№ 10 (299) 8–15 марта 2001

еженедельное приложение к газете «ПЕРВОЕ СЕНТЯБРЯ»

ИНФОРМАТИК

Георг Ом и его закон

175 лет назад, в 1826 году, был открыт закон Ома — центральный закон электродинамики, который наиболее часто используется специалистами в области электросвязи

Главной научной заслугой выдающегося немецкого физика Георга Симона Ома (1787—1854) является открытие основного закона электрической цепи, связывающего сопротивление цепи, электродвижущую силу и силу тока (*закон Ома*). Закон был открыт им экспериментально и сформулирован в 1826 году (в работе "Определение закона, по которому металлы проводят электричество") [1, 2].

Окончание читайте на с. 32

Читайте в номере

Основы программирования

С.М. Окулов,
г. Киров

Продолжение. См. № 42, 43, 44, 45, 46, 47, 48/2000; 6, 7, 8/2001

Занятие 11. Процедуры

План занятия

- структура программы;
- вызов процедур, передача данных;
- глобальные и локальные переменные, формальные и фактические параметры процедур;
- экспериментальная работа с программами перестановки значений переменных *a*, *b*, *c* в порядке возрастания; вычисления значения выражения $y = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + ... + a_2 \cdot x^2 + a_1 \cdot x^1 + a_0$; формирования значений элементов одномерного массива с помощью датчика случайных чисел; поиска элементов, принадлежащих двум массивам одновременно; перестановки частей массива;
- выполнение самостоятельной работы.

Структура программы

Программа на языке Турбо Паскаль состоит из заголовка, раздела описаний и тела программы. Раздел описаний может включать разделы описания меток, констант, типов, переменных, процедур и функций. Последовательность упомянутых разделов описаний может быть произвольной, но естественно, что если вводится переменная нового типа, заданного в разделе описания типов `Type`, то данный раздел `Type` должен предшествовать разделу описания переменных `Var`. Принцип *то, что используется, должно быть описано* справедлив и для раздела описаний.

```
Program имя программы;
 Label <метки>;
 Const <описание констант>;
 Type <описание типов данных>;
 Var <описание переменных>;
 <процедуры и функции>;
 Begin
     <тело программы>
 End.
```

Примечание. В наших программах мы постараемся метки не использовать.

На предыдущих занятиях мы познакомились с тем, как описываются переменные и типы. На этом занятии мы начнем изучать процедуры. Структура процедуры похожа на структуру программы. Отличия выделены жирным шрифтом.

```
Procedure имя процедуры (<параметры>);
 Label <метки>;
 Const <описание констант>;
 Type <описание типов данных>;
 Var <описание переменных>;
     <вложенные процедуры и функции>;
 Begin
   <основное тело процедуры>
 End;
```

Вызов процедур

Чтобы объяснение было наглядным, обратимся к примеру, приведенному на рисунке.

```
Procedure Sum(x,y:Integer; Var z:Integer)
 Begin
   z:=x + y
End;
...
ReadLn(a, b, c);
Sum(a, b, c);
WriteLn(c);
```

Приведен фрагмент текста программы, в котором вызывается процедура вычисления суммы двух целых чисел. Процедура вызывается по имени. При вызове процедуры управление передается на соответствующий участок программного кода. После выполнения процедуры осуществляется возврат на оператор основной программы, следующий за вызовом процедуры. Мы не рассматриваем вопрос, как это осуществляется. Например, как происходит возврат на оператор `WriteLn(c)`. При вызове процедуры в нее могут передаваться данные. В нашем примере в процедуру `Sum` передаются переменные `a, b, c`. Процедура имеет три параметра — `x, y` и `z`. Причем `x, y` описаны без идентификатора `Var`, а `z` — с идентификатором `Var`. Поясним разницу следующей схемой.

```
a → x
b → y
c ↔ z
```

При вызове процедуры `Sum(a,b,c)` из основной программы значение переменной `a` присваивается переменной `x` процедуры `Sum`, а значение переменной `b` — переменной `y`. Иначе обстоит дело с переменными `c` и `z`. Не поясняя пока "кухни" взаимодействия этих переменных, будем считать, что процедура `Sum` "знает" о том, где в памяти компьютера находится переменная `c` и что при изменении переменной `z` необходимо произвести соответствующее изменение переменной `c`. На это указывает идентификатор `Var` в описании параметра `z`.

Дадим ряд стандартных определений. В любой программе все переменные делятся на глобальные и локаль-

ные. *Глобальные переменные* описываются в разделе описаний основной части программы, а *локальные* — в разделах описаний процедур и функций. Но, конечно, дело не только в том, где описываются переменные. Локальные переменные существуют только в течение времени работы процедуры, определяются (создаются) при ее вызове и "исчезают" после завершения работы процедуры (после выполнения оператора End).

Параметры. При описании процедуры указывается список *формальных* параметров. Каждый параметр является локальным, к нему можно обращаться только в пределах данной процедуры (в нашем примере x, y, z — формальные параметры). *Фактические параметры* — это параметры, которые передаются процедуре при обращении к ней (a, b, c — фактические параметры). *Количество и типы формальных и фактических параметров должны совпадать.*

Параметры-значения. В нашем примере фактические параметры a и b передавались "по значению". При таком способе передачи параметров значение фактического параметра становится значением соответствующего формального параметра. Внутри процедуры можно производить любые действия с данным формальным параметром (допустимые для его типа), но эти изменения никак не отражаются на значении фактического параметра, то есть каким он был до вызова процедуры, таким же и останется после завершения ее работы (x, y — формальные параметры-значения).

Параметры-переменные. Это те формальные параметры, перед которыми стоит идентификатор Var. При таком способе передачи параметров в процедуру передается не значение, а адрес фактического параметра (обязательно переменной). Любые операции с формальным параметром выполняются непосредственно над фактическим параметром.

Договоримся о том, как мы будем стараться использовать процедуры, может быть, в ущерб эффективности программ, например, с точки зрения количества переменных и т.д. Каждая процедура должна иметь одну точку входа и одну точку выхода, использование глобальных переменных в процедуре должно быть минимальным, передавать данные в процедуры мы будем лишь посредством параметров. Почему? Мы осваиваем структурную технологию разработки программ, при этом на каждом этапе задача разбивается на ряд подзадач, определяя тем самым некоторое количество отдельных подпрограмм (*подпрограмма* — это повторяющаяся группа операторов, оформленная в виде самостоятельной программной единицы). При этом мы стараемся структурировать задачу не только по управлению, но и по данным, используя при этом весьма ограниченный набор инструментов (параметры-значения, параметры-переменные). Концепция процедур и функций — один из механизмов второго витка развития технологий программирования, а именно, структурного проектирования.

Экспериментальный раздел занятия

1. Составить программу перестановки значений переменных a, b, c в порядке возрастания, т.е. так, чтобы $a \le b \le c$.

```
Program My11_1;
 Var a,b,c:Integer;
 Procedure Swap(Var x,y:Integer);
  Var t:Integer;
  Begin
   t:=x;x:=y;y:=t
  End;
 Begin
  WriteLn('Введите три числа ');
  ReadLn(a,b,c);
  If a>b Then Swap(a,b);
  If b>c Then Swap(b,c);
  If a>c Then Swap(a,c);
  WriteLn(a:5,b:5,c:5);
  ReadLn
 End.
```

Найдите ошибку в этом решении. Требуется исправить имя одной переменной в одной строке программы. Составьте для поиска ошибки полную систему тестов. Измените программу так, чтобы аналогичная задача решалась для четырех переменных.

2. Составить программу вычисления выражения $y = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \ldots + a_2 \cdot x^2 + a_1 \cdot x^1 + a_0$, где все данные — целые числа. Коэффициенты a_n, a_{n-1}, ..., a_1, a_0 являются первыми членами арифметической прогрессии, определяемой первым элементом (q) и разностью между соседними элементами (d). Значения q, d, n, x вводятся с клавиатуры. Например, $q = 2$, $d = 3$, $n = 5$ и $x = 4$. Нам необходимо вычислить выражение $2 \cdot 4^5 + 5 \cdot 4^4 + 8 \cdot 4^3 + 11 \cdot 4^2 + 14 \cdot 4^1 + 17 \cdot 4^0$.

```
Program My11_2;
 Var  q,d,n,x,t,w,s: Integer;
 Procedure Degree(x,y: Integer;
                  Var st: Integer);
  Var i:Integer;
  Begin
   st:=1;
   For i:=1 To y Do st:=st*x
 End;
Begin
 Writeln('Введите исходные данные - четыре
         не очень больших числа');
 Readln(q,d,n,x);
 s:=0;t:=n;
 For i:=1 To n+1 Do Begin
  Degree(x,t,w);s:=s+q*w;
  Dec(t);Inc(q,d)
 End;
 Writeln('Результат ',s);
 Readln
End.
```

Использование процедуры для решения этой задачи несколько искусственно, но наша цель — отработать механизм использования процедур. Составьте таблицу изменения значений переменных q, t, w при работе

программы. Проверьте правильность заполнения таблицы, используя отладчик и режим пошагового исполнения программы. Зафиксируйте значения q, d, x и найдите экспериментальным путем (или модифицируя программу) то значение n, при котором диапазона типа `Integer` уже не хватает для хранения результата.

Примечание. Переменные с именем x в основной программе и в процедуре `Degree` — это разные переменные!

Запишем приведенное выше выражение по-другому: $17+4\cdot(14+4\cdot(11+4\cdot(8+4\cdot(5+4\cdot2))))$. Результат вычислений, естественно, тот же самый. В общем виде запись выглядит следующим образом: $a_0+x\cdot(a_1+x\cdot(a_2+x\cdot(\ldots+x\cdot(a_{n-1}+x\cdot a_n))))$. Приведенная запись называется схемой Горнера. Измените программу так, чтобы вычисление выражения осуществлялось по схеме Горнера.

