечетных чисел берется целая часть от деления) и при каждом делении класть перед собой спичку, направленную вдоль, если делится число четное, и поперек, если нечетное. По полученной фигуре фокусник всегда безошибочно отгадывает число. Как он это делает?

**5.** Можно ли любое целое число представить в виде суммы степеней двойки?

**6.** У меня 100 братьев. Младшему 1000 лет, а старшему 1111 лет. Старший учится в 1001-м классе. Может ли такое быть?

**7.** Составьте задачу, аналогичную задаче 6.

**8.** Переведите данные десятичные числа в двоичную систему: 10, 20, 100, 200, 1000.

**9.** Переведите следующие десятичные дроби в двоичную систему счисления: 0,25; 0,5; 0,625.

**10.** Сформулируйте критерий четности чисел в двоичной системе счисления.

Информатика — это наука, которая занимается вопросами представления и обработки информации.
(Манфред Брой, профессор, лауреат премии Лейбница в области информатики)

# Системы счисления и компьютерная арифметика

Е.В. АНДРЕЕВА, И.Н. ФАЛИНА

*Продолжение. Начало см. в № 14/99*

## Тема 4. Двоичное кодирование информации

С точки зрения математики все позиционные системы счисления являются равноправными, но на практике это далеко не всегда так. Из всех позиционных систем счисления наибольшее распространение, за исключением десятичной, получила двоичная система счисления. И в первую очередь это связано с надежностью представления информации, т.е. при кодировании, передаче и декодировании вероятность ошибки или потери информации мала по сравнению с использованием других систем счисления.

### 4.1. Примеры из истории

Двоичная система проста, так как для представления информации в ней используется всего два состояния, или две цифры. Такое представление информации принято называть ***двоичным кодированием***.

Представление информации в двоичной системе использовалось человеком с давних времен. Так, жители островов Полинезии обменивались информацией при помощи барабанов, чередуя звонкие и глухие удары. Звук над поверхностью воды распространялся на достаточно большое расстояние, и таким образом работал полинезийский телеграф.

Привычный нам телеграф в XIX—XX веках тоже передавал сообщения с помощью азбуки Морзе — в виде последовательности из точек и тире. Да и при простом стуке в дверь "условным сигналом" мы используем комбинацию коротких и длинных ударов.

Двоичная система используется для решения головоломок и построения выигрышных стратегий в некоторых играх. Одним из сложных, но наиболее впечатляющих примеров является стратегический анализ в игре "Ним".

**Правила игры "Ним".** Имеется три кучки камней. Двое играющих поочередно берут камни из этих кучек, причем при каждом ходе играющий может взять любое отличное от нуля число камней из любой (но только из одной) кучки. Выигрывает тот, кто возьмет последний камень.

"Ним" — старинная китайская игра. Научиться играть в нее несложно, значительно труднее научиться выигрывать. "Ним" относится к антагонистическим играм, т.е. к играм, в которых ничьей быть не может. Описание беспроигрышной стратегии можно найти в книге С.В. Фомина "Системы счисления" (*С.В. Фомин.* Системы счисления. М.: Наука, 1987).

### 4.2. Булева алгебра

Двоичные числа, состоящие из цифр 0 и 1, — рабочий инструмент раздела математики "Булева алгебра", или "Алгебра логики". Основатель этого раздела математики — английский математик XIX века Джордж Буль. Булева алгебра оперирует с высказываниями, относительно которых можно сказать, истинны они или ложны. Джордж Буль предложил кодировать такие высказывания при помощи двухсимвольного алфавита. Например, "T" и "F" или 0 и 1: "T" (*true* — правда) или единица соответствует истинному высказыванию, а "F" (*false* — ложь) или ноль — ложному. Примером таких высказываний является: "Сейчас на улице идет дождь". Если дождь действительно идет, то это высказывание истинно, в противном случае оно ложно.

Другой пример высказывания: "В этом году в феврале месяце 29 дней". Если текущий год високосный, то высказывание истинно, если текущий год не високосный, то высказывание ложно. Пример высказывания, относительно которого нельзя сказать, истинно оно или ложно: "Который сейчас час?".

Мы можем соединить высказывания в более сложные предложения (функции) при помощи ***связок,*** или ***логических операций***. Относительно этих сложных предложений также можно сказать, истинны они или ложны.

Для логических операций в алгебре логики и в языках программирования используются следующие обозначения:

1) логическое НЕ (логическое отрицание): $\neg$, $\overline{A}$, ***not***;
2) логическое И (конъюнкция, или логическое умножение): &, $\wedge$, $\cdot$, ***and***;
3) логическое ИЛИ (дизъюнкция, или логическое сложение): $\vee$, $+$, ***or***;
4) исключающее ИЛИ (сумма по модулю 2): $\oplus$, ***xor***.

Значение логических операций задается через ***таблицы истинности***. В этих таблицах для всех возможных значений операндов (в данном случае высказываний) указываются результаты соответствующей логической операции. Значение высказывания $a$ будем обозначать 0, если высказывание ложно, и 1 при истинном значении $a$.

Операция логического отрицания является ***унарной***, т.е. она применима только к одному операнду. Приведем таблицу истинности для логического отрицания ($f=\bar{a}$):

| $a$ | $f=\bar{a}$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

Таблица достаточно очевидна: “не истина” — это “ложь”, а “не ложь” — это истина.

Операция “логическое И” применяется к двум операндам, то есть является ***бинарной*** ($f=a\&b$), и определяется следующей таблицей истинности:

| $a$ | $b$ | $f=a\&b$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Операция “логическое ИЛИ” применяется также к двум операндам ($f=a\vee b$), и ее таблица истинности имеет следующий вид:

| $a$ | $b$ | $f=a\vee b$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Логическая операция “исключающее ИЛИ” применяется к двум операндам ($f=a\oplus b$) и является остатком от деления на два суммы операндов, что соответствует следующей таблице истинности:

| $a$ | $b$ | $f=a\oplus b$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Кроме того, можно заметить, что результатом этой операции является истина (единица) тогда и только тогда, когда операнды различны.

В алгебре логики доказано, что любую логическую функцию можно выразить через комбинацию всего лишь трех логических операций: И, ИЛИ и НЕ.

Для операций “И” и “ИЛИ” справедливы следующие законы:

**Закон коммутативности:**

$$a\&b = b\&a;$$
$$a\vee b = b\vee a.$$

**Закон ассоциативности:**

$$(a\&b)\&c = a\&(b\&c);$$
$$(a\vee b)\vee c = a\vee(b\vee c).$$

**Закон дистрибутивности:**

$$a\&(b\vee c) = (a\&b)\vee(a\&c);$$
$$a\vee(b\&c) = (a\vee b)\&(a\vee c).$$

**Закон поглощения 0 или 1:**

$$a\vee 0 = a;$$
$$a\&1 = a.$$

**Закон противоречия:**

$$a\&\bar{a} = 0$$

**Закон исключенного третьего:**

$$a\vee\bar{a} = 1$$

**Законы де Моргана:**

$$\overline{(a\&b)} = \bar{a}\vee\bar{b};$$
$$\overline{(a\vee b)} = \bar{a}\,\&\,\bar{b}.$$

**Закон двойного отрицания:**

$$\bar{\bar{a}} = a.$$

**Законы поглощения:**

$$a\&(a\vee b) = a;$$
$$a\vee(a\&b) = a.$$

Используя двоичное кодирование для высказываний (1 — истина, 0 — ложь), достаточно легко решить следующую логическую задачу:

“Царь позвал троих мудрецов и показал им пять колпаков: 2 черных и 3 белых. Потом он посадил всех мудрецов друг за другом. Первый мудрец сел лицом к стене. Второй мудрец сел за ним и видел перед собой только первого. Третий видел перед собой и первого, и второго. После этого царь надел колпак на каждого мудреца, но так, чтобы ни один мудрец не мог видеть цвет своего колпака.

Царь сказал, что признает самым мудрым того мудреца, который первый скажет, какой колпак надет на него. Кто оказался самым мудрым?”

*Решение.* Обозначим белый цвет колпака через 0, а черный — 1. Тогда может существовать только 7 вариантов расположения колпаков:





**Пример 25.** Рассмотрим еще один пример деления в двоичной системе счисления. Предварительно заметим, что в общем случае неизвестно, получим мы конечную дробь или бесконечную периодическую.

```
 1010₂ | 11₂
- 11   |-------
 ---   11,0101... = 11,(01)₂
  100
  - 11
  ----
    100
    - 11
    ----
      100
      - 11
      ----
        ...
```

В этом примере период дроби состоит из двух цифр. Для его определения можно выполнять деление до тех пор, пока не будет заметно повторение группы цифр в результате. Точнее, должно обнаружиться, что на каком-то этапе вычислений результат последнего вычитания совпал с неким предыдущим (но встречавшимся ранее уже при подсчете именно дробной части).

Следовательно, все остальные цифры дробной части результата будут повторяться такими же группами. Повторяющаяся группа и образует период дроби, в данном случае двоичной.

***Вопросы и задания***

**1.** Выполните следующие действия сложения:
$11010101_2+1110_2$;
$1011101_2+11101101_2$;
$11101011_2+10011101_2$.

**2.** Выполните следующие действия вычитания:
$11010101_2-1110_2$;
$110101110_2-10111111_2$;
$11011011_2-110101110_2$.

**3.** Выполните следующие действия умножения:
$11010101_2\times 1110_2$;
$111110_2\times 1000010_2$;
$111100_2\times 100100_2$.

**4.** Выполните следующие действия деления:
$10010000_2 1110_2$;
$101111001101_2:110101_2$;
$100011111111_2:101111_2$.

**5.** Выполните деление, получив в дробной части результата не менее 3 цифр: $1001_2:101_2$.

**6.** Восстановите цифры двоичной системы счисления, на месте которых в приведенных ниже арифметических действиях стоит знак “*”:
$**0*0*1**1_2+10111*10**_2=100*1*00010_2$;
$***0**00_2-11*11*11_2=1101*1_2$.

**7.** Подсчитайте сумму двоичных чисел в диапазоне от $100_2$ до $111_2$, включая границы диапазона. Ответ запишите в двоичной системе счисления.

**8.** Было 11 яблок. После того как каждое из них разрезали пополам, стало 110 половинок. В какой системе счисления вели счет?

**9.** В нашем классе $100011_2$ учеников. $111100_2$% из них учатся на “хорошо” и “отлично”. Сколько учеников учатся на “хорошо” и “отлично”?

## Тема 6
## Перевод чисел из двоичной системы счисления в десятичную и обратно

Более подробно материал этой темы изложен в главе 3 второй части.

### 6.1. Перевод целых чисел из двоичной системы счисления в десятичную

Дано число в двоичной системе счисления: $a=a_n a_{n-1}\ldots a_1 a_0$. Требуется получить запись этого числа в десятичной системе счисления.