А что означает запись

$$4\ 2\cdot5+4\cdot8+4\cdot11+4\cdot14+4\cdot17+\ ?$$

Можно ли вычислить это выражение? Оказывается, да. Берем два числа — 4 и 2 — и выполняем с ними операцию, записанную после них, т.е. умножение. Сохраняем результат и продолжаем вычисление. Берем 8 и 5 и выполняем сложение. Потренируйтесь. Напишите несколько примеров и преобразуйте их запись к такому виду. Она называется обратной польской записью, в честь польского математика Яна Лукашевича. Для общего случая в нашей задаче обратная польская запись имеет вид: $a_n\ x\cdot a_{n-1}+\ldots\ a_2\ x\cdot a_1+x\cdot a_0+$. Пусть у нас есть процедура `Part`, выполняющая одно арифметическое действие (умножение или сложение) над двумя числами. Напишите программу для решения нашей задачи с использованием процедуры `Part`.

```
Procedure Part(x,y:Integer; t:Byte;
               Var z:Integer);
 Begin
  If t=1 Then z:=x+y Else z:=x*y
 End;
```

У нас есть три способа вычисления значения одного и того же выражения. Сравните количество операций умножения и сложения, необходимых для получения результата в каждом из случаев. Какой вывод вы можете сделать?

3. Задание значений элементам одномерного массива (с помощью генератора случайных чисел) и вывод результата на экран мы рассмотрели на предыдущем занятии. Оформим эти действия как процедуры.

```
Program My11_3;
 Const n=8;l=-10;h=21;
 Type MyArray=Array[1..n] Of Integer;
 Var A:MyArray;

 Procedure Init(t,v,w:Integer;
                Var X:MyArray);
 Var i:Integer;
 Begin
   Randomize;
   For i:=1 To t Do X[i]:=v+Integer(Random(w))
 End;

 Procedure Print(t:Integer;X:MyArray);
  Var i:Integer;
 Begin
   For i:=1 To t Do Write(X[i]:5);
   WriteLn;
 End;
 Begin
  WriteLn('Формирование значений элементов
           массива A');
  Init(n,l,h,A);
  {Значения n элементов массива A формируются
   из интервала целых чисел от l до h.}
  WriteLn('Вывод');
  Print(n,A)
  {n первых элементов массива A выводятся на
   экран (в данном случае).}
 End.
```

Зачем нам понадобилось оформлять такие простые действия в виде процедур? Было несколько строк программного кода, а сейчас... Пожертвуем кажущейся простотой предыдущей версии программы. С этого момента будем стараться создавать программы так, чтобы основная программа состояла из вызовов процедур и функций. Нарисуйте схему (по подобию тех, что приведены в начале занятия), отражающую взаимосвязь основной программы и процедур как по управлению, так и по данным.

Изменим заголовок процедуры `Print` на

```
Procedure Print(n:Integer;X:Array[1..n] Of Integer)
```

и запустим программу. Результат не заставит себя ждать: `Error 54: OF expected`. Курсор находится на квадратной скобке, т.е. после слова `Array` ожидается `Of`. Пойдем на поводу у системы программирования, изменим описание на

```
Procedure Print(n:Integer;X:Array Of Integer).
```

Результат чуть-чуть изменился, уже знакомая ошибка — `Error 201: Range check error`. Отключим контроль на выход за пределы диапазона — `{$R-}`. Программа заработала, точнее, она выдает результат. Итак, сплошные загадки. Попробуйте дать им разумное объяснение.

4. Даны два одномерных массива. Найти элементы, принадлежащие и тому и другому массивам.

Наша технология написания программ (использование процедур и функций) начинает приносить дивиденды. Процедуру `Init` при решении задачи требуется использовать два раза с различными параметрами, процедуру `Print` — три раза. Сделаем "костяк" программы. Процедуры `Init` и `Print`, естественно, не приводятся, они у нас почти универсальны и берутся из предыдущего задания. Программа (ее "костяк") должна компилироваться. Сохраните ее. Набирать весь текст программы, а затем приступать к отладке — это дурной тон в программировании. Возьмите за правило и всегда его придерживайтесь — "в любой момент времени у нас должна быть компилируемая программа, сохраненная на внешнем носителе". Кроме того, текст любой процедуры и, естественно, основной программы должен помещаться на экране и, конечно, быть читаемым.

```
{$R+}
Program My11_4;
 Const n=10;la=-10;ha=21;
       m=8; lb=-15;hb=31;
 Type MyArray=Array[1..n] Of Integer;
 Var A,B,C:MyArray;
     k:Integer;
Procedure Solve(qx,qy:Integer; Var qz:Integer;
                X,Y:MyArray; Var Z: MyArray);
  Begin
  End;
 Begin
  Init(n,la,ha,A);
  WriteLn('Вывод значений элементов массива A');
  Print(n,A);
  Init(m,lb,hb,B);
  WriteLn('Вывод значений элементов массива B');
  Print(m,B);
  Solve(n,m,k,A,B,C);
  WriteLn('Вывод тех целых чисел, которые
           есть и в A, и в B');
  Print(k,C);
  ReadLn
 End.
```

Начнем уточнять наше решение, а именно, процедуру Solve.

```
Procedure Solve(qx,qy:Integer;
                Var qz:Integer;
                X,Y:MyArray;Var Z: MyArray);
  Var i,j:Integer;
  Begin
   qz:=0;
   For i:=1 To qx Do
    For j:=1 To qy Do
     If X[i]=Y[j] Then Begin
                        Inc(qz);
                        Z[qz]:=X[i]
                        {j:=qy}
                       End
  End;
```

После запуска программы окажется, что при некоторых исходных данных она выдает неправильный результат. Пусть, например, в первом массиве имеется один элемент со значением 10, а во втором — пять таких элементов. Согласно формулировке задачи в ответе 10 должна присутствовать один раз, а она выводится пять раз. Уберем фигурные скобки у оператора j:=qy. Мы "насильно" изменяем управляющую переменную цикла For j:=1 ... Выводится правильный результат. Однако этот прием мы считаем признаком плохого стиля программирования. Изменим внутренний цикл.

```
Procedure Solve(qx,qy:Integer;
                Var qz:Integer;
                X,Y:MyArray;Var Z: MyArray);
  Var i,j:Integer;
  Begin
   qz:=0;
   For i:=1 To qx Do Begin
    j:=1;
    While (j<=qy) And (X[i]<>Y[j]) Do Inc(j);
    If j<=qy Then Begin
                   Inc(qz);
                   Z[qz]:=X[i]
                  End
                    End
  End;
```

5. Даны массив A из n элементов и число m, где $1 < m < n$. Не используя дополнительных массивов, переставить первые m элементов в конец массива, а элементы с $(m + 1)$-го по n-й — в начало, сохраняя порядок элементов.

Подскажем идею решения. Переворачиваем первые m элементов, затем переворачиваем элементы, начиная с $(m + 1)$-го. Затем переворачиваем все элементы массива. Приведем пример.

Пусть $n = 10$, $m = 6$,

массив A:

5 1 −4 3 7 2 −1 9 8 6

2 7 3 −4 1 5 −1 9 8 6

первое переворачивание m элементов

2 7 3 −4 1 5 6 8 9 −1

второе переворачивание элементов с $m + 1$ по n

−1 9 8 6 5 1 −4 3 7 2

третье переворачивание элементов с 1 по n

В тексте решения не приводится реализация процедур Init и Print. Для процедуры Rev рекомендуется выполнить "ручную" трассировку при различных значениях t и l.

```
{$R+}
Program My11_5;
 Const n=10;m=6;
 Type MyArray=Array[1..n] Of Integer;
 Var A:MyArray;
 Procedure Init(Var X:MyArray);
Procedure Print(X:MyArray);
Procedure Swap(Var a,b:Integer);
{Из первого задания.}
Procedure Rev(t,l:Integer;Var X:MyArray);
  Var i:Integer;
  Begin
   For i:=t To t+(l-t) Div 2 Do Swap(X[i],X[l-i+t])
  End;
 Begin
  Init(A); Print(A);
  Rev(1,m,A); Rev(m+1,n,A); Rev(1,n,A);
  Print(A);
  ReadLn
 End.
```

Задания для самостоятельной работы

1. Даны два различных выражения вида:

$$y = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \ldots + a_2 \cdot x^2 + a_1 \cdot x^1 + a_0,$$

$$z = b_n \cdot x^n + b_{n-1} \cdot x^{n-1} + \ldots + b_2 \cdot x^2 + b_1 \cdot x^1 + b_0,$$

— где все коэффициенты — целые числа. Коэффициенты хранятся в массивах A и B. При заданном значении x найти максимальное значение из y и z.

2. Даны два одномерных массива из целых чисел. Найти элементы, которые есть в первом массиве и которых нет во втором.

3. Даны два одномерных массива из целых чисел. Найти элементы, которые входят только в один из массивов.

4. Решить задачу 4 из раздела экспериментальной работы для трех массивов.

5. Даны 3 одномерных массива из целых чисел. Найти элементы, которые есть в первом массиве и которых нет во втором и третьем массивах.

6. Решить задачу 3 для трех одномерных массивов.

7. Даны два одномерных массива из целых чисел разной размерности. Найти среднее арифметическое элементов каждого массива и их сумму. Решить задачу для трех массивов.

8. Даны три одномерных массива из целых чисел одинаковой размерности. Сформировать четвертый массив, каждый элемент которого равен максимальному из соответствующих элементов первых трех массивов.

9. Даны два одномерных массива (A, B) одинаковой размерности n и число m. Поменять местами первые элементы массивов и выполнить перестановку элементов массивов в обратном порядке (задание № 5 раздела экспериментальной работы).

10. Дано четное число n. Проверить для этого числа гипотезу Кристиана Гольдбаха (1742 год). Эта гипотеза (по сегодняшний день не опровергнутая и полностью не доказанная) заключается в том, что каждое четное n, большее двух, представляется в виде суммы двух простых чисел. Ограничим диапазон проверяемых чисел интервалом $2 \le n \le 999$. Фрагмент проверки числа на простоту оформить в виде процедуры.

Примеры:

$6 = 3 + 3$ $\quad$ $12 = 5 + 7$

$30 = 7 + 23$ $\quad$ $308 = 31 + 277$

$992 = 73 + 919$

Исследовать гипотезу К.Гольдбаха для больших значений n.