Для решения этой задачи используем формулу (3) и представим число в развернутой форме: $a=a_n 2^n+a_{n-1}2^{n-1}+\ldots+a_1 2+a_0$. Значение числа $a$ в десятичной системе счисления как раз и равно значению этого многочлена. Данный способ можно сформулировать в виде следующего алгоритма.

**Алгоритм перевода целых чисел из двоичной системы счисления в десятичную:**

1) *пронумеруем цифры в двоичной записи числа справа налево начиная с нуля (номера соответствуют степеням 2 в многочлене (3);*
2) *сложим те степени двоек, которые соответствуют номерам цифр “1” в двоичной записи числа (все арифметические действия проводятся в десятичной системе). Полученный результат является значением числа в десятичной системе.*

**Пример 26**

Переведем число $1101_2$ в десятичную систему счисления:

$$1101_2=1\cdot 2^3+1\cdot 2^2+0\cdot 2^1+1\cdot 2^0=2^3+2^2+2^0=$$
$$=8+4+1=13_{10}.$$

**Пример 27**

Переведем двоичное число $1001101_2$ в десятичное:

$$1001101_2=2^6+2^3+2^2+2^0=64+8+4+1=79.$$

В процессе перевода требуется знание значений степеней числа 2. Приводим значения первых 10 степеней:

| | |
|---|---|
| $2^1=2$ | $2^6=64$ |
| $2^2=4$ | $2^7=128$ |
| $2^3=8$ | $2^8=256$ |
| $2^4=16$ | $2^9=512$ |
| $2^5=32$ | $2^{10}=1024$ |

в двоичной системе равно $11_2$=3). То есть результат сложения двух положительных чисел либо имеет столько же цифр, сколько и максимальное из двух слагаемых, либо на одну цифру больше, но этой цифрой может быть только единица.

### 5.2. Вычитание

При выполнении операции вычитания всегда из большего по абсолютной величине числа вычитается меньшее и у результата ставится соответствующий знак.

**Пример 17**

$11_2-1011_2$. Так как уменьшаемое меньше, чем вычитаемое, то следует вычесть $11_2$ из $1011_2$, приписав затем к результату знак "–": $11_2-1011_2=-(1011_2-11_2)=-1000_2$.

**Пример 18**

$$\begin{array}{r} 1011_2 \\ -\ 111_2 \\ \hline 100_2 \end{array}$$

При вычислении цифры третьего разряда результата произошел "заем" единицы из четвертого разряда. Так как каждая единица более старшего разряда равна основанию системы, то есть двум, то в третий разряд при "заеме" пришли две единицы.

**Пример 19.** Вычитание действительных чисел.

$$\begin{array}{r} 101_2 \\ -\ 10,1_2 \\ \hline 10,1_2 \end{array}$$

### 5.3. Умножение

Операция умножения выполняется с использованием таблицы умножения по обычной схеме, применяемой в десятичной системе счисления, с последовательным умножением множимого на очередную цифру множителя.

**Пример 20**

$$\begin{array}{r} \times\ 1011_2 \\ \times\ 101_2 \\ \hline 1011 \\ +\ 1011 \\ \hline 110111_2 \end{array}$$

**Пример 21**

$$\begin{array}{r} 1011_2 \\ \times\ 111_2 \\ \hline 1011 \\ +\ 1011 \\ 1011 \\ \hline 1001101_2 \end{array}$$

Обратите внимание, что для вычисления цифры третьего справа разряда результата в этом примере мы выполняем следующие действия: 1+1+1 (перенос из второго разряда)=1+$10_2$ (основание)=$11_2\equiv$ ≡1 + перенос в старший разряд.

**Пример 22**

$$\begin{array}{r} 10100_2 \\ \times\ 1010_2 \\ \hline +\ 101 \\ 101 \\ \hline 11001000_2 \end{array}$$

В данном примере, как и при десятичном умножении столбиком, умножение на 0 не производится, а все оставшиеся справа нули, не участвующие в умножении, приписываются справа к результату умножения.

Из приведенных примеров видно, что в двоичной системе операция умножения сводится к сдвигам множимого и сложению промежуточных результатов.

### 5.4. Деление

Операция деления выполняется по правилам, подобным правилам выполнения деления в десятичной системе счисления.

При делении столбиком в качестве промежуточных вычислений приходится выполнять действия умножения и вычитания, но в двоичной системе промежуточные умножения сводятся к умножению делителя или на 0, или на 1, поэтому наиболее сложной остается лишь операция вычитания, которую надо научиться делать безошибочно.

**Пример 23**

$$\begin{array}{r|l} -\ 11110_2 & 110_2 \\ \cline{2-2} 110 & 101_2 \\ \cline{1-1} -\ 110 & \\ 110 & \\ \cline{1-1} 0 & \end{array}$$

Если в результате выполнения операции деления не получается конечная дробь, то выполнять операцию деления можно до выделения периода или до получения требуемого количества знаков после запятой.

**Пример 24**

Выполнить деление до получения 5 знаков в дробной части.

$$\begin{array}{r|l} 1000_2 & 11_2 \\ \cline{2-2} -\ 11 & 10,10101_2 \\ \cline{1-1} 100 & \\ -\ 11 & \\ \cline{1-1} 100 & \\ -\ 11 & \\ \cline{1-1} 100 & \\ -\ 11 & \\ \cline{1-1} 100 & \\ -\ 11 & \\ \cline{1-1} 1 & \end{array}$$



| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Мог ли умный царь использовать последний вариант (110)? Нет! В этом случае третий мудрец сразу догадается, что на нем колпак белого цвета, — ведь черных колпаков всего два.

Но по этой же причине царь не может расположить колпаки по схеме 100 и 101. В этом случае второй мудрец легко может догадаться, что на нем белый колпак, — ведь вариант 110 исключен!

Итак, для умного царя возможны только 4 варианта: 000, 001, 010 и 011.

В любом случае на 1-го мудреца он может надеть колпак только белого цвета. Если этот мудрец действительно мудр, то он без труда может повторить наши рассуждения и решить задачу царя.

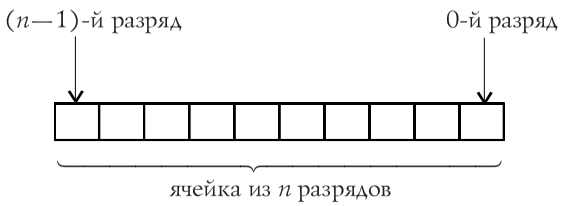
Следует отметить: двоичная система издавна была предметом пристального внимания многих ученых. Великий немецкий математик Г.Ф. Лейбниц, изрядное время уделивший двоичной (бинарной) математике, видел в ней "...прообраз творения". Он считал, что "единица представляет божественное начало, а нуль — небытие. Высшее существо создает все сущее из небытия точно таким же образом, как единица с помощью нуля выражает все числа".

### 4.3. Двоичное кодирование и компьютер

В конце XX века, века компьютеризации, человечество пользуется двоичной системой ежедневно, т.к. вся информация, обрабатываемая современными ЭВМ, хранится в них в двоичном виде. Каким же образом осуществляется это хранение?

Чисто конструктивно, информация в ЭВМ хранится в ячейках памяти, а обрабатывается в регистрах арифметико-логического устройства (АЛУ). И регистр АЛУ, и ячейка памяти представляют собой физическую систему, состоящую из некоторого числа однородных элементов.

Такой элемент способен находиться в нескольких состояниях и служит для изображения одного из разрядов числа. Именно поэтому каждый элемент ячейки называют ***разрядом***.




Нумерацию разрядов в ячейке принято вести справа налево, причем самый левый разряд имеет порядковый номер 0.

Если при записи чисел в ЭВМ мы хотим использовать обычную десятичную систему счисления, то, как легко понять, в каждом разряде может быть одна из десяти цифр, которым должны соответствовать 10 устойчивых состояний. Такие машины существуют. Однако конструкция элементов такой машины оказывается чрезвычайно сложной, что сказывается на надежности и скорости работы ЭВМ.

Наиболее надежной и дешевой является конструкция, когда каждый разряд может принимать всего лишь два состояния: намагничено — не намагничено, высокое напряжение — низкое напряжение и т.д. В современной электронике развитие аппаратной базы ЭВМ идет именно в этом направлении.

Следовательно, использование двоичной системы счисления в качестве внутренней системы представления информации вызвано чисто конструктивными особенностями элементов вычислительных машин.

### 4.4. Единицы измерения объема информации в компьютере

В современные компьютеры мы можем вводить текстовую информацию, числовые значения, а также графическую и звуковую информацию. Количество информации, хранящейся в ЭВМ, измеряется ее "длиной" (или "объемом"), которая выражается в ***битах*** (от английского ***binary digit*** — двоичная цифра). Бит — минимальная единица хранения информации, в каждом бите может храниться или 0, или 1. Для измерения объема хранимой информации используются следующие единицы:

1 байт = 8 битам;
1 Кбайт = 1024 байтам
(Кбайт читается как килобайт);
1 Мбайт = 1024 Кбайтам
(Мбайт читается как мегабайт);
1 Гбайт = 1024 Мбайтам
(Гбайт читается как гигабайт).

Для того чтобы запомнить эту таблицу единиц измерения объема информации, приведем известную программистскую шутку: "У программиста в одном километре 1024 метра".

Число 1024 как множитель при переходе к более высокой единице измерения обязано своим происхождением двоичной системе счисления (1024 — это десятая степень двойки):

1 Кбайт = $2^{10}$ байтам;

1 Мбайт = $2^{10} \cdot 2^{10}$ байтам = $2^{20}$ байтам

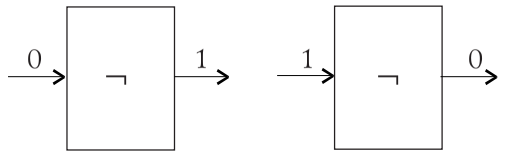
– и так далее.

### 4.5. Логические элементы ЭВМ

Работа современных вычислительных машин по своей сути сводится к обработке последовательностей нулей и единиц, которыми закодирована различная информация (числовая, текстовая, графическая, звуковая), и пересылке этой информации. Обработку двоичной информации производит ***арифметико-логическое устройство***, являющееся частью процессора. Состоит оно из ***логических элементов***.

***Логические элементы*** — это электронные схемы с одним или несколькими входами и одним выходом, через которые проходят электрические сигналы, представляющие цифры 0 и 1 и реализующие различные логические операции.

Так, простейшим логическим элементом является элемент НЕ. Он имеет один вход и один выход, и работа этого элемента состоит в инвертировании (то есть в замене на противоположный) значения поступившего в него сигнала:




Построение логических схем из логических элементов состоит в следующем:

- Прежде всего выписывается таблица истинности для функции, которую необходимо реализовать.
- Затем по каждой строке таблицы, в которой в столбце результата стоит 1, выписывается следующая комбинация входных переменных, элементы которой соединяются логической операцией “И”: если переменная в этой строке имеет значение 0, то в эту логическую комбинацию записывается ее отрицание, а если 1, то сама переменная.
- Полученные комбинации совмещаются с помощью логической операции “ИЛИ”.
- После того как функция восстановлена по таблице, по законам алгебры логики производится ее сокращение (преобразование к наиболее короткому виду), а уж затем она реализуется с помощью логических элементов, ее составляющих.

Проиллюстрируем эту процедуру путем построения логических функций для одного элемента многоразрядного ***сумматора*** — устройства для сложения двух многоразрядных двоичных чисел. Один элемент сумматора отвечает за сложение двоичных цифр из одного и того же разряда слагаемых и имеет три входа (так как, помимо значений двух суммируемых разрядов ($a$ и $b$), мы должны учитывать значение переноса из младшего разряда $p$) и два выхода: младшую цифру результата суммирования ($r$) и значение переноса в старший разряд ($p_1$):

| $a$ | $b$ | $p$ | $r$ | $p_1$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

Эта таблица соответствует следующим логическим функциям:

$$r(a, b, p) = \bar{a}\,\&\,\bar{b}\,\&p \vee \bar{a}\,\&b\&\,\bar{p} \vee a\&\,\bar{b}\,\&\,\bar{p} \vee a\&b\&p =$$

$$= p\&(\bar{a}\,\&\,\bar{b} \vee a\&b) \vee \bar{p}\,\&(\bar{a}\,\&b \vee a\&\,\bar{b}\,) =$$

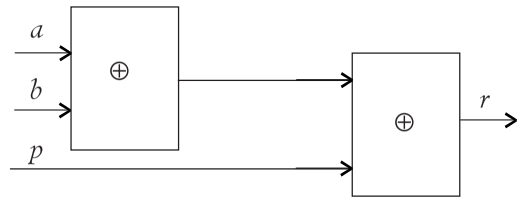
$$= p\&(\overline{a \oplus b}) \vee \bar{p}\,\&(a \oplus b) = a \oplus b \oplus p$$ (при построении использованы вторая, третья, пятая и восьмая строки таблицы истинности);

$$p_1(a, b, p) = \bar{a}\,\&b\&p \vee a\&\,\bar{b}\,\&p \vee a\&b\&\,\bar{p} \vee a\&b\&p =$$

$$= p\&(\bar{a}\,\&b \vee a\&\,\bar{b}\,) \vee a\&b\&(p \vee \bar{p}\,) = p\&(a \oplus b) \vee a\&b$$

(при построении использованы четвертая, шестая, седьмая и восьмая строки таблицы истинности).

Таким образом, например, логическую схему функции $r$ сумматора можно реализовать с помощью двух логических элементов *xor* (исключающее ИЛИ, обозначаемое также $\oplus$):




При построении логической схемы для этой и других функций всегда можно обойтись лишь тремя видами логических элементов: НЕ, ИЛИ и И. Объясняется это тем, что по описанной выше процедуре мы можем с помощью только этих операций записать любую логическую функцию, таблица истинности которой нам известна. Однако применение в нашем примере операции ***xor*** позволило сделать логическую схему максимально простой.

***Вопросы и задания***

**1.** Почему двоичная система используется для представления информации в ЭВМ?

**2.** Приведите примеры использования двоичного кодирования информации в повседневной жизни.

**3.** Известно, что некая информация занимает в компьютере 3,3 Кбайта памяти. Выразите это значение в байтах.





**4.** Допустим, что ячейка памяти компьютера содержит 16 разрядов. Какой номер имеет десятый справа разряд?

**5.** Приведите пример высказывания, относительно которого можно сказать, истинно оно или ложно.

**6.** Приведите пример высказывания, относительно которого нельзя сделать заключение о его истинности или ложности.

**7.** Используя двоичное кодирование для значений высказываний, решите следующую логическую задачу:

“Виктор, Роман, Юрий и Сергей заняли на математической олимпиаде первые четыре места. Когда их спросили о распределении мест, они дали три таких ответа:

1) Сергей — первый, Роман — второй;
2) Сергей — второй, Виктор — третий;
3) Юрий — второй, Виктор — четвертый.

Как распределились места, если в каждом ответе только одно утверждение истинно?”

**8.** Нарисуйте логическую схему для функции переноса в следующий разряд сумматора:

$$p_1(a, b, p) = p\&(a \oplus b) \vee a\&b.$$

**9.** Обозначим $S(a, b, p)$ логическую схему для одного разряда сумматора, имеющую на выходе значение функций $r$ и $p_1$. Изображая эту схему как один элемент с тремя входами и двумя выходами, нарисуйте схему для четырехразрядного сумматора. В первом разряде на входе в $S(a, b, p)$ вместо $p$ можно подавать константу 0.

## Тема 5. Двоичная арифметика

Более подробно материал этой темы изложен в главе 2 второй части.

В двоичной системе счисления для записи чисел используются только две цифры: 0 и 1, основание двоичной системы равно 2. Запись числа $110_2$, например, означает число 6, т.к. старшая единица в значение числа вносит $2^2$, а вторая единица вносит $2^1$. Выпишем первые 9 чисел двоичной системы

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 10 | 11 | 100 | 101 | 110 | 111 | 1000 |

Двоичное число представляет собой цепочку нулей и единиц. Используя развернутую форму записи (определение 7), любое число в двоичной системе счисления можно представить в виде суммы степеней двойки.

**Пример 14.** Приведем двоичную запись некоторых степеней числа десять:

$$10_{10} = 1010_2$$
$$100_{10} = 1100100_2$$
$$1000_{10} = 1111101000_2$$
$$10000_{10} = 10011100010000_2$$

Из приведенного примера видно, что самым существенным недостатком двоичной системы является слишком большое количество цифр в записи числа.

Зато у нее есть и довольно большие плюсы.

Вспомним, что все позиционные системы счисления “одинаковы”, а именно: арифметические операции везде выполняются по одним и тем же правилам. Для всех систем счисления справедливы одни и те же законы арифметики: коммутативный, ассоциативный, дистрибутивный, а также правила сложения, вычитания, умножения и деления столбиком. Эти правила знакомы нам по действиям в десятичной системе счисления и опираются на таблицы сложения и умножения десятичных цифр.

Для того чтобы производить арифметические операции над двоичными числами, надо знать таблицы сложения и умножения в двоичной системе.

Приведем их.

Таблица сложения:

| + | **0** | **1** |
|---|---|---|
| **0** | 0 | 1 |
| **1** | 1 | $10_2$ |

Таблица умножения:

| × | **0** | **1** |
|---|---|---|
| **0** | 0 | 0 |
| **1** | 0 | 1 |

### 5.1. Сложение

Как легко видеть, таблица сложения в двоичной системе предельно проста. Так как 1+1=10, то при сложении столбиком в данном разряде остается 0, а 1 переносится в следующий разряд.

Рассмотрим несколько примеров выполнения операции сложения в двоичной системе.

**Пример 15**

$$\begin{array}{r} 1001_2 \\ +\ 1010_2 \\ \hline 10011_2 \end{array} \qquad \begin{array}{r} 1111_2 \\ +\ \ 1_2 \\ \hline 10000_2 \end{array}$$

**Пример 16**

Сложение столбиком действительных чисел.

$$\begin{array}{r} 101,011_2 \\ +\ \ 1,11_2 \\ \hline 111,001_2 \end{array}$$

Из приведенных примеров видно, что при сложении столбиком двух цифр справа налево в двоичной системе, как и в любой позиционной системе счисления, в следующий разряд может переходить только единица. А результат выполнения сложения в новом разряде все равно будет меньше, чем $2 \cdot P$ (максимум $2 \cdot P$—1, что

# Практикум по информатике в среде LogoWriter

Продолжение. Начало на с. 5

Таким образом, овладение программированием рассматривается в курсе как средство, позволяющее ученику, например, моделировать движение (а значит, глубже понять законы Ньютона), создавать "коллекции" графиков функций (и при этом освоить влияние коэффициентов в уравнениях на сдвиги и растяжения-сжатия кривых), заинтересоваться закономерностями красивых геометрических композиций, явившихся результатом программы, и т.д. Можно сказать, что важнейшей задачей курса информатики автор считает развитие познавательной самостоятельности старшеклассников в изучении информатики, математики, физики.

Работа учащихся по предлагаемому пособию позволяет затем (при желании учителя) осуществить переход к проектной форме работы, при которой учащиеся выбирают темы для индивидуальных проектов исследовательского характера, рассчитанные на работу в течение одного полугодия или всего учебного года. Темы могут быть связаны с математикой и физикой (моделирование и исследование колебаний, фрактальные кривые, исследование графиков функций, построение картины электростатического поля и др.). Сборник таких проектов (учебных исследований) автор надеется вскоре предложить коллегам-учителям.

## Глава 7. ПОСТРОЕНИЕ ГРАФИКОВ ФУНКЦИЙ

### § 27. Движение Черепашки в декартовых координатах

Перемещение Черепашки может осуществляться посредством команд `FD`, `BK`, `RT` и `LT`, которые продвигают и поворачивают Черепашку относительно ее предыдущего положения. Это команды относительного перемещения, результирующее положение и направление Черепашки зависят от предыдущих команд.

Существует другая возможность перемещать Черепашку, задавая координаты точек. Каждая точка экрана имеет координаты, начало отсчета — в центре. Для перемещения Черепашки по горизонтали до точки с заданной абсциссой служит команда `SETX` (установи $x$). Например, команда `SETX 100` передвинет Черепашку до точки с абсциссой 100.

При повторном вводе той же команды Черепашка останется на месте; `SETX` — команда абсолютного перемещения.

Для перемещения по вертикали до точки с заданной ординатой служит команда `SETY` (установи $y$). После нее также нужно указать число.

Кроме того, есть два датчика `X` и `Y`, они позволяют узнать положение Черепашки. Например, можно вывести на экран координаты Черепашки командами `PR X` и `PR Y` (точнее, это будут координаты точки, находящейся под центром формы Черепашки).

При выполнении всех заданий в этой главе для перемещений Черепашки можно использовать только команды абсолютного перемещения `SETX` и `SETY` (вместо команд `FD`, `BK`, `RT` и `LT`).