11. Известны следующие признаки делимости числа n:

- для делимости на 2 необходимо, чтобы последняя цифра числа делилась на 2;
- для делимости на 3 требуется, чтобы сумма цифр числа делилась на 3;
- для делимости на 4 необходимо, чтобы число из последних двух цифр делилось на 4;
- для делимости на 5 необходимо, чтобы последняя цифра числа была 0 или 5;
- для делимости на 8 необходимо, чтобы число из 4 последних цифр делилось на 8;
- для делимости на 9 необходимо, чтобы сумма цифр числа делилась на 9;
- для делимости на 11 необходимо, чтобы разность между суммой цифр, стоящих на четных местах, и суммой цифр, стоящих на нечетных местах, делилась на 11.

Написать процедуры проверки признаков делимости. Проверить их для различных значений n.

Материал для чтения

1. *Некоторые факты из элементарной теории вероятностей.* Всем нам достаточно часто приходится сталкиваться с ситуациями, когда, зная возможные исходы некоторого события (опыта), мы не можем точно предсказать его результат. Например, при бросании монеты — какой стороной она упадет вверх; при бросании игральной кости — какая из шести сторон окажется сверху; при вытаскивании карты из игральной колоды — какая карта будет вытянута и т.д. Главное, что присуще всем этим ситуациям, то, что исходы равноправны. Выпадет орел или решка; выпадет сторона игральной кости с числом от 1 до 6; вытянута одна из 52 карт. Как оценивать исходы, какую меру взять для оценки того, что произойдет то или иное событие. Пусть, например, событие — это выпадение четного числа при бросании игральной кости. Общее количество исходов — 6, число благоприятных исходов — 3. Вытянута карта пиковой масти, общее количество исходов — 52, благоприятных — 13. Понятие меры в теории вероятности вводится следующим образом. Определяется множество элементарных событий S, его элементы считаются равноправными, равновероятными. Любое событие A — подмножество S также состоит из элементарных событий. Отношение $|A|/|S|$, где через $||$ обозначено количество элементов в множестве, называется вероятностью события A и обозначается $P(A)$. Вероятность любого события A заключена между нулем и единицей: $0 \le P(A) \le 1$. Приведем примеры:

- При бросании двух игральных костей вероятность получения 10 и более очков равна $4/36 = 1/9$, так как всего имеется 36 равновероятных исходов и четыре из них благоприятны — (5, 5), (5, 6), (6, 5) и (6, 6).
- Вероятность того, что на первой кости выпадет меньше очков, чем на второй, равна 15/36.
- Стрелок попадает в цель в среднем 92 раза из 100 выстрелов. Вероятность попадания равна 92/100 (0,92).
- На каждую 1000 готовых деталей некоторого предприятия приходится в среднем 16 бракованных. Вероятность изготовления брака для данного производства равна 0,016.
- Первый стрелок попадает в цель с вероятностью 0,8, второй — 0,7. Найти вероятность поражения цели, если оба стрелка стреляют одновременно. Цель считается пораженной при попадании в нее хотя бы одной из двух пуль. Пусть производится 100 двойных выстрелов. Примерно в 80 из них цель будет поражена первым стрелком, а в 20 случаях он промахнется. Второй стрелок из 10 выстрелов примерно 7 раз поражает цель, в 20 выстрелах — 14 раз. Таким образом, при 100 выстрелах цель окажется пораженной примерно 94 раза. Вероятность поражения — 0,94.
- Событие, вероятность которого равна 1, называется достоверным — оно наступает всегда. Монета

упадет гербом или решкой, считаем, что на ребро она встать не может. Событие, вероятность которого равна 0, называется невозможным — оно не наступает никогда. В нашем примере бросания игральной кости (и других) есть то, что называют элементарным событием — выпадение стороны кости с определенным числом. Остальные события составляются из элементарных событий, например, выпадение стороны кости с четным числом. События A и B называют *несовместимыми*, если появление одного из них исключает появление другого. При бросании монеты орел и решка не могут выпасть одновременно. Если A и B — несовместимые события, то $P(A + B) = P(A) + P(B)$. Утверждение обобщается на n несовместимых событий. Говорят, что несколько событий $A_1, A_2, ..., A_n$ образуют *полную группу*, если вероятность того, что произойдет одно из них, является достоверным событием. Примеры:

- Опыт — бросание двух монет. События "два орла", "две решки" и "один герб, одна решка" образуют полную группу и являются несовместимыми.
- Опыт — бросание игральной кости. События "1 или 2 очка", "2 или 3 очка", "3 или 4 очка", "4 или 5 очков" и "5 или 6 очков" образуют полную группу, но не являются несовместимыми.
- Опыт — вынимание одной карты из колоды в 36 карт. События — вынимание туза, короля, дамы, валета, десятки, девятки, восьмерки, семерки и шестерки — образуют полную группу и являются несовместимыми.
- Опыт — передача в одинаковых условиях трех сообщений равной длины. События "искажено первое сообщение", "искажено второе сообщение" и "искажено третье сообщение" не образуют полную группу и не являются несовместимыми.

2. *Информация.* Итак, об информатике говорят как о науке, изучающей обработку информации с помощью ЭВМ. Мы называем ЭВМ универсальной машиной по обработке информации. Обсудим кратко термин *информация*, понимая при этом, что строгого определения нам не дать, ибо понятие информации является первичным, неопределяемым. Слово *информация* используется в двух значениях — качественном и количественном. С одной стороны, мы понимаем под информацией конкретные сведения о чем-либо, представленные в виде речи, текста, изображения, цифровых данных, графиков, таблиц и т.п. С другой стороны — ее численную меру, то есть выраженное в битах количество абстрактной информации. Остановимся на втором аспекте. Одним из параметров измерения информации является ее объем: количество бит, байт и т.д. Есть последовательность из единиц и нулей. Длину этой последовательности в одних книгах называют количеством информации, в других — объемом информации. Другой способ оценки количества информации основан на вероятностном подходе. Вводится мера неопределенности, мера нашего незнания чего-либо. Устранение неопределенности достигается за счет получения информации. Если есть "лапоть" для измерения неопределенности, то он может быть использован для наших целей — определения количества информации. Приведем традиционное рассмотрение этой схемы на примере игры "Бар — Кохба".

Игра "Бар — Кохба". Один из игроков что-то загадывает, а второй должен отгадать загаданное, задавая вопросы, которые предполагали бы только ответы типа "да", "нет". Предположим, что в классе 16 учеников и учитель загадал номер по журналу одного из учеников, они по какой-то причине пронумерованы числами от 0 до 15. В начальный момент времени у нас полное незнание того, какой номер загадан.

Первая серия наших вопросов.

1-й вопрос. Находится ли загаданный номер в интервале от 8 до 15? Ответ — да (1).

2-й вопрос. Находится ли загаданный номер в интервале от 12 до 15? Ответ — нет (0).

3-й вопрос. Находится ли загаданный номер в интервале от 11 до 12? Ответ — да (1).

4-й вопрос. Загадан номер 11? Ответ — нет (0).

Вторая схема формулировок вопросов.

1-й вопрос. Находится ли загаданный номер в интервале от 0 до 3? Ответ — нет.

2-й вопрос. Находится ли загаданный номер в интервале от 12 до 15? Ответ — нет.

3-й вопрос. Находится ли загаданный номер в интервале от 4 до 7? Ответ — нет.

4-й вопрос. Находится ли загаданный номер в интервале от 8 до 11? Ответ — да.

5-й вопрос. Находится ли загаданный номер в интервале от 10 до 11? Ответ — да.

6-й вопрос. Загадан номер 11? Ответ — нет.

Чем отличаются серии вопросов? Мера нашего незнания в том и в другом случаях одинакова, то есть получено одно и то же количество информации. Примем за единицу измерения информации (1 бит) количество информации, содержащееся в ответе, при условии, что ответы *равновероятны*. В первом случае каждый ответ содержит один бит информации, ибо ответы "да" и "нет" были равновероятны. Во втором случае — нет, в среднем один ответ содержал 4/6 бита информации. Всего получено 4 бита информации. Неопределенность полностью устранена, мы знаем загаданный номер. Пусть есть N равновероятных событий и выделено одно из событий (t). Для того чтобы определить t, необходимо получить количество информации I, равное $\log_2 N$. Эта формула для вычисления количества информации предложена американским инженером Р.Хартли в 1928 году. Если события не равновероятные, то количество информации вычисляется по формуле американского математика К.Шеннона:

$$I = -(p_1 \log_2 p_1 + p_2 \log_2 p_2 + ... + p_N \log_2 p_N),$$

— где p_i — вероятности событий. Из формулы К.Шеннона получается формула Р.Хартли при $p_i = 1/N$ для всех значений i. Действительно,

$$I = -(1/N \cdot \log_2(1/N) + \ldots + 1/N \cdot \log_2(1/N)) = \log_2 N.$$

В скобках первой серии вопросов указаны цифры 0 или 1. Если выписать ответы в виде двоичного числа 1010_2 и перевести его в десятичную систему счисления, то получим число 10 — номер загаданного ученика! Итак, каждая двоичная цифра несет в себе один бит информации, отсюда и название единицы измерения — бит (*binary digit* — двоичная цифра). В игре "Бар — Кохба" вопросы задаются последовательно, на основании ответов учителя. А можно ли задать сразу четыре вопроса, а затем получить четыре ответа учителя и определить номер загаданного ученика? Оказывается, да. Вопросы формулируются так: "Вы загадали номер, двоичная запись которого содержит единицу в первом разряде?" Обратите внимание на то, что и в этом случае ответы "да" и "нет" равновероятны.