***Задания***

*27.1. Напишите процедуру, которая рисует координатные оси.*

*27.2. Напишите процедуру, рисующую прямоугольник в первой четверти координатной плоскости. Его стороны не должны лежать на осях. Напишите такие же процедуры для трех других четвертей.*

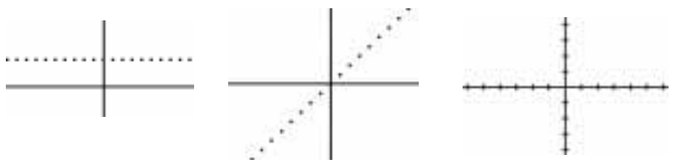
*27.3. Напишите процедуру с двумя параметрами, которая ставит (отпечатывает) точку с заданными координатами. Используя ее, поставьте по три точки в каждой из четвертей так, чтобы они могли быть вершинами непрямоугольных треугольников.*

*27.4. Напишите процедуру, которая ставит точки одну за другой параллельно оси абсцисс.*

*В программе должна многократно вызываться процедура из задания 27.3. Каждый раз абсцисса точки немного увеличивается. Ордината точки все время остается неизменной.*

*27.5. Напишите процедуру, располагающую точки наклонно (так, как показано на рисунке).*

*Необходимо, чтобы в цикле изменялась не только абсцисса точки, но и ее ордината. Ордината точки зависит от абсциссы.*



*27.6. Напишите процедуру для разметки координатных осей штрихами с некоторым шагом.*

*Сначала напишите четыре отдельные процедуры для разметки осей от центра вправо, влево, вверх и вниз. Расстояние от центра до очередного штриха — переменная величина. Ее значение нужно изменять в цикле.*

*27.7. Введите в процедуру разметки осей параметр — шаг разметки; свяжите с ним число повторений каждого цикла (так, чтобы при любом шаге оси были размечены на максимальную длину).*

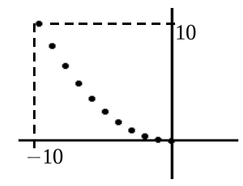
### § 28. Построение точечного графика функции

Как строятся графики функций в тетради? По точкам, которые затем соединяются. Если точки расположены достаточно часто, то их не обязательно соединять — получится точечный график. Для каждой точки произвольно устанавливается значение абсциссы и по заданной формуле вычисляется значение ординаты.

По оси $x$ точки удобно расположить равномерно, на небольшом расстоянии друг от друга. Значит, абсцисса каждой точки (кроме самой первой) будет отличаться от абсциссы предыдущей точки на одну и ту же величину — шаг по оси $x$.

$$y = 0{,}1\ x^2$$

| $x$ | −10 | −9 | −8 | −7 | −6 | −5 | −4 | −3 |
|---|---|---|---|---|---|---|---|---|
| $y$ | 10 | 8,1 | 6,4 | 4,9 | 3,6 | 2,5 | 1,6 | 0,9 |




Программу можно написать в виде цикла с изменяющейся переменной — абсциссой очередной точки. Сценарий программы:

- выполнить начальные установки формы, цвета Черепашки, начального значения переменной — абсциссы первой точки графика;
- в цикле повторять следующие действия:
  a) вычислить по формуле значение ординаты;
  b) поставить точку с получившимися координатами (можно воспользоваться готовой процедурой из задания 27.3);
  c) изменить значение переменной (абсциссы точки) в цикле.

***Задания***

*28.1. Напишите процедуру для построения графика функции* $y = 0{,}1\ x^2$.

*Команда для вычисления ординаты —* `MAKE "Y 0,1 * :X * :X`

*28.2. Подберите такие значения начальной абсциссы и количества повторений, чтобы график занимал максимальную область, но Черепашка, ставя точки, не пересекала бы границы экрана. Введите в процедуру параметр — шаг по оси $x$.*

*28.3. Напишите процедуры для построения графиков функций:*

$y = 50 \sin 5x$; $y = x \sin 5x$

*Команды для вычисления ординат будут выглядеть так:*

```
MAKE "Y 50 SIN 5 * :X
MAKE "Y :X * SIN 5 * :X
```

*Введите в процедуры второй параметр — цвет графика. Как и в предыдущем задании, добейтесь того, чтобы каждый график занимал максимальную область экрана.*

### § 29. Универсальная программа — построитель графиков

Три процедуры для трех разных графиков из заданий предыдущего параграфа очень похожи. Менялась формула вычисления; кроме того, приходилось подбирать начальное значение абсциссы и число точек, чтобы график "не вылезал" за границы экрана, а также и шаг, чтобы график не был слишком "редким". Как автоматизировать этот подбор и сделать универсальную процедуру для разных графиков?

Прежде всего можно предотвратить появление "лишних" точек графика. Они оказываются на экране в том случае, если ордината очередной точки слишком велика (больше 180) или слишком мала (меньше −180). Черепашка при этом пересекает границу экрана и появляется с противоположной стороны. Чтобы избежать этого, нужно изменить процедуру, которая ставит точку, сделать это действие условным. То есть ввести проверку (команду условного ветвления). Если выполняется двойное неравенство $-180<y<180$, то "отпечатать" точку; в противном случае ничего не делать. Такое составное условие задается через два простых условия: ордината меньше 180 и ордината больше −180:

```
AND   Y < 180    Y > -180
```

`AND` (и) — логическая связка для двух простых условий.

После внесения изменений в процедуру отпечатывания точки можно строить графики во всю ширину экрана — от левой границы (−300) до правой (300) (если только у этой функции нет особых точек). А количество точек надо вычислять исходя из расстояния от левой границы экрана до правой, а также шага.

После этого процедуры для построения разных графиков будут отличаться только командой вычисления ординаты по той или иной формуле. Чтобы избежать описания отдельной процедуры для каждого нового графика (они все будут однотипные), можно поступить следующим образом: вычисление ординаты выделить во вспомогательную процедуру — "копилку" функций.

```
ЭТО ВЫЧИСЛЕНИЕ_ОРДИНАТЫ  :N  :X
IF :N = 1 [MAKE  "Y 0.1  *  :X  *  :X ]
IF :N = 2 [MAKE  "Y 50  *  SIN 5  *  :X ]
IF :N = 3 [MAKE  "Y :X  *  SIN 5  *  :X ]
END
```

Вызов этой вспомогательной процедуры будет первым действием в цикле универсальной процедуры построения графиков. А номер функции, занесенной в "копилку", станет третьим параметром (после шага и цвета). "Копилку" можно сколько угодно расширять, добавляя туда новые функции.

***Задания***

*29.1. Введите условие в процедуру, которая ставит точку с заданными координатами.*

*29.2. Напишите процедуру-копилку функций.*

*29.3. Напишите процедуру с тремя параметрами (номер функции, шаг и цвет) по следующему сценарию:*

- *выполнить начальные установки формы, цвета Черепашки, начального значения абсциссы первой точки графика (на левой границе экрана);*
- *в цикле каждый раз выполнять следующие действия:*
  *a) вычислить ординату (вызвать процедуру-копилку);*
  *b) поставить точку (теперь это действие — условное);*
  *c) изменить значение переменной (абсциссы) на шаг.*

*Цикл надо повторить столько раз, сколько точек с заданным шагом уместится в ширину экрана.*

*Проверьте правильность работы для всех трех функций, записанных в "копилке".*

*Выполняя следующие задания, добавляйте в "копилку" линейные функции.*

*29.4. Получите на экране координатные оси и 10 прямых, проходящих через начало координат (пучок прямых должен быть по возможности равномерной "густоты").*

*29.5. Постройте две взаимно перпендикулярные прямые, проходящие через начало координат. Сформулируйте зависимость между их уравнениями.*

*29.6. Проведите несколько прямых, параллельных оси абсцисс, на равном расстоянии друг от друга.*

*29.7. Добавьте в "копилку" следующие функции и постройте их графики:*

$$y = 50\ (\sin x) \cdot (\cos 15x)$$
$$y = 50\ (\sin 3x) \cdot (\cos 25x)$$

## § 30. Изменение масштаба графика

Если попробовать построить график обыкновенной параболы $y=x^2$, то изображение получится слишком мелкое, т.к. одна единица (одно деление на осях) равняется одному шагу Черепашки. Во многих случаях удобно было бы рассматривать графики "под лупой". Другими словами, необходимо добавить в программу возможность изменять масштаб изображения.

Пусть мы хотим "растянуть" график по осям в *M* раз. Значит, в *M* раз должны увеличиться абсцисса и ордината каждой точки графика. При вызове процедуры, которая ставит точку, надо указать в качестве параметров не `:X` и `:Y`, а `(:X * :M)` и `(:Y * :M)`.

Вся картинка "вырастет" в *M* раз и, конечно, займет больше места на экране. Поэтому начальное значение абсциссы (—300) нужно уменьшить в *M* раз. То есть команда установки начальной абсциссы должна принять вид:

```
MAKE  "X  -300 / :M
```

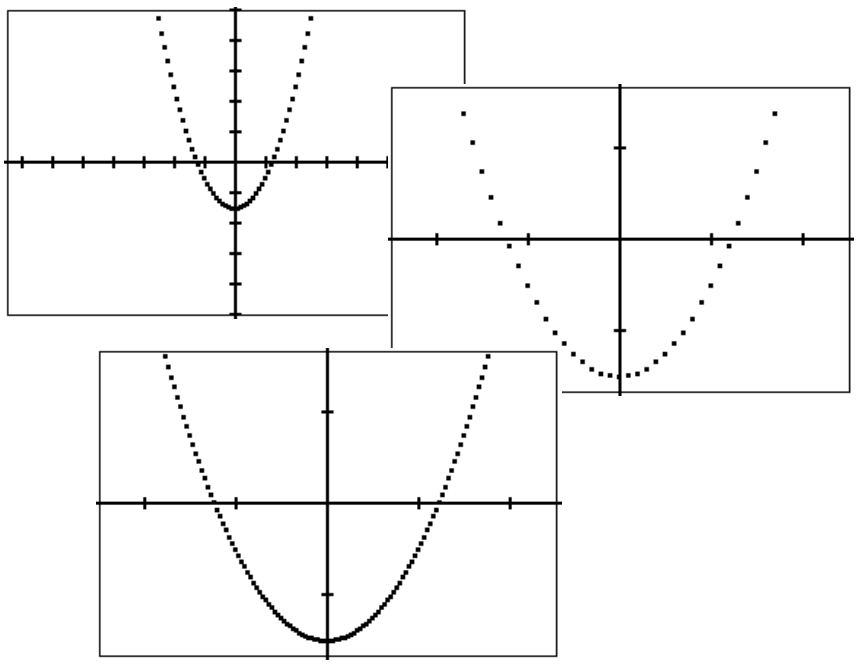
Последнее изменение необходимо внести в то место в программе, где увеличивается абсцисса. Эта команда выглядела так:

```
MAKE  "X :X + :шаг
```

При увеличении координат каждой точки расстояния между соседними точками также увеличатся и график станет "редким". (Представьте себе, что вы рассматриваете точечный график "под лупой".) Чтобы плотность точек не изменялась, необходимо уменьшить шаг (пропорционально растяжению).

```
MAKE  "X :X + :шаг / :M
```

На рисунках показан один и тот же график: на первом — с увеличением в 20 раз, а на двух других — с увеличением в 60 раз.



При построении второго графика шаг не был уменьшен, поэтому и расстояния между точками выросли втрое, что нежелательно.

Чтобы видеть на экране, в каком масштабе построен график, удобно использовать размеченные оси координат из задания 27.7. Расстояние между штрихами должно быть равно коэффициенту увеличения. Если изображение увеличено, скажем, в 30 раз, то единичный отрезок равен 30 (это как раз и есть расстояние между штрихами).

Следовательно, перед построением графика надо получить на экране оси, размеченные через *M* единиц. Для этого достаточно вызвать процедуру из задания 27.7, задав значение параметра, равное *M*.

Рядом с графиками можно указать формулы соответствующих функций. Для создания надписей существует специальный режим. Для перехода в этот режим нужно нажать на клавишу F8. Курсор окажется на рабочем поле. Перемещение курсора осуществляется с помощью клавиш ↑ → ← ↓.

При печатании надписей может возникнуть необходимость исправить ошибку. Клавиши BackSpace и Del в режиме надписей не работают, поэтому обычные способы исправления не подойдут. Чтобы стереть ошибку, надо набрать тот же символ на том же месте. Для выхода из режима надписей достаточно нажать клавишу Esc.

Построенные графики можно собирать в альбом, сохраняя картинки и затем считывая их при помощи команд `SAVEPIC` и `LOADPIC`.

***Задания***

*30.1. Подберите масштаб для функций и решите графически уравнения:*

$$x^4 - 3x^2 - x + 0{,}5 = 0 \qquad 2x^3 + x^2 - 2x + 1 = 0$$

*Для каждого из заданий 30.2—30.6 сформулируйте, какими должны быть коэффициенты уравнения прямой. При построении графиков выбирайте разные цвета. Графики подпишите и картинки сохраните для альбома графиков функций.*

*30.2. Проведите несколько прямых через точку, лежащую на оси ординат.*

*30.3. Проведите несколько прямых через точку, лежащую на оси абсцисс.*

*30.4. С помощью процедуры из задания 27.3 поставьте цветную точку с заданными координатами (не лежащую на осях) и проведите через нее пучок прямых.*

*30.5. Расчертите экран "наискосок" параллельными прямыми через равные промежутки.*

*30.6. Дополните рисунок из предыдущего задания серией параллельных прямых, перпендикулярных первой серии (должны получиться "клеточки").*

*30.7. Поставьте на экране две точки с заданными координатами и проведите через них прямую.*

*30.8. Постройте параболу $y=x^2$ с увеличением в 20—40 раз.*

*30.9. Получите "перевернутую" параболу с вершиной в начале координат.*

*30.10. Постройте серию парабол с вершинами, лежащими на оси ординат.*

*30.11. Постройте серию парабол с вершинами, лежащими на оси абсцисс.*

*30.12. Постройте серию парабол различной "крутизны" с вершинами в начале координат.*

*30.13. Получите на одном рисунке оси; параболу $y=x^2$; ту же параболу, сдвинутую на 4 единицы вниз; ту же параболу, сдвинутую на 3 единицы вправо; и ту же параболу, сдвинутую на 4 единицы вниз и на 3 единицы вправо. При построении графиков выбирайте разные цвета. Графики подпишите и картинку сохраните для альбома графиков функций.*

*30.14. Постройте параболу с вершиной в заданной точке (не лежащей на осях).*

*30.15. Постройте параболу с корнями 1 и —3; по графику найдите минимум функции.*

*30.16. На том же рисунке постройте еще две параболы с теми же корнями и с минимальными значениями, равными —2 и —1 (для этого нужно подобрать коэффициент при $x^2$). Графики подпишите и картинку сохраните для альбома.*

*30.17. Постройте график многочлена с корнями —7, —1, 2; подберите коэффициент при старшей степени $x$ так, чтобы были видны максимумы и минимумы. Подпишите картинку и сохраните ее для альбома.*

*30.18. Решите графически уравнение: $0{,}01\,x^3 - 0{,}19\,x^2 - 0{,}22\,x + 0{,}4 = 0$*

## § 31. Построение графиков тригонометрических функций

Угол для поворотов Черепашки надо указывать в градусной мере. Функции `SIN` и `COS` в LogoWriter также вычисляют результат от аргумента, заданного в градусах. Например, набрав команду `PR SIN 30`, вы получите на экране число 0,5, а набрав `PR COS 90` — число 0.

Для построения графиков градусная мера неудобна, вместо нее используют более "крупную" единицу — радиан. В 180 градусах — "пи" радиан (это число — отношение длины окружности к ее радиусу — примерно равно 3,1415). Значит, чтобы перевести любое число градусов в радианы, нужно его умножить на 180 и разделить на 3,1415. Превратить тригонометрические функции от градусного аргумента в функции от радианного аргумента поможет специальная команда `OP` (`output` — вывести, считать результатом):

```
ЭТО #SIN :X
OP  SIN :X * 180 / 3,1415
END
```

С помощью команды `OP` записано действие: считать результатом вычисления функции `#SIN` от некоторого числа результат вычисления другой функции — `SIN` от этого же числа, умноженного на коэффициент 180 / 3,1415.

***Задания***

*31.1. Опишите тригонометрические функции синуса и косинуса от аргумента, выраженного в радианах.*

*31.2. Постройте на одном рисунке с увеличением в 30—50 раз графики функций $y = \sin x$ $y = \cos x$*

*31.3. На том же рисунке постройте синусоиду, сдвинутую вверх на три единицы. Графики подпишите и картинку сохраните для альбома.*

*31.4. Постройте на одном рисунке графики функций*

$$y = \sin x \quad y = 5\sin x \quad y = \sin 5x$$



Окончание на с. 12

# Практикум по информатике в среде LogoWriter

**Окончание. См с. 5, 6, 11**

*31.5. Постройте график функции* $y = (\sin x) \cdot (\cos x)$
*На этом же рисунке постройте другой график, подтверждающий тригонометрическое тождество.*

*31.6. Постройте на одном рисунке графики функций*

$y = \sin x$ $\qquad$ $y = \cos x$

$y = (\sin^2 x) + (\cos^2 x)$ $\qquad$ $y = (\sin^2 x) - (\cos^2 x)$

## РЕКОМЕНДАЦИИ ПО ПРОВЕДЕНИЮ УРОКОВ И ТЕКСТЫ ПРОГРАММ К ЗАДАНИЯМ

*Слышу — и забываю,*
*вижу — и запоминаю,*
*делаю — и понимаю.*
(Восточная мудрость)

Учитель математики обычно использует ограниченный набор средств: доску и мел, тетрадь и ручку. В соответствии с этими средствами в течение длительного времени происходил отбор тем и формировалась методика. В ряде случаев применение новых технических средств может существенно изменить школьную математику. Ежегодно с 7-го класса школьники проходят по алгебре раздел о функциях (в 7-м — линейная зависимость, в 8-м — квадратичная и т.д.). Но только в десятом классе учащиеся знакомятся с влиянием коэффициентов на сдвиги и сжатия-растяжения графиков функций. В девятом классе заучивают тригонометрические формулы, а с графиками тригонометрических функций знакомятся только на следующий год. В этом разделе математики компьютер можно использовать для повышения наглядности, а также и как инструмент для математических экспериментов. Именно недостаток наглядности и отсутствие возможности осваивать функциональные зависимости опытным путем приводят к тому, что большая часть школьников, не имеющих "готового" хорошо развитого формально-логического мышления, попадает в разряд неспособных к математике. Для них освоение этого раздела сводится к овладению отдельными утилитарными навыками типа нахождения корней квадратного трехчлена.

Работая над этой главой пособия, школьники сначала самостоятельно подготовят инструмент — построитель графиков, а затем получат возможность строить десятки графиков, экспериментировать с коэффициентами, подбирать формулы для причудливых кривых, собирать красивые экземпляры в альбомы графиков. То есть смогут почувствовать красоту математики, пусть и лежащую на самой поверхности. Но основное — это углубление понимания.

Зачем учителю информатики "вторгаться" в другую предметную область?

Во-первых, учитель получает хорошую возможность показать своим ученикам, зачем они учились разрабатывать алгоритмы, писать и отлаживать программы. Все это они могут применить уже сегодня в своей главной деятельности — школьной учебе.

Во-вторых, связав свою работу с основными школьными предметами, учитель информатики повышает свой "внутришкольный" статус. Можно сказать, включается в общее дело — решение базовых задач школы. (Заметим, что освоение офисных технологий — это скорее предпрофессиональная подготовка, а не задача базовой школы.)

Кроме того, на уроках информатики по этой теме учащиеся знакомятся с учебной исследовательской работой. Ведь они должны не только написать программу, но и научиться использовать ее для небольшого исследования: провести эксперименты, собрать данные (в числовом или графическом виде), проанализировать полученные результаты, сформулировать выводы. Наконец, подготовить простую презентацию (электронный альбом графиков).

Важный момент в работе учителя — как ставить оценки? На компьютерных уроках особенно большую роль играет разброс в индивидуальных способностях. Да и те, у кого дома уже есть компьютер, оказываются в привилегированном положении. Лучше плохих отметок не ставить совсем, стимул для учащихся на таких уроках — привлекательность работы на компьютере.

В большинстве случаев рекомендуемый текст программы — только один из нескольких возможных вариантов. Конечно, вы должны всемерно поощрять неодинаковость решений, стремление сделать по-своему. Сравнение различных подходов даст вам прекрасную возможность вызвать учеников на обсуждение, спор, соревнование.

### § 27. Движение Черепашки в декартовых координатах

В предыдущих параграфах все графические задачи решались с помощью команд относительного перемещения FD, BK и относительного поворота RT, LT. То есть в относительных полярных координатах. Переход к другому способу получения изображений, к абсолютным декартовым координатам существенно расширяет круг возможностей, но и вызывает естественные затруднения. Обратите внимание ваших учеников на то, что, решая задания этой главы, они не должны пользоваться командами FD, BK, RT и LT.

```
27.3
это тчк :x :y
setsh 1
pu setx :x sety :y pd stamp
end
```

```
27.5
это оси :M
ox1 :M ox2 :M oy1 :M oy2 :M
end

это ox1 :M
make "S 0
repeat 300 / :M [setx :S штрих1 make "S :S + :M]
setx 0
end

это ox2 :M
make "S 0
repeat 300 / :M [setx :S штрих1 make "S :S - :M]
setx 0
end

это oy1 :M
make "S 0
repeat 180 / :M [sety :S штрих2 make "S :S + :M]
sety 0
end

это oy2 :M
make "S 0
repeat 180 / :M [sety :S штрих2 make "S :S - :M]
sety 0
end

это штрих1
sety 5 sety 0
end
это штрих2
setx 5 setx 0
end
```

### § 28. Построение точечного графика функции

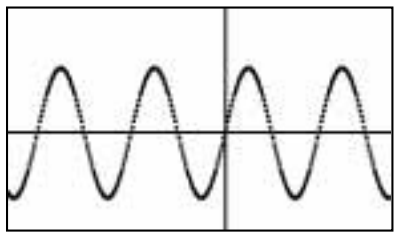
Ваши ученики, вероятно, привыкли строить график параболы в тетради в два приема: от нуля вправо, затем от нуля влево. Подскажите им, что в программе нецелесообразно повторять такую последовательность действий. Начальное значение абсциссы естественно выбирать в левой полуплоскости (отрицательное число) и затем последовательно ставить точки слева направо, увеличивая абсциссу.