Предположим, что учитель одновременно загадал номера двух учеников (t_1 и t_2) в разных классах. Количество учеников в классах — N_1 и N_2. Необходимо отгадать пару (x_1, x_2), принадлежащую множеству всех пар — $(N_1 \times N_2)$. Согласно формуле Хартли, необходимо задать $\log_2(N_1 \cdot N_2)$ вопросов, или получить $\log_2(N_1 \cdot N_2)$ бит информации. С другой стороны, номера учеников можно отгадывать независимо, для отгадывания t_1 требуется задать $\log_2 N_1$ вопросов, для отгадывания t_2 — $\log_2 N_2$ вопросов. Следовательно, общее число вопросов равно $\log_2 N_1 + \log_2 N_2$. Итак, мы получили $\log_2(N_1 \cdot N_2) = \log_2 N_1 + \log_2 N_2$, что совпадает с хорошо известным свойством логарифмической функции. Это закон аддитивности информации.

Примеры

Имеется 27 монет: 26 настоящих и одна фальшивая, она легче настоящих. Сколько взвешиваний необходимо произвести, чтобы определить фальшивую монету?

Фальшивой может оказаться любая из 27 монет, следовательно, по формуле Хартли количество недостающей информации равно $\log_2 27$ битов. Любое взвешивание имеет три исхода и может дать нам только $\log_2 3$ битов информации. Если мы производим X взвешиваний, то они дадут $X \cdot \log_2 3$ битов информации.

Итак, $X \cdot \log_2 3 \geq \log_2 27 = \log_2 3^3 = 3 \cdot \log_2 3$.

Следовательно, $X \geq 3$. На самом деле достаточно ровно 3 взвешиваний: первое по 9 монет, второе по 3 монеты из найденной группы и, наконец, по одной монете из найденной группы по 3 монеты.

Чтобы найти элемент множества, состоящего из 7 элементов, необходимо задать три вопроса, а чтобы найти элемент множества, состоящего из 9 элементов, — 4 вопроса, то есть по формуле Хартли получить $\log_2 7 + \log_2 9$ битов информации. Но если необходимо отгадать пару элементов из этих множеств, то требуется получить $\log_2(7 \cdot 9)$ битов информации, а это меньше 6 вопросов. Противоречия нет —

$$\log_2(7 \cdot 9) = \log_2 7 + \log_2 9 = 5{,}97728 < 6.$$

Как понимать формулу Хартли, если $\log_2 N$ не является целым числом? Предположим, что мы выполняем отгадывание не один, а k раз, то есть строим последовательность из k неизвестных элементов — $(x_1, x_2, \ldots, x_k)$. Таких последовательностей N^k. Очевидно, что для числа вопросов (s_k) в этом случае выполняются следующие неравенства:

$$\log_2 N^k < s_k < \log_2 N^k + 1$$

или

$$\log_2 N < s_k/k < \log_2 N + 1/k.$$

Число s_k/k показывает, сколько вопросов в среднем необходимо для того, чтобы отгадать один элемент множества, содержащего N элементов. Выбирая значение k достаточно большим, величину $1/k$ можно сделать сколь угодно малой. Итак, в том случае, когда N не совпадает со степенью двойки, число вопросов, которое в среднем необходимо задать для отгадывания одного элемента множества мощности N, будет отличаться от $\log_2 N$ на сколь угодно малую величину.

Продолжение следует

Выпуск 7

Зимние вечера

Информатика для начинающих

Ю.А. Первин,
г. Переславль-Залесский

Содержание выпусков
1 (№ 1/2001), 2 (№ 5/2001), 3 (№ 6/2001), 4 (№ 7/2001), 5 (№ 8/2001), 6 (№ 9/2001)

ВЕЧЕР ТРИНАДЦАТЫЙ. ПРОМЕЖУТОЧНЫЙ ФИНИШ

Дедушка Фёрстов в молодости увлекался велосипедным спортом. Поэтому его внуки Тим и Дина не очень удивились, что в тот вечер, когда по телевизору шла трансляция с чемпионата мира по велосипедному спорту, дедушка сидел около телевизора, а не, как всегда, у компьютера. А когда они решились все же спросить у дедушки, когда им сегодня можно будет прийти в его кабинет, к компьютеру, он умоляюще попросил их:

— Подождите, пожалуйста, всего одну минутку. Сейчас гонщики разыграют промежуточный финиш...

— Ну, раз промежуточный финиш, — сказал Тим, — то тогда, конечно...

— А ты знаешь, что такое "промежуточный финиш"? — шепотом поинтересовалась сестра, когда они вышли в коридор.

— Нет, конечно. Но раз дедушку это так волнует и он об этом говорит со всей серьезностью, значит, это что-то важное.

Когда дедушка вошел в кабинет, где его уже ожидали дети, первый вопрос задала Дина:

— Дедушка, что такое "промежуточный финиш"?

На этот вопрос дедушка ответил с улыбкой:

— Это то, что нас с вами ждет сегодня!

— Как? — удивилась Дина. — Ты хочешь устроить нам велогонки? Это нечестно. Ты же знаешь, что Тим лучше меня ездит на велосипеде.

— Нет, конечно, — успокоил дедушка. — На велосипедах вы гоняться не будете. Но термин "промежуточный финиш" из велоспорта к сегодняшнему нашему компьютерному занятию очень подходит.

Когда для велосипедистов устраивают гонки с промежуточным финишем, то на дистанции (обычно очень длинной) отмечают несколько пунктов, которые называют промежуточными финишами. На промежуточном финише гонщик не останавливается, а продолжает мчаться вперед. Однако тем, кто первым пройдет промежуточный финиш, начисляют дополнительные очки, которые учитываются при подведении итогов всего соревнования. Таким образом, гонщик старается не только вырваться вперед в конце гонки, но очень активно ведет себя на каждом промежуточном финише. Гонка становится очень интересной, увлекательной и для гонщиков, и для зрителей.

— А к нам это какое имеет отношение? — продолжала недоумевать Дина.

— Сегодня мы не заканчиваем наши занятия, и наши зимние вечера будут продолжаться долго. Но, согласитесь, пора уже подвести некоторые итоги. Ведь вы многому научились: знаете, что такое алгоритм и исполнитель, умеете исправлять ошибки клавиатурного набора и управлять ветвящимися сказками, можете найти закономерности в множестве элементов и разделить

множество на два подмножества по характеристическому признаку. Поэтому сегодня я ничего нового вам не расскажу, только предложу новые задачи. Это и будет промежуточный финиш, после которого мы помчимся дальше.

— Другими словами, это будет контрольная работа по пройденной теме, как любит говорить наша учительница, — вспомнил Тим родную школу.

— Можно, конечно, сказать и так, — не стал сильно возражать дедушка Фёрстов, — но я предпочитаю назвать наше нынешнее занятие турниром юных информатиков. Вам нравится такое название?

— Нравится! — дружно согласились ребята. — Давай нам турнирные задачи!

И дедушка показал Дине и Тиму записанный на экране компьютера список задач. Перед каждой задачей были написаны два числа.

— Первое число — это, понятно, номер задачи. А второе — баллы, которые начисляются за решение каждой задачи, — пояснил дедушка. — Как видите, баллы могут быть разными, потому что задачи различаются по сложности — есть простые, а есть и потруднее. Хотя вам и не обязательно решать все задачи, но, конечно, надо постараться набрать побольше баллов.

Для каждой задачи указано наибольшее число баллов, которые можно получить за решение задачи. Но можно получить и меньше. Например, вы правильно решили задачу, но не объяснили, как это решение получено. Или правильно определили лишний элемент в множестве, но не указали характеристический признак, по которому элементы собраны в множество. Во всех таких случаях окончательный балл будет снижен.

Решать турнирные задачи вы можете в любом порядке. Все равно итоговой оценкой станет сумма баллов за все решенные задачи.

Дина задала дедушке самый главный вопрос:

— А решать задачи надо вместе?

— Да, — подтвердил дедушка. — Ведь вы — одна команда.

— А ты — наш капитан?

Дедушка согласился:

— Да. В Роботландском университете я считаюсь руководителем вашей команды.

— С кем же мы тогда будем соревноваться в турнире? — удивился Тим.

— С такими же командами, как ваша, с такими же третьеклассниками, как и вы. Только живут и учатся они в разных городах и селах страны — либо, как вы, братья и сестры из одной семьи, либо соседи из одного дома, либо группа ребят из одной школы. Закончив решение, вы пошлете его в Роботландский университет, из которого нам прислали эти задачи. Там же вашей команде, то есть нам с вами (как и каждой другой команде), поставят оценку и определят победителей турнира.

— Турнир — это здорово! — с энтузиазмом отреагировал Тим на необычное предложение дедушки, но вдруг сменил свой восторженный тон. — Только...

— Что "только"?

Вместо ответа Тим показал на окно. На улице шел мокрый снег, да и сильный ветер заставлял прохожих поднимать воротники и кутаться в шарфы. Погода была неважная.

— Не хочется идти на почту в такую противную погоду, — объяснил Тим свое сменившееся настроение.

— Никуда не придется ходить, — успокоил его дедушка. — Не забывайте, ребята, что наш компьютер подключен к информационной сети Интернет. Так же, как и компьютеры других команд, наших соперников по турниру. А это значит, что ваши письма с решениями задач можно (и даже нужно) отправить не обычной почтой, а электронной — прямо с компьютера.

Дедушкины слова об электронной почте еще больше раззадорили ребят. Они, не долго размышляя, согласились с условиями турнира и... помчались к промежуточному финишу.

Турнирные задачи дедушки Фёрстова

1. (*4 балла*) Задана последовательность чисел:

1, 2, 3, 5, 8, 13, 21, ...

Какое число должно быть следующим?

2. (*3 балла*) Заданы несколько чисел:

23, 13, 17, 15, 31, 47, 5

Какое среди них лишнее?

3. (*3 балла*) У Кати Пушковой хороший почерк. Поэтому учительница попросила Катю перед уроком написать на доске заранее подготовленный список. Катя выполнила задание. Даже немножко перестаралась — она написала один город лишний:

Владимир, Суздаль, Ярославль, Кострома, Переславль-Залесский, Иркутск, Сергиев Посад.

На какую тему хотела поговорить учительница на уроке? Что лишнего написала Катя?

4. (*4 балла*) Как надо продолжить цепочку чисел?

1, 2, 4, 5, 7, 8, 10, 11 ...

5. (*3 балла*) Какая буква продолжает цепочку?