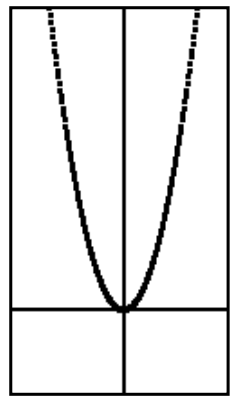
Сценарий программы, приведенный в пособии, соответствует итеративному построению процедуры. Однако, работая над главами 5 и 6, школьники освоили оба способа организации повторений в программах: и итерацию, и рекурсию. Теперь они могут выбирать тот способ, который им больше понятен. Поэтому приведены оба варианта решений.

```
28.1
это график1
setc 14 setsh 1
make "x -40
repeat 80
[make "y 0,1 * :x * :x
тчк :x  :y
make "x :x + 1]
end

это график2 :x
if :x > 80 [stop]
тчк :x  0,1 * :x * :x
график2 :x + 1
end
```

### § 29. Универсальная программа — построитель графиков

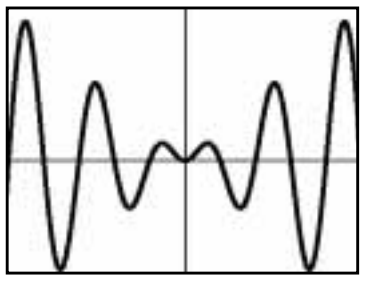
```
29.1
это тчк :x :y
[pu setx :x sety :y pd stamp]
end

29.2
это функция :N :x
if :N = 1 [make "y :x * :x * 0.1]
if :N = 2 [make "y 50 * sin 5 * :x]
if :N = 3 [make "y :x * sin 5 * :x ]
end

29.3
это график :N :цвет :шаг
setc :цвет setsh 1
make "x -300
repeat 600 / :шаг     [функция :N :x
тчк :x  :y
make "x :x + :шаг]
end
```





Рекурсивный вариант:

```
это график :N :x :шаг :x_кон
if :x > :x_кон [stop]
функция :N :x тчк :x  :y
график :N :x + :шаг :шаг
end
```



Перед запуском рекурсивной процедуры построения графика необходимо задать начальные установки: цвет и форму Черепашки. Параметры `:x` и `:x_кон` задают границы отрезка, на котором будет построен график.

## § 30. Изменение масштаба графика

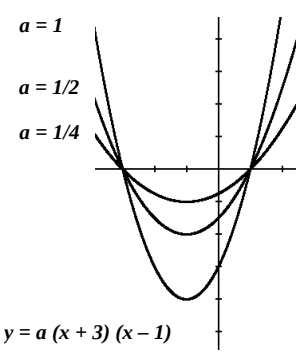
Введение масштаба — последнее усовершенствование программы. После этого можно провести разнообразные "лабораторные работы" по исследованию графиков. Большая часть заданий к этому параграфу предполагает построение некоторого семейства графиков по заданному признаку. После того как школьники справятся с этой задачей, необходимо помочь им в определении общей формулы полученного семейства. Часто такое задание вызывает затруднения.

```
30.1
это график :N :цвет :шаг :M
setc :цвет setsh 1
make "x -300 / :M
repeat 600 / :шаг     [функция :N :x
точечка :M * :x  :M * :y
make "x :x + :шаг / :M]
end
```

Рекурсивный вариант:

```
это график :N :x :шаг :x_кон :M
if :x > :x_кон [stop]
функция :N :x тчк :x  :y
график :N :x + :шаг / :M :шаг :M
end
```

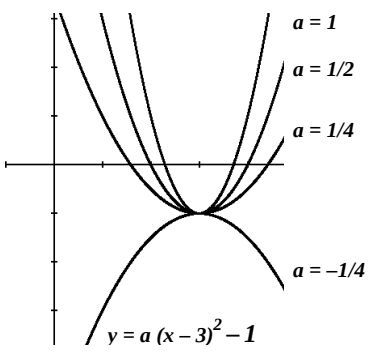
***Семейство парабол, имеющих общие корни –3 и 1***




При запуске рекурсивной процедуры нужно указывать границы отрезка, на котором строится график, с учетом масштаба. Поэтому сначала нужно построить оси с разметкой через *M* единиц и затем по картинке на экране задать начальное и конечное значения ***x***.

В блоке заданий с параболами необходимо, чтобы учащиеся связали с соответствующими заданиями различные способы представления квадратичной функции.

***Семейство парабол, имеющих общую вершину (3, –1)***



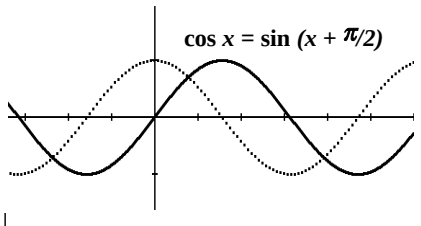

Для проведения параболы через две точки, лежащие на оси абсцисс, целесообразно использовать запись в виде произведения двучленов; для перемещения вершины в заданную точку — выделение полного квадрата.

Важно, чтобы ваши ученики не только получили правильные результаты, но и аккуратно и достаточно полно оформили их в виде "электронного альбома графиков". Желательно, чтобы на каждом листе были помещены поясняющие и обобщающие надписи: уравнение для каждой кривой, общая формула всего семейства кривых, название (заголовок) листа.

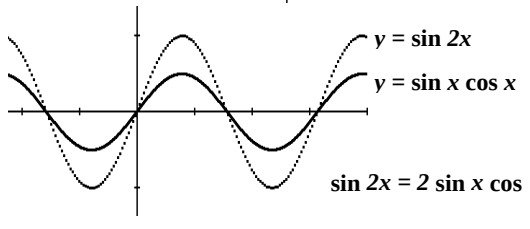
## § 31. Построение графиков тригонометрических функций

Задание 31.3 позволяет познакомить учащихся с влиянием коэффициентов на сжатия-растяжения графиков по осям.

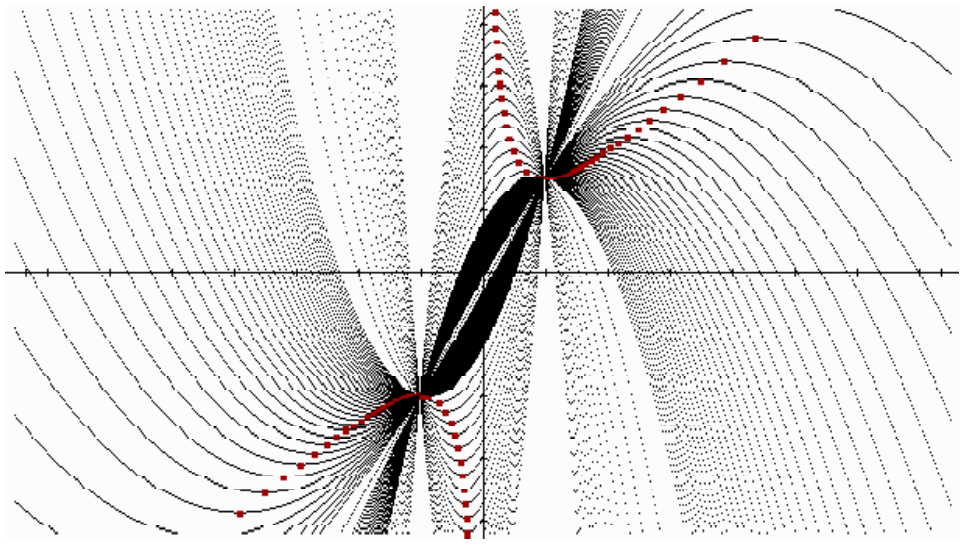
Задания 31.2, 31.4, 31.5 представляют наглядные "доказательства" тригонометрических тождеств.




Материал этой главы может послужить основой для индивидуальных проектов ваших учеников. Результаты такой работы учащегося вы сможете засчитать как сдачу экзамена по информатике в 9-, 10-, 11-м классах. Вот примеры тем для проектов:




- Построить семейства касательных к параболам, гиперболам, окружностям, эллипсам.
- Построить семейство парабол, имеющих две общие точки, не лежащие на одной горизонтали. Выяснить, как будут располагаться вершины этих парабол.
- Дополнить процедуру построения графика функции одновременным построением графика производной этой функции. (Нужно вычислять отношение приращения ординаты к шагу по оси ***x***.)
- Построить семейство графиков степенных функций (с натуральным, дробным, рациональным показателем).



Задания этой главы можно рассматривать и как подготовку к освоению задач с параметрами, которые обычно вызывают затруднения на вступительных экзаменах по математике.

*Продолжение в следующем номере*



## Институт информатизации образования Министерства образования России, Российский фонд компьютерных учебных программ при институте

**сообщают о продолжении работ по оценке качества педагогических программных средств**

**Недавно директор ИНИНФО Я.А. Ваграменко и руководитель Российского фонда компьютерных программ (РОСФОКОМ) А.И. Галкина вручили сертификаты Министерства образования фирмам "Интеллект-Сервис" и "1С" на следующие программные средства:**

Номер сертификата: 0000029
Дата выдачи сертификата: 30.12.98
Наименование программы: Комплексная бухгалтерская, торговая и складская система "БЭСТ-4"
Тип ЭВМ: IBM PC 486 и выше
Тип и версия ОС: MS DOS 5.0 и выше
Организация-разработчик: ЗАО "Интеллект-Сервис"

Номер сертификата: 0000032
Дата выдачи сертификата: 15.02.99
Наименование программы: "1С:Репетитор.Химия"
Тип ЭВМ: IBM PC
Тип и версия ОС: Windows 95
Авторы: Москаленко К.Л., Матвеев В.К., Галин А.М.
Организация-разработчик: АОЗТ "1С"

Номер сертификата: 0000030
Дата выдачи сертификата: 15.02.99
Наименование программы: "1С:Репетитор.Биология"
Тип ЭВМ: IBM PC
Тип и версия ОС: Windows 95
Автор: Москаленко К.Л., Элинзон Д., Дмитриева А.Г., Рябчикова Н.А.
Организация-разработчик: АОЗТ "1С"

Номер сертификата: 0000031
Дата выдачи сертификата: 15.02.99
Наименование программы: "1С:Репетитор.Физика"
Тип ЭВМ: IBM PC
Тип и версия ОС: Windows 95
Авторы: Москаленко К.Л., Берков А.В., Грибов В.А.
Организация-разработчик: АОЗТ "1С"

Работа фонда компьютерных учебных программ вышла на новый этап.