Б, В, Г, Д, Ж, З, ...

6. (*3 балла*) Страничка разделена на две части. Вверху написаны слова

волк, бегемот, крокодил, лиса, лось,

а внизу —

карась, жираф, петух, лошадь, медведь.

В нижнюю часть надо перенести одно слово из верхней. Какое это слово?

7. (*4 балла*) Страничка разделена на две части. Вверху написаны слова

домовой, строитель, лесник, засоня, равенство,

а внизу —

рассвет, подножка, загон, постройка.

В нижнюю часть надо перенести одно слово из верхней. Какое это слово?

8. (*5 баллов*) Написать программу для исполнителя Плюсик, вычисляющую следующее арифметическое выражение:

$$1111 \cdot 7 - 17 \cdot (874 - 6 \cdot (37 + 9 \cdot 11) + 345).$$

9. (*8 баллов*) Написать программу для исполнителя "Плюсик", вычисляющую следующее арифметическое выражение:

$$1004 - \frac{\dfrac{823 - 11 \cdot 7}{373} \cdot \dfrac{34 \cdot 15 - 45 \cdot 3 + 685}{31 \cdot 11 + 27 \cdot 7}}{822 - 67 \cdot 3 + 45 \cdot (417 - 46 \cdot 7) - 3896}$$

10. (*9 баллов*) Прочитайте отрывок из книги А.Дуванова и Ю.Первина "Необыкновенные приключения Пети Кука в Роботландии".

В нем рассказывается, как поезд, которым управлял робот по имени "Машинист", тронулся в путь.

"...Но тут произошло непредвиденное: навстречу на своем вороном коне скакал "Конюх".

— Стой! — осадил коня около локомотива запыхавшийся и взволнованный "Конюх". — Вперед нельзя! Там мост обвалился!

Выбравшись из вагонов, роботы собрались вокруг "Конюха".

— Придется назад, в объезд! Вот беда! Это сколько же времени мы будем тащиться задним ходом?

Но "Машинист", успокаивая друзей, возразил:

— А зачем задним? Здесь рядом, в двухстах метрах, есть тупик. Там мы развернемся. Я ведь помню алгоритм разворота. И мы поедем обратно, но поедем быстро — передним ходом.

Пассажиры расселись по местам и успокоились. Но вот подъехали к тупику, и волнения начались снова.

— Как же он сможет тут развернуться? Ведь в тупике есть место либо только для локомотива, либо для одного вагона. А ведь в нашем составе целых два вагона да паровоз.

Но Петя уже знал, как поступить:

— Давайте составим алгоритм разворота. Как вы думаете, какие команды можно использовать в этом алгоритме?

"Конюх" заметил:

— Надо спросить "Машиниста". Ведь ему можно давать только те команды, которые он понимает. А кто это знает лучше его самого?

Кук взглянул на "Машиниста", и он не только назвал понятные ему команды одну за другой, но и объяснил, как они выполняются:

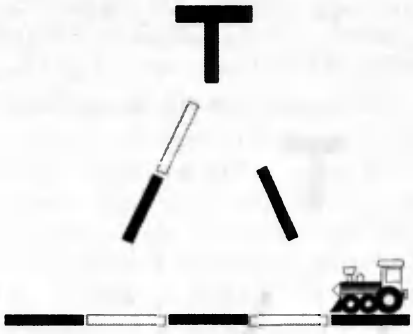
ВПЕРЕД — по этой команде локомотив движется вперед, или до ближайшей разведенной стрелки, или до свободного участка пути, когда впереди нет стрелок, или до тех пор, пока не помешает тупик, или до стоящего на пути вагона (локомотив может двигать лишь сцепленный с ним вагон);

НАЗАД — движение задним ходом по тем же правилам;

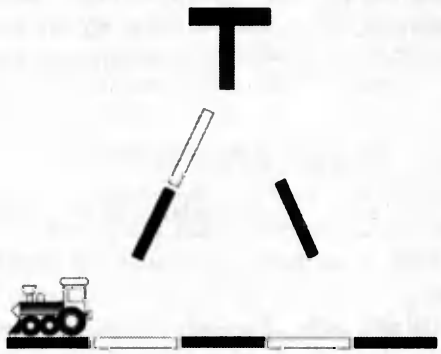
ОТЦЕПИ — отцепляется один последний вагон, или, точнее, последний из прицепленных вагонов;

ПРИЦЕПИ — прицепляется один ближайший неприцепленный вагон;

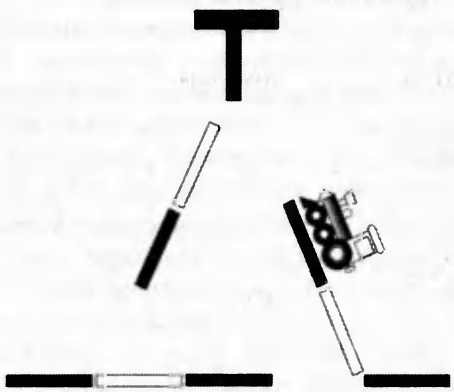
СТРЕЛКА — перевод стрелки в противоположное состояние.



Последняя команда вызвала самое бурное обсуждение у всех, кто окружил "Машиниста" и внимательно его слушал. Дело в том, что она изменяет направление дальнейшего движения поезда. Если локомотив стоял так, как показано на рисунке слева, и в этом положении получил команду ВПЕРЕД, то в результате ее выполнения получится ситуация, которая изображена на следующем рисунке: локомотив проехал по тому пути, на который его привела открытая стрелка. Поэтому в тупик ему заезжать совсем не пришлось.



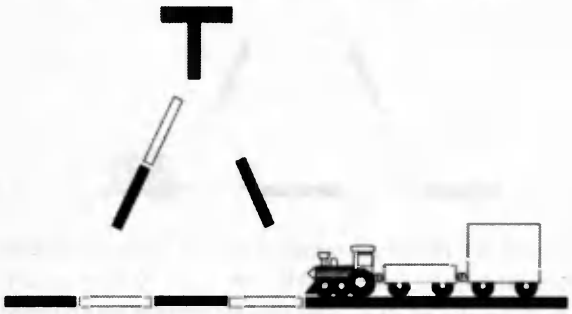
Если же в исходной ситуации, справа от тупика, локомотив получает команду СТРЕЛКА, а сразу после этого — команду ВПЕРЕД, то в результате возникает положение, которое изображено на следующем рисунке.

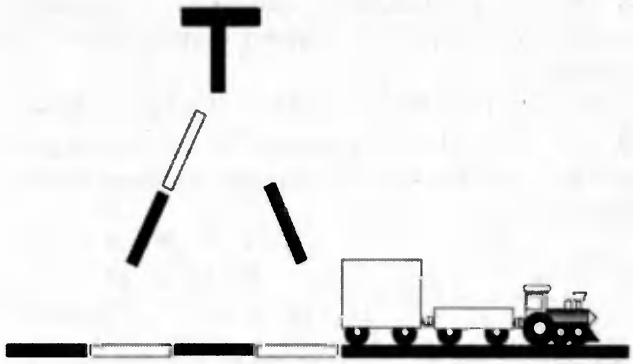


Переводить железнодорожную стрелку можно, когда локомотив находится в любом положении: слева от тупика, справа от него и в тупике. Но при этом должно обязательно соблюдаться такое условие: с помощью команды СТРЕЛКА можно выполнить перевод только

одной-единственной стрелки — той, что находится ближе других к локомотиву на пути следования поезда.

Петя старательно начертил два рисунка: сначала начальное положение состава (обратите внимание на то, в каком состоянии находятся стрелки на Петином рисунке), а затем — конечное его положение.





После этого остается только записать алгоритм!"

Именно это и требуется сделать в последнем задании турнира дедушки Фёрстова.

Книга для учителя

ЗАНЯТИЕ 13

КОЛЛЕКТИВНОЕ СОРЕВНОВАНИЕ

Тема занятия: турнир "Компьютерная смекалка".

Цель занятия: закрепить знания по завершенным темам-разделам — алгоритмы, исполнители, классификаторы.

План

1. О роли турнирной формы занятий.
2. Рассказ о предстоящем соревновании.
3. Общее представление задач учащимся.
4. Обсуждение организационных положений соревнования.

Программное обеспечение: пакет "Зимние вечера" — набор программ и исполнителей, освоенных детьми в первом семестре курса.

Комментарии

1. О роли турнирной формы занятий

Турнирная (конкурсная, соревновательная) форма проведения учебных занятий — не новинка в современном школьном учебном процессе. Такая форма урока наиболее актуальна в начальной школе, поскольку в младшем школьном возрасте игровая форма деятельности еще сохраняет свою значимость для детей. Вместе с тем в младших классах соревновательные формы занятий представляются скорее исключением, чем правилом. Это связано с тем, что турнирная форма занятия отличается от традиционного урока высоким психологическим и эмоциональным подъемом всего коллектива учащихся, а потому требует от учителя:

— серьезной организационной подготовки с тщательным планированием временных ресурсов; если турнир включает компьютеризованные фрагменты урока, подготовка требует детального планирования доступных ресурсов компьютеров и периферийной техники;

— эффективного, насыщенного проведения занятия, позволяющего учащимся раскрыть свои умения и навыки;

— предельно внимательного анализа результатов турнира, учитывающего повышенную заинтересованность учащихся в оценке, особенно в командных соревнованиях; требуется высокая объективность оценок; это последнее требование инициировало появление такой новой формы контроля, как перекрестные проверки.

Турниры и конкурсы стали неотъемлемой частью в ряде курсов раннего обучения информатике. Как правило, уроки-соревнования являются кульминацией темы и непосредственно предшествуют ее завершению. К этому моменту многие из инструментальных навыков у детей сформированы, это позволяет сосредоточить внимание и на содержании турнирных задач, делая турнир более увлекательным.