По решению Государственного комитета связи и информатизации и Министерства общего и профессионального образования Российской Федерации Фонду компьютерных учебных программ при ИНИНФО присвоен статус испытательной лаборатории (государственный аттестат аккредитации испытательной лаборатории № РОСИ.А115.643.22011 от 10 марта 1999 г.), что означает возможность ввода сертификата соответствия государственного образца на программную продукцию, прошедшую испытания в фонде.

Подробности о сертификации программной продукции, осуществляемой Фондом компьютерных учебных программ, читайте на страницах журнала "Компьютерные учебные программы". Кроме этого, в журнале публикуются информация о победителях лотереи среди подписчиков прошлого года и условия участия в лотерее этого года. В этом году разыгрываются 100 обучающих программ и несколько компьютеров.

**Адрес:** Россия,103041, Москва, ул. Рождественка, 6/20
**E-mail:** in.info@g23.relcom.ru
**Тел.** 924-61-97
**Факс** 925-29-69

Вручение сертификатов Министерства образования представителям фирм "Интеллект-Сервис" и "1С".


# РОБОТЛАНДСКИЙ СЕТЕВОЙ УНИВЕРСИТЕТ-2000

## осуществляет предварительный набор студентов на 1999/2000 учебный год

Прошло 3 года с начала вступления Роботландии в информационные сети со своей версией заочной школы дополнительного образования учителей и школьников.

Обучающая олимпиада 1996/97 учебного года, курсы 97/98, университет 98/99. Каждый год Роботландия привносит в свою сетевую школу что-то новое, экспериментальное, одновременно совершенствуя удачные формы коммуникативной работы, учебники, методику, дидактические материалы.

У Роботландии появился свой фанклуб. В него входят постоянные студенты: учителя и школьники, каждый год аккуратно вновь приступающие к обучению. Но это не двоечники, оставшиеся на второй (третий) год, это как раз самые лучшие ученики и фактически соавторы нашей школы. Ведь в основе обучения лежит тесная кооперация студентов и преподавателей, активная творческая деятельность. Члены фан-клуба знают: интересно будет каждый год.

Сегодня мы представляем предварительные планы на 1999/2000 учебный год и надеемся, что фан-клуб Роботландии пополнится новыми членами.

Традиционно в 1999/2000 учебном году университет откроет свои виртуальные классы для коллективных и индивидуальных учеников.

КОЛЛЕКТИВНЫЙ СТУДЕНТ — это группа детей, работающая под руководством одного или нескольких наставников. Наставником может быть школьный учитель, родитель, ученик старших классов. Число детей в группе не ограничено и может принимать любые разумные значения.

ИНДИВИДУАЛЬНЫЙ СТУДЕНТ — это учитель, желающий пройти обучение индивидуально, без группы детей. Такое обучение можно рассматривать как опережающее знакомство с методическими идеями роботландских курсов перед внедрением их у себя в школе или последующего обучения в университете вместе с группой детей.

В новом учебном году в составе университета предполагаются следующие годовые курсы:

### I. ВВЕДЕНИЕ В ИНФОРМАТИКУ (куратор — Ю.А. Первин)

Этот курс ориентирован только на индивидуальных учеников —учителей, преподающих или собирающихся преподавать информатику в младших классах на базе программно-методической системы “Роботландия”.

Эксперименты по преподаванию информатики в начальной школе сейчас приняли широкий размах, и школьники первого класса, изучающие информатику, стали обычным явлением для многих современных школ. Тем не менее единой утвержденной государственной программы изучения информатики для начальной школы еще нет (несмотря на включение отдельных учебников в Федеральный комплект Министерства образования РФ). Поэтому так важно обсудить возможные подходы к преподаванию основ информатики, содержание обучения и методику преподавания этой дисциплины в начальной школе. Студенты примут участие в дискуссиях, которые будут инициированы куратором.

Одна из задач курса состоит в том, чтобы дать в руки учителю программные и методические инструменты, научить ими пользоваться, обсудить педагогические и инструментальные проблемы, снабдить учителей дополнительными дидактическими материалами.

Обучаясь с нами, можно сразу использовать полученные материалы и результаты обсуждений для обучения детей, но непосредственная работа детского коллектива на курсе не предусмотрена.

В качестве одной из форм взаимодействия по каждой теме будет предложена разработка новых упражнений для школьников. Эти новые разработки, собранные воедино, становятся достоянием каждого участника обучения.

Студенты, которые не только прочитают распространяемые по сети методические материалы, но и примут активное участие в семинарах, обсуждениях, дискуссиях и напишут самостоятельную курсовую работу, получат удостоверение — документ, который сможет фигурировать в работе комиссий по аттестации учителей.

### II. Компьютерное конструирование (куратор — Ю.А. Первин)

Этот курс ориентирован на руководимые школьными учителями группы детей 4—7-х классов, изучивших основы информатики (желательно в рамках первого года обучения по программе школьного курса “Роботландия”).

Это самый обкатанный и стабильный курс, поскольку его материалы были опробованы и в Роботландской олимпиаде младших школьников (1997), и в двухсеместровых дистанционных курсах (1997—1998), и в Роботландском сетевом университете (1998—1999).

Вместе с тем к новому учебному году планируются некоторые изменения как в содержании обучения, так и в методике проведения занятий (например, увеличение числа перекрестных проверок, показавших свою эффективность как специфический методический инструмент дистанционных форм обучения).

“Компьютерное конструирование” может быть факультативным дополнением к основному курсу раннего обучения информатике или рассматриваться как продолжение изучения информатики на базе практического конструирования компьютерных приложений без использования языка программирования.

В качестве конструкторских сред предлагаются открытые программные системы для изготовления и сборки готового продукта из отдельных информационных фрагментов.

### III. Азы и Буки программирования

**Отделение 1. Азы программирования (куратор — А.А. Дуванов)**

Для детей 5—8-х классов, изучивших основы информатики (исполнитель, алгоритм, первичные инструментальные навыки работы с компьютером).

Инструментальная поддержка: роботландские исполнители Плюсик, Кукарача, Корректор.

Курс является факультативным продолжением основного курса “Роботландия” и ориентирован на детей, проявляющих склонность к программированию.

Открываются новые и порой неожиданные возможности роботландских исполнителей. Рекурсивные приемы становятся обычным инструментом программирования наряду с итерационными методами.

Повышенное внимание уделяется методологии построения алгоритмов и практической работе по тестированию и отладке программ на компьютере.

Азы программирования на базе простых исполнителей и языка, содержащего основные конструкции в минимальном объеме, не покажутся примитивными ни школьникам, ни их учителям. Это введение — первый шаг детей по ступенькам непростой, но такой увлекательной профессии программиста.

В курсе рассматриваются наиболее важные понятия структурного программирования как с точки зрения методов построения алгоритмов, так и с точки зрения практического кодирования и отладки программ.

**Отделение 2. Буки программирования (куратор — Я.Н. Зайдельман)**

**Новинка следующего учебного года! Информация от куратора:**

Что такое БУКИ программирования?

Вас заинтриговало это название? Может быть, оно показалось вам угрюмым и мрачным? Не пугайтесь! Наши БУКИ не имеют ничего общего с бяками. БУКИ — это то, что идет после АЗов, то, что, на наш взгляд, должны изучать начинающие программисты после того, как они освоили основные элементы своего первого языка, радостно вывели на экран “Hello, world!” и научились подсчитывать сумму элементов массива.

Чем мы будем заниматься?

Мы будем учиться решать задачи, в том числе олимпиадного типа.

Будем рассматривать классические задачи и методы их решения. Будем составлять и анализировать алгоритмы, причем составлять не по наитию, а по науке.

Будем классифицировать задачи, чтобы, несмотря на маскировку и внешнюю необычность, узнавать типовые ситуации и применять к ним стандартные приемы.

И конечно же мы будем писать программы — ведь без этого научиться программировать просто невозможно.

Чего у нас не будет?

Совсем не будет производственного программирования. Мы не будем рассматривать проблемы компьютерной графики, вопросы организации интерфейса, проблемы взаимодействия с операционной системой и другие подобные вещи.

Конечно, все это необходимо, без этого невозможно написать профессиональную программу, но мы считаем, что этим нужно заниматься потом — после прочного освоения базовых алгоритмических методов.

На кого рассчитан курс?

Мы предполагаем, что обучаться будут группы школьников под руководством учителя. В принципе возможно индивидуальное участие (как для школьников, так и для учителей), но мы будем ориентироваться в первую очередь на коллективных учеников.

Мы предполагаем, что ученики уже знают элементы какого-нибудь языка программирования (предпочтительные языки — КуМир, Паскаль, годятся также Си, Бейсик, другие процедурные языки), имеют представления о переменных, присваивании, массивах, циклах.

В принципе возможно параллельное изучение указанных основ и нашего курса. В этом случае учитель самостоятельно проводит занятия по избранному им языку, а решение задач проходит в соответствии с нашей программой. При необходимости мы готовы оказать методическую помощь в организации такого параллельного обучения.

Никаких возрастных ограничений мы не ставим, но предполагаем, что курс будет наиболее эффективен для старшеклассников.

### IV. Интернет-технологии

**Отделение 1. Введение в Интернет (куратор — А.А. Дуванов)**

**Для детей 5—8-х классов.**

Предлагаемый курс —это введение в сетевые коммуникации, ориентированное на детей. Обучение построено на базовых понятиях Интернета и общепринятых практических приемах работы в сети.

Наличие электронной почты достаточно для занятий на этом курсе.

Основа практической деятельности на курсе — коммуникативные сетевые обучающие игры, ориентированные на общение детей друг с другом и управление роботландскими сетевыми исполнителями.

Детей нужно учить сетевым технологиям. Это важная общеобразовательная задача современности и прямая обязанность современной школы. Информационная деятельность давно вышла из пеленок пионерских коммуникативных проектов и стала повседневной рутиной. Без сетевых знаний и практических информационных навыков выпускник школы получает неполноценное среднее образование. Информационная деятельность давно уже имеет самостоятельное значение, и вес ее в чемоданчике умений и навыков выпускника должен непрерывно возрастать.

Основные сдерживающие факторы решения этой задачи: отсутствие достаточного финансирования в школах и неадекватная организация сетевой деятельности детей.

Первая проблема решается путем смещения акцента с online-режима просмотра www-страничек (порой бестолкового и даже вредного) на offline-режим электронной почты.

Электронная почта — очень дешевый сервис сети. Разориться на ней практически невозможно. Однако электронной почтой можно делать в сети многое, почти все.

Решение второй проблемы — организацию интересной и полезной сетевой деятельности детей — университет берет на себя. В его арсенале богатый экспериментами текущий учебный год.

Тем не менее опыт этого года показал: работа на курсе будет эффективна лишь тогда, когда учитель сумеет организовать в школе ежедневный (пусть кратковременный) доступ ребенка к своему электронному почтовому ящику. Мы можем дать предварительные советы и рекомендации, но сделать эту работу может только сам учитель.

**Отделение 2. Интернет-программирование (куратор — А.А. Дуванов)**

**Для старшеклассников.**

Создание HTML-документов с элементами программирования на JavaScript. Традиционный набор начинающего web-мастера. Однако основное направление курса связано не с созданием интернетовских страниц, а с проектированием HTML-приложений, пригодных для использования в локальном режиме на школьных уроках в качестве обучающих программ для поддержки различных школьных дисциплин.

Для занятий на этом курсе подключение к сети совсем не обязательно. На каждом рабочем месте необходим какой-нибудь браузер (без выхода в сеть) и текстовый редактор. Если под рукой дополнительно есть графические и звуковые средства, приложения станут существенно лучше, как в плане привлекательности, так и в плане выразительности, но это требование не является обязательным для успешных занятий на курсе.

В большой степени этот курс близок к курсу II "Компьютерное конструирование", но рассчитан на детей постарше, готовых наряду с простым HTML-проектированием окунуться немного и в программирование. Причем совсем не обязательно, чтобы каждый ребенок в группе имел программистские наклонности. Ведь хороший компьютерный продукт, как правило, создается не одним универсалом, а коллективом разнопрофильных специалистов.

**V. Английский с компьютером (куратор — Н.А. Прохорова)**

В новом учебном году на этот курс будут приниматься как индивидуальные, так и коллективные ученики (сейчас обучаются только индивидуальные).

Методические вопросы преподавания языка с использованием компьютера рассматриваются автором апробированного многолетней учительской практикой курса английского языка — Magic Land.

Учителя научатся использовать готовые компьютерные упражнения; создавать новые, оригинальные; запишут в свою творческую копилку новые дидактические материалы, ориентированные как на традиционные приемы преподавания, так и на обучение с использованием компьютера.

Университет работает на основе самофинансирования, поэтому обучение в нем — платное. Ориентировочная цена — 300—400 руб. за семестр с одной группы обучаемых (независимо от числа детей в группе) или с одного индивидуального ученика. Учебный год состоит из двух семестров.

Предварительную заявку можно (и даже желательно) присылать прямо сейчас. Заявка не накладывает на абитуриента никаких финансовых обязательств, но позволит университету получить предварительную информацию об ожидаемом количестве учеников. Сразу после получения заявки абитуриент включается в список рассылки, и ему будут высылаться все объявления, связанные с подготовкой нового учебного года.

**ОБРАЗЕЦ ЗАЯВКИ**

**Предварительная заявка на участие в Роботландском университете 1999/2000 года**

| | |
|---|---|
| Предполагаемый курс (отделение): | Буки программирования |
| Форма обучения: | Коллективный ученик |
| Руководитель (обучаемый): | Иванов Петр Николаевич |
| Профессия: | Учитель информатики средней шк. |
| Стаж работы в школе: | 3 года |
| Директор школы: | Петров Семен Сергеевич |
| Электронный адрес: | ivanov@sch62.nsk.ru |
| Предполагаемое число детей в группе и их возраст (для коллективного ученика): | 20 школьников 5—7-х классов |

Заявки принимаются по адресу: **kurs@robotland.users.botik.ru**

Для получения более подробной информации можно написать по адресу:
**kurs@robotland.botik.ru** (администрация университета),
**auto@robotland.botik.ru** (автоответчик)
или заглянуть на сайт **http://www.botik.ru/~robot**

## Общедоступная библиография университета


1. *Дуванов А.А., Первин Ю.А., Скородумов В.Е.* Телекоммуникационная обучающая олимпиада по информатике. Информатика (приложение к газете "Первое сентября"), № 1, 1997.
2. *Дуванов А.А., Первин Ю.А., Скородумов В.Е.* Дистанционные обучающие Роботландские курсы для школьников старше 10 лет. Информатика (приложение к газете "Первое сентября"), № 22, 1997.
3. *Дуванов А.А., Первин Ю.А., Скородумов В.Е.* Совместное использование компьютерной сети для обучения учителей и школьников. Информатика и образование, № 8, 1997.
4. *Дуванов А.А.* Сетевой роботландский университет. Информатика (приложение к газете "Первое сентября"), № 14, 1998.
5. *Козлова В.А.* Дети — соавторы Роботландии (заметки педагога о Роботландских курсах). Информатика (приложение к газете "Первое сентября"), № 23, 1998.
6. *Дуванов А.А.* Роботландские сетевые курсы. Годовой обучающий марафон завершен! Устарела ли Роботландия? Обучать, пока деревья большие. Есть и будут учителя на наших широких просторах! Информатика (приложение к газете "Первое сентября"), № 23, 1998.
7. *Ломтева Е.В., Кудяева И.Л.* Обнинский опыт работы в рамках сетевого проекта "Роботландские курсы". Информатика и образование, № 8, 1998.
8. *Дуванов А.А.* Приглашение в Роботландский университет. Информатика (приложение к газете "Первое сентября"), № 35, 1998.
9. *Прохорова Н.А.* Подготовка компьютерных учебных заданий в открытых средах конструирования и заочная форма обучения учителей в сетевом Роботландском университете. English (приложение к газете "Первое сентября"), № 36, 1998.
10. *Дуванов А.А.* Интернет для начинающих (фрагмент учебника университета). Информатика (приложение к газете "Первое сентября"), № 36, 40, 44, 46/98; 2, 5, 7, 9, 11, 12/99.

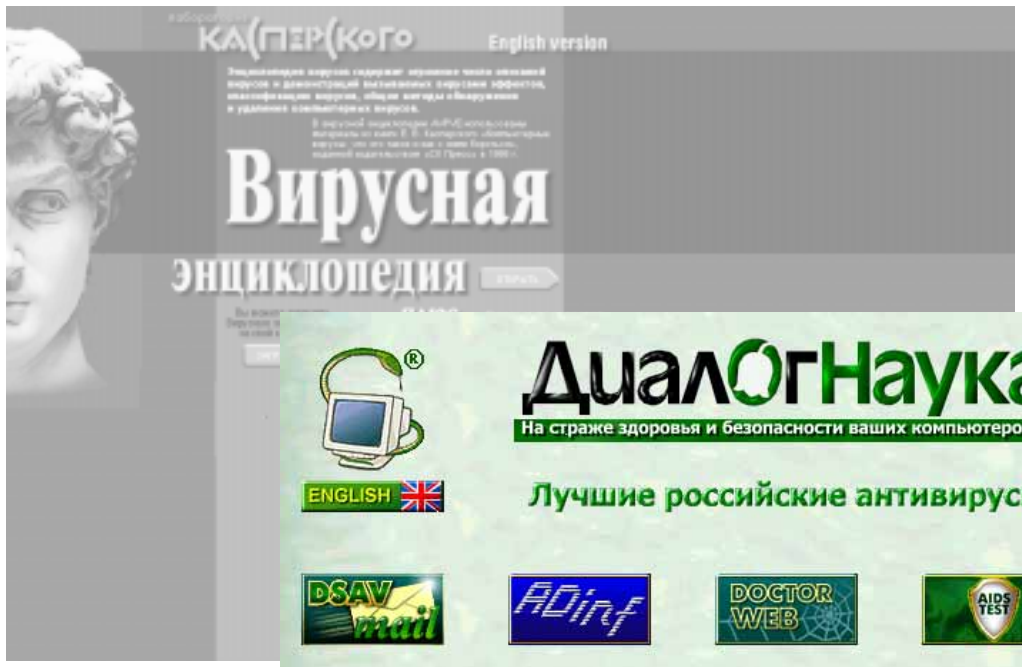


**Руководитель университета—**
**Дуванов А.А.**
duvanov@robotland.users.botik.ru
**Роботландия**
http://www.botik.ru/~robot
**Переславский филиал Вузтелекомцентра**
**(лаборатория телекоммуникаций "Ботик")**
http://www.botik.ru
**Академия повышения квалификации**
**и переподготовки работников образования**
http://www.redline.ru/ripc.html
**Информатика (приложение к газете "Первое сентября")**
http://www.1september.ru

# Узелки на память


При подготовке уроков по истории информатики полезно заглянуть в "Виртуальный музей информатики", расположенный на страницах московской школы № 444 (**www.techno.ru/win/webr**).


В поисках олимпиадных задач не стоит, конечно, ограничиваться тринадцатыми номерами "Информатики". Огромное количество олимпиадных материалов можно найти на страницах, посвященных различным турнирам программистов. Например, на (**www.usu.ru/win/events/1998/contest98**).





Вопреки многочисленным прогнозам появление новых операционных систем вовсе не положило конец компьютерным вирусам. Наоборот, вирусов стало больше, а авторы их изобретательнее. Но часто информация о компьютерных вирусах доходит до нас в форме страшилок: "А вы слышали, что..." С антивирусными программами еще интереснее — все прекрасно знают, что от устаревшей антивирусной программы мало проку, но упорно продолжают использовать полюбившиеся антивирусы *N*-летней давности, которым разве что в "Виртуальном музее информатики" и место. И это при том, что последние, пусть и некоммерческие, версии антивирусов совершенно свободно доступны на WWW-сайтах производителей (например, на двух российских серверах **www.dials.ru** и **www.avp.ru**). Помимо собственно программ, на серверах имеется масса другой полезной и интересной информации, например, "Антивирусная энциклопедия Касперского" (ее можно "скачать" и для просмотра в off-line, если вас не испугает объем архива).








**Гл. редактор**
Е.Б.Докшицкая
**Зам. гл. редактора**
С.Л.Островский
**Редакция:**
Л.Н.Картвелишвили,
Ю.А.Соколинский,
Н.Л.Беленькая,
Н.П.Медведева
**Дизайн и компьютерная верстка:**
Н.И.Пронская
**Корректоры:**
Е.Л.Володина,
С.М.Подберезина





121165, Москва,
Киевская, 24
тел. 249 4896
Отдел рекламы
тел. 249 9870



**ИНДЕКС ПОДПИСКИ**
**для индивидуальных подписчиков 32291**
**для предприятий и организаций 32591**
**комплекта приложений 32744**

**Internet: inf@1september.ru**
**Fidonet: 2:5020/69.32**
**WWW: http://www.1september.ru**

Отпечатано с готовых диапозитивов редакции в ОАО ПО «Пресса-1», 125865 ГСП, Москва, ул. Правды, 24.

Тираж 7000 экз.

Заказ №