Характерно, что учебно-контролирующая функция соревновательной формы занятия оказалась весьма эффективной в дистанционном обучении детей. И если в базовом образовании эта форма до сих пор рассматривается как экзотическая и в методическом плане все же несколько отклоняющаяся от основных дидактических линий курса, в дистанционное обучение формы турнирные формы вписываются органично и естественно. Четко организованное соревнование компенсирует те ограничения дистанционного, непосредственно не ощущаемого общения со сверстниками, которые не позволяют увидеть лицо соперника-товарища и услышать его голос.

Структура курсов и понятие коллективного ученика (команды) дают прекрасную возможность комбинировать индивидуальные (личные) и командные соревнования. Готовя команду к соревнованию, учитель получает возможность обсудить со школьниками организацию деятельности в коллективном проекте. Навыки такой деятельности составляют неотъемлемую

часть образования молодого человека информационного общества. Участвуя в коллективном проекте, дети формируют высокое чувство ответственности за общее дело. Воспитательный потенциал таких соревнований велик.

Не менее важен учет индивидуальных качеств ученика в командной работе над коллективным проектом. Учитель может (и должен) учесть склонности одних детей к рисованию, умение логично и правильно излагать свои мысли в текстовой форме у других, видеть общую структуру системы в множестве элементов — у третьих. (Может быть, дедушка Фёрстов располагает меньшими возможностями распределить между детьми турнирные задачи в соответствии с априорной подготовленностью своих учеников (в связи с их малочисленностью), однако у большинства руководителей команд такие возможности есть, и этот замечательный шанс целесообразно использовать как в педагогическом, воспитательном, так и технологическом плане.)

При той общей характеристике конкурсов-турниров, которая приводится здесь, следует отметить, что нынешний, первый турнир — самый простой. Во всяком случае, в организационном плане. В нем практически не задействованы возможности индивидуального зачета: конкурс — полностью командный.

Одна из особенностей первого турнира состоит в том, что ему предшествует тренировка в решении типовых задач. В книге для школьника целый "вечер" отведен задачам дедушки Фёрстова. Такие (но не те же самые!) задачи будут предложены (присланы по сети из дистанционного обучающего центра) командам за несколько дней до начала турнира. К этому времени многие задачи дедушки Фёрстова учитель уже посмотрит с детьми, и команда будет готова к проведению соревнования.

2. Рассказ о предстоящем соревновании

Вся предшествующая работа на курсе, по существу, не требовала знакомства с однокурсниками из других команд. Сейчас учитель рассказывает о наличии многих коллективов сверстников в разных уголках страны, имеющих возможность быстро и содержательно обмениваться информацией между собой благодаря информационной сети, к которой подключены компьютеры. По этой сети турнирные задачи доставляются командам, по той же сети дети отправляют свои решения турнирных задач в центр — "судейскую коллегию", методическую комиссию Роботландского университета.

Нынешний турнир — первый. Впереди еще несколько таких турниров. Отсюда готовое толкование названия занятия — "Промежуточный финиш".

Цель — получить максимальное суммарное количество баллов. При этом не обязательно решать все задачи. Можно выбрать посильные и понравившиеся. Впрочем, не следует исходно планировать отступление (это не в характере ребенка), лучше прийти к такому решению при нескольких неудачных подходах к трудной задаче. Однако обратить внимание на то, что за трудные задачи можно получить большее количество баллов, необходимо.

3. Общее представление задач учащимся

Учитель показывает детям все турнирные задачи. Вообще говоря, для первого знакомства с задачей достаточно зачитать их по сообщению куратора. Однако учитель имеет возможность подготовить к моменту представления рисунки, схемы, демонстрации, которые он сочтет необходимыми. Обязательный элемент знакомства с задачей — максимальная оценка, которая может быть получена за решение задачи.

Важно объяснить, что в большинстве задач оценивается не столько запись результата, а, скорее, его объяснение. В задачах-"классификаторах" это особенно важно, поскольку в простом выделении лишнего элемента множества или в выделении одного элемента, который "с нами", еще не видно главного, чем руководствуются дети в выборе такого решения, — характеристического признака элемента в исследуемом множестве. За верное, но необоснованное решение оценка снижается.

4. Обсуждение организационных положений соревнования

В зависимости от состава группы (количества учащихся, их подготовленности, их индивидуальных качеств), а также в зависимости в первую очередь от пожеланий школьников могут быть предложены разные организационные схемы: от закрепления за каждой задачей одного или нескольких учеников до последовательного решения всех задач всей командой. Выбор такой организационной схемы полностью ложится на учителя — руководителя команды. Однако чрезвычайно важны открытость обсуждения, участие детей в таком общем разговоре. (Поскольку этот фрагмент занятия должен учитывать специфику условий каждой отдельной команды, становится интересным и важным последующий коллективный его анализ (здесь речь идет о коллективе руководителей команд, участвующих в курсе "Зимние вечера. Информатика для начинающих"). Поэтому возникает совершенно естественная просьба к руководителям команд: в своих письмах куратору курса, или в письмах однокурсникам по списку рассылки, или в тезисах выступления на семинаре или конференции поделиться своим мнением о проведенной вами работе по организации турнира в ваших конкретных условиях.)

Продолжение следует

Графика в LOGO

Поурочная методическая разработка по информатике для класса УКНЦ. 7—8-е классы

Г.В. Луканина,
Москва

Продолжение. Начало в № 5/2001

УРОКИ 16—18

Суперпроцедура и субпроцедура

Суперпроцедурой (или вызывающей процедурой) называют процедуру, которая вызывает другую процедуру.

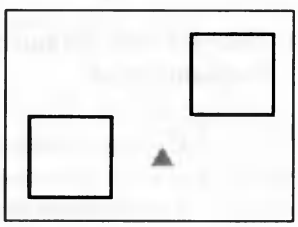
Субпроцедурой (или вызываемой процедурой) называют процедуру, которая используется другой процедурой в качестве команды. В субпроцедуре нежелательно использовать команды CS и HOME.

Если требуется нарисовать одну и ту же фигуру несколько раз, но в разных местах экрана, то сначала создают субпроцедуру, рисующую эту фигуру, а затем — суперпроцедуру, которая осуществляет невидимое перемещение Черепашки в нужную точку экрана и вызывает субпроцедуру.

Пример

Субпроцедура, рисующая один квадрат:

```
TO KV
REPEAT 4 [FD 80 RT 90]
END
```



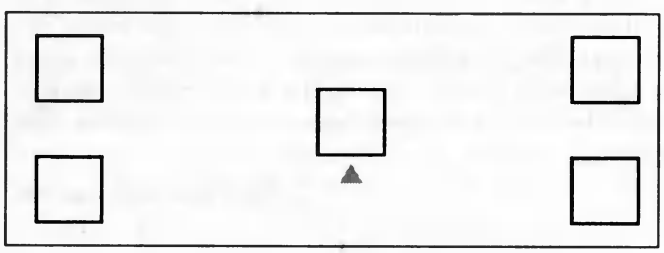
Суперпроцедура:

```
TO SUPER
CS
PU BK 50 LT 90 FD 280 RT 90 PD
KV HOME
PU FD 40 RT 90 FD 200 LT 90 PD
KV HOME
END
```

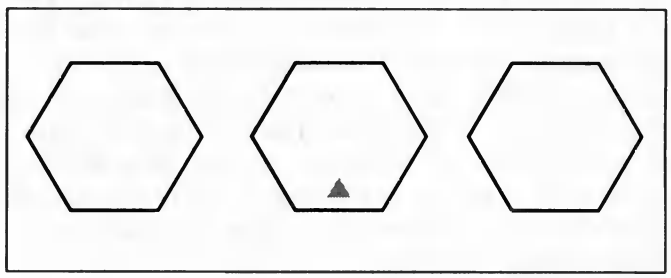
Чтобы теперь увидеть весь рисунок на экране, надо вызвать суперпроцедуру, то есть SUPER ↵.

Примечание. Вызвать процедуру — это значит в суперпроцедуре написать имя субпроцедуры, и тогда Черепашка выполнит все команды, находящиеся в субпроцедуре (то есть все команды между TO и END).

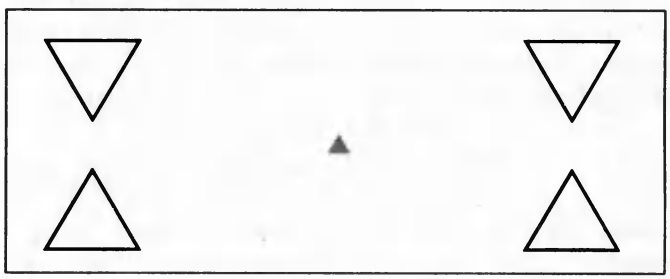
Упражнение 28



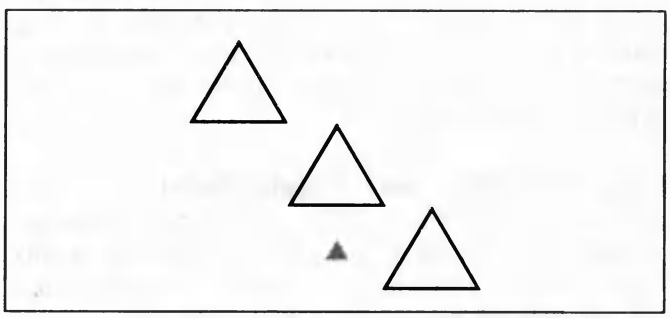
Упражнение 29



Упражнение 30

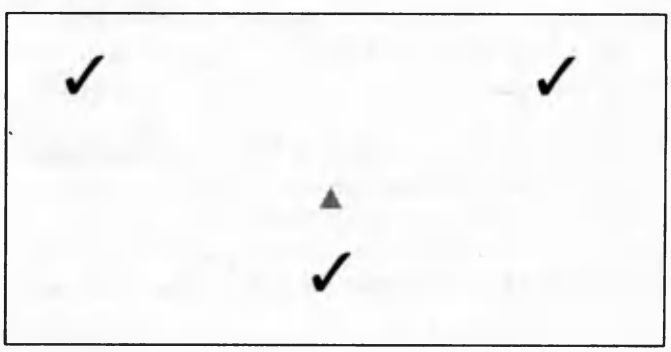


Упражнение 31



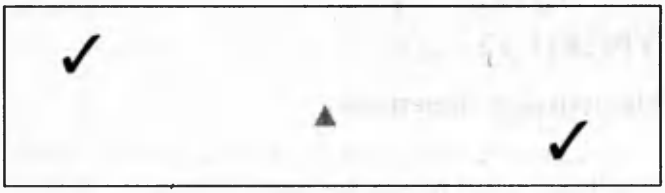
Упражнение 32

Нарисовать восьмиугольник в указанных местах экрана.



Упражнение 33

Нарисовать двенадцатиугольник в указанных местах экрана.



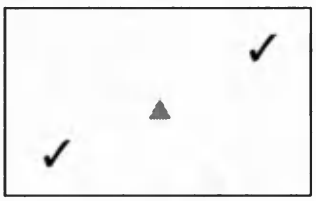
Урок 19

Контрольная работа № 3

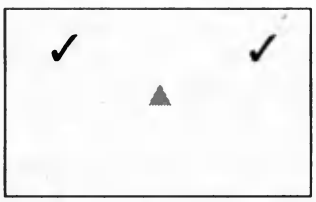
Цель урока: проверка освоения учащимися тем "Построение правильного *N*-угольника" (команда REPEAT) и "Субпроцедура и суперпроцедура".

Контрольную работу можно проводить по индивидуальным карточкам. Предлагаю следующие 6 вариантов, в каждом из которых необходимо нарисовать *N*-угольник в указанных местах экрана.

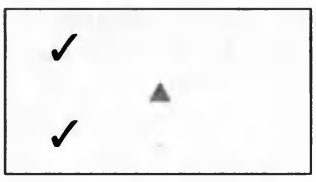
1. Нарисовать пятиугольник в заданных местах экрана.



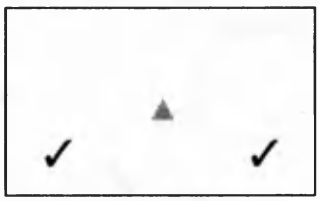
2. Нарисовать четырнадцатиугольник в заданных местах экрана.



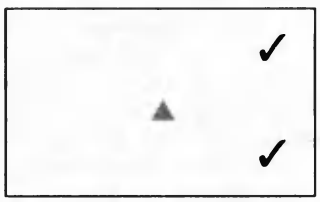
3. Нарисовать девятиугольник в заданных местах экрана.



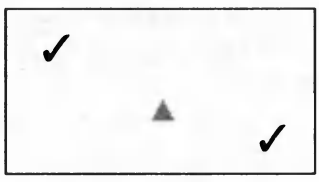
4. Нарисовать семиугольник в заданных местах экрана.



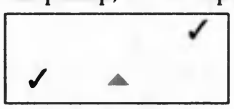
5. Нарисовать десятиугольник в заданных местах экрана.



6. Нарисовать тринадцатиугольник в заданных местах экрана.



Примечание. Эти задания — на оценку "3". На оценку "4" предложить ученику нарисовать еще один *N*-угольник, например, в 1-м варианте было



станет

А на оценку "5" ученик выполняет два варианта.

УРОК 20

Рисунок типа "цветок"

Чтобы создать рисунок типа "цветок", надо:

1) написать субпроцедуру, рисующую один "лепесток" (это может быть окружность, ромб или любой *N*-угольник);
2) написать суперпроцедуру, в которой весь "цветок" рисуется с помощью команды REPEAT:

```
REPEAT N [<имя> RT 360/N]
```

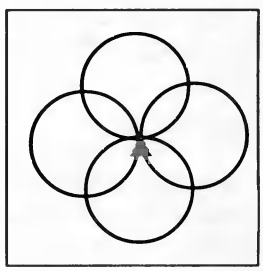
или

```
REPEAT N [<имя> LT 360/N].
```

Здесь *N* — это число "лепестков", *<имя>* — имя субпроцедуры, рисующей один "лепесток".

Пример

"Цветок" из 4 окружностей.



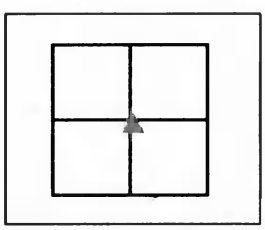
```
TO OKR
REPEAT 120 [FD 1.5 RT 3]
END

TO CVETOK
CS
REPEAT 4 [OKR RT 90]
END
```

Упражнение 34

а) Нарисовать "цветок" из 6 окружностей,
б) Нарисовать "цветок" из 10 окружностей,
в) Нарисовать "цветок" из 15 окружностей.

Упражнение 35

а) Нарисовать "цветок" из 4 четырехугольников,



б) Нарисовать "цветок" из 8 четырехугольников,
в) Нарисовать "цветок" из 24 четырехугольников.

Упражнение 36

а) Нарисовать "цветок" из 7 пятиугольников,
б) Нарисовать "цветок" из 10 пятиугольников,
в) Нарисовать "цветок" из 12 пятиугольников.

УРОК 21

Копирование процедур

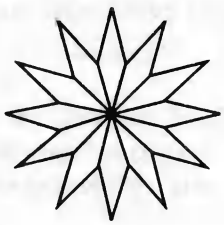
Если новая процедура должна немного отличаться от уже написанной, то можно войти в режим редактирования старой процедуры, то есть набрать

TO *<имя старой процедуры>* ↵.

заменить имя процедуры на новое, внести нужные изменения в текст процедуры и выйти из режима редактирования (посредством комбинации клавиш УПР + E). При этом старая процедура сохранится, и еще будет создана новая процедура.

Пример

Это "цветок" из 12 ромбиков.



```
TO ROMB
REPEAT 2 [FD 40 RT 30 FD 40 RT 150]
END

TO ROMB12
CS
PU FD 32 PD
REPEAT 12 [ROMB RT 360/12]
END
```

Примечание. Вторая строка суперпроцедуры, то есть PU FD 32 PD, выполняет невидимое перемещение Черепашки в центр экрана.

Допустим, надо нарисовать "цветок" из 6 ромбиков. Для этого достаточно отредактировать процедуру ROMB12, заменить в ней число 12 на 6 (в двух местах), и мы получим новую процедуру (процедура ROMB12 также сохранится в памяти):

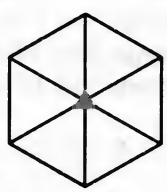
```
TO ROMB6
CS
PU FD 32 PD
REPEAT 6 [ROMB RT 360/6]
END
```

Упражнение 37

На основе вышеизложенного написать процедуру для рисования "цветка" из 20 ромбиков.

Упражнение 38

а) Нарисовать "цветок" из 5 треугольников,



б) Нарисовать "цветок" из 6 треугольников.

Упражнение 39

а) Нарисовать "цветок" из 6 шестиугольников,
б) Нарисовать "цветок" из 12 шестиугольников,
в) Нарисовать "цветок" из 18 шестиугольников.

Примечание. Можно предложить детям поэкспериментировать — менять число лепестков (которое не должно превышать 50, так как при этом получается почти полностью закрашенный круг).

УРОКИ 22—23

Невидимая Черепашка

Есть две ситуации, когда может пригодиться невидимая Черепашка (то есть нет изображения самой Черепашки на экране, но она выполняет все команды):

1) изображение Черепашки на экране мешает рисунку;
2) необходимо ускорить процесс вычерчивания рисунка ("невидимая" Черепашка рисует быстрее).

Познакомимся с двумя новыми командами.

HT ↵ — Черепашка, спрячься!

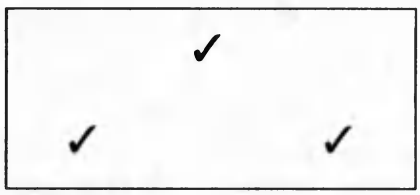
ST ↵ — Черепашка, покажись!

Команда HT действует до тех пор, пока не встретится команда ST.

Упражнение 40

1) Нарисовать "цветок" из 5 треугольников (см. упражнение 38а).

2) Нарисовать 3 таких "цветка" в указанных местах экрана.



Примечание. Для выполнения этого задания надо написать 3 процедуры:

TRE — субпроцедура, рисующая один треугольник;

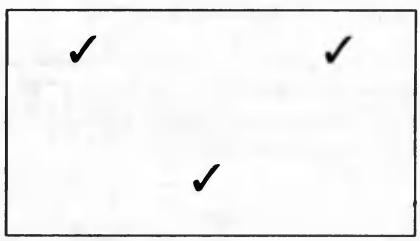
CVET — субпроцедура, рисующая "цветок" из 5 треугольников. Эта процедура одновременно будет и суперпроцедурой по отношению к процедуре TRE, так как процедура CVET вызывает процедуру TRE;

SUPER — суперпроцедура, выполняющая невидимое перемещение Черепашки в нужные точки экрана и вызов процедуры CVET.

Упражнение 41

1) Нарисовать "цветок" из 6 восьмиугольников,

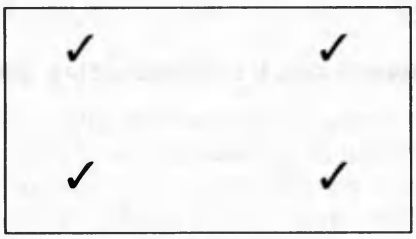
2) Нарисовать 3 таких "цветка" в указанных местах экрана.



Надо написать 3 процедуры: 8UG, CVET и SUPER.

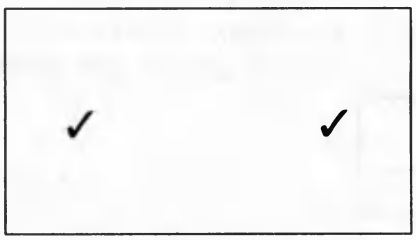
Упражнение 42

1) Нарисовать "цветок" из 9 семиугольников,
2) Нарисовать 4 таких "цветка" в указанных местах экрана.



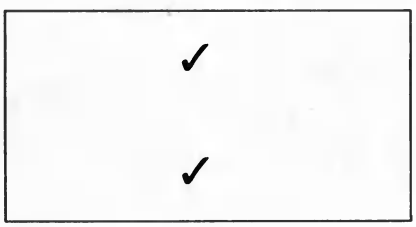
Упражнение 43

1) Нарисовать "цветок" из 12 окружностей,
2) Нарисовать 2 таких "цветка" в указанных местах экрана.



Упражнение 44

1) Нарисовать "цветок" из 8 десятиугольников,
2) Нарисовать 2 таких "цветка" в указанных местах экрана.



УРОК 24

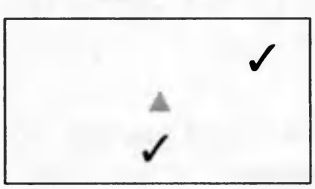
Контрольная работа № 4

Цель урока: проверка освоения учащимися темы "Рисунок типа "цветок".

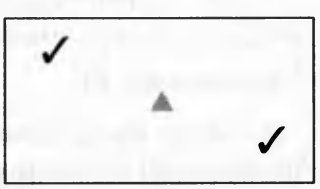
Задание: во всех вариантах требуемый "цветок" надо нарисовать в указанных местах экрана.

Варианты заданий

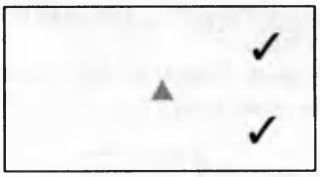
1. "Цветок" из 8 пятнадцатиугольников.



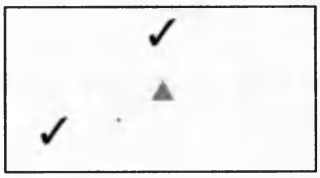
2. "Цветок" из 13 семиугольников.



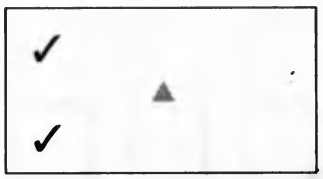
3. "Цветок" из 12 девятиугольников.



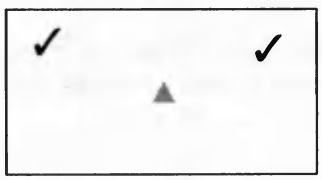
4. "Цветок" из 10 шестиугольников.



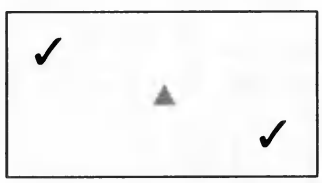
5. "Цветок" из 11 восьмиугольников.



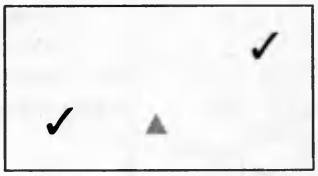
6. "Цветок" из 9 пятиугольников.



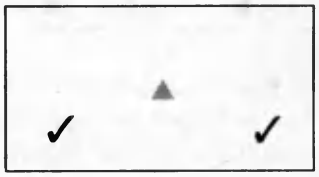
7. "Цветок" из 6 семиугольников.



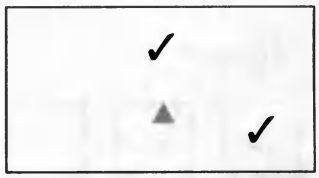
8. "Цветок" из 12 треугольников.



9. "Цветок" из 48 четырехугольников.

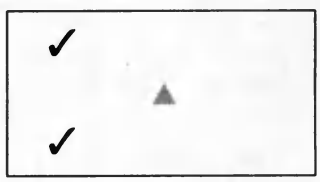


10. "Цветок" из 8 двенадцатиугольников.



11. "Цветок" из 5 окружностей.

12. "Цветок" из 7 десятиугольников.



Примечание. В качестве дополнительного задания на оценку "5" можно предложить:

— изменить количество лепестков,

— изменить сам лепесток, то есть *N*-угольник. Например, если в задании требуется нарисовать "цветок" из 10-угольников, то можно попросить ученика нарисовать "цветок" из 7-угольников, а количество лепестков не менять.

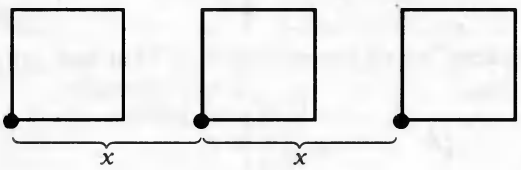
УРОКИ 25—26

Горизонтальный ряд из одинаковых фигур

Для того чтобы равномерно (на одинаковом расстоянии друг от друга) расположить на экране *N* одинаковых фигур по горизонтали, надо:

1) написать субпроцедуру, рисующую одну фигуру;
2) написать суперпроцедуру, содержащую команду:

REPEAT *N* [<*имя*> PU RT 90 FD *X* LT 90 PD],
где *N* — число фигур в ряду,
<*имя*> — имя субпроцедуры, рисующей одну фигуру,
X — расстояние между контрольными точками двух соседних фигур.



Контрольная точка — это точка, от которой Черепашка начинает рисовать фигуру.

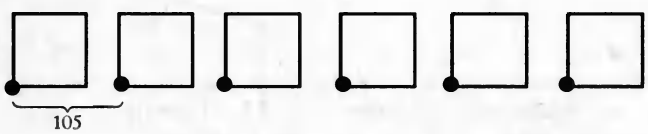
X вычисляют по формуле:

$$X \approx 640 : N.$$

Округляем *X* до целого в меньшую сторону, например,

$$X = 31{,}7 \approx 30.$$

Пример



$$X = 640 : 6 = 106{,}66... \approx 105$$

Отсюда следует, что сторона квадрата должна быть меньше 105, в нашем примере сторона квадрата равна 85.

```
TO KV
REPEAT 4 [FD 85 RT 90]
END

TO SUPER
CS HT
PU LT 90 FD 290 RT 90 PD
REPEAT 6 [KV PU RT 90 FD 105 LT 90 PD]
END
```

Упражнение 45

Построить горизонтальный ряд из 10 четырехугольников.

Упражнение 46

Построить горизонтальный ряд из 8 шестиугольников.

Упражнение 47

Построить горизонтальный ряд из 7 окружностей.

Упражнение 48

Построить горизонтальный ряд из пяти "цветков", состоящих из 6 окружностей.

В данном упражнении надо написать 3 процедуры:
OKR — субпроцедура, рисующая одну окружность,
CVET — субпроцедура, рисующая один цветок из 6 окружностей,
SUPER — суперпроцедура, вызывающая процедуру CVET в команде REPEAT 5 [CVET PU...].

Упражнение 49

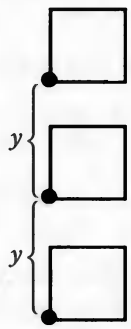
Построить горизонтальный ряд из четырех "цветков", состоящих из 8 пятиугольников.

УРОК 27

Вертикальный ряд из одинаковых фигур

Для того чтобы равномерно расположить на экране по вертикали *N* одинаковых фигур, надо:

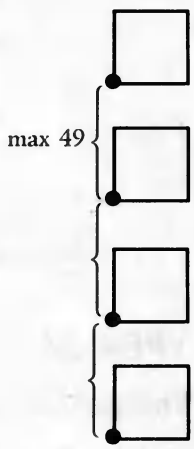
1) написать субпроцедуру, рисующую одну фигуру;
2) написать суперпроцедуру, в которой сначала Черепашку опустить в низ экрана, а затем воспользоваться командой REPEAT в виде:
REPEAT *N* [<*имя*> PU FD *Y* PD],
где *N* — число фигур в ряду,
<*имя*> — имя субпроцедуры, рисующей одну фигуру,
Y — расстояние между контрольными точками двух соседних фигур.



$$Y \approx 196 : N$$

Пример

$Y = 196 : 4 = 49$




```
TO KV
REPEAT 4 [FD 30 RT 90]
END

TO SUPER
CS HT
PU BK 60 PD
REPEAT 4 [KV PU FD 45 PD]
END
```

Упражнение 50

Построить вертикальный ряд из 7 четырехугольников.

Упражнение 51

Построить вертикальный ряд из 5 шестиугольников.

Упражнение 52

Построить вертикальный ряд из трех "цветков", состоящих из 7 пятиугольников.

Упражнение 53

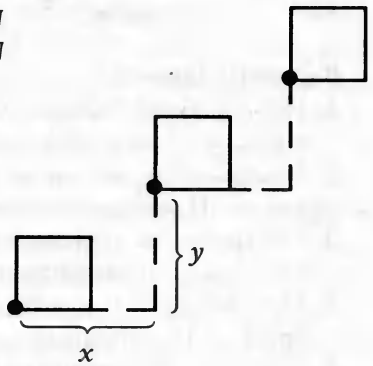
Построить вертикальный ряд из двух "цветков", состоящих из 10 восьмиугольников.

УРОКИ 28—29

Диагональный ряд из одинаковых фигур

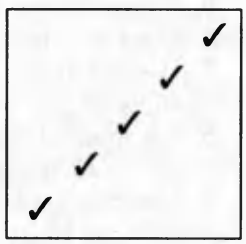
При построении диагонального ряда требуется перемещать Черепашку по вертикали и горизонтали, поэтому надо вычислять и X, и Y:

$X \approx 640 : N$
$Y \approx 196 : N$



Мы будем рассматривать диагонали двух видов.

I (или основная) диагональ

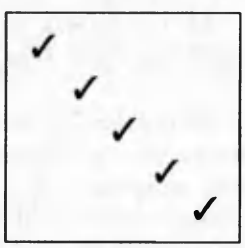


В этом случае нужно поднять перо и переместить Черепашку в нижний левый угол, а затем воспользоваться командой REPEAT в виде:

```
REPEAT N [<имя> PU FD Y RT 90 FD X LT 90 PD]
```

вертикальный ряд — горизонтальный ряд

II диагональ



А в этом случае надо поднять перо и переместить Черепашку в верхний левый угол, команда REPEAT будет иметь вид:

```
REPEAT N [<имя> PU BK Y RT 90 FD X LT 90 PD]
```

вертикальный ряд, который рисуется сверху вниз — горизонтальный ряд

Упражнение 54

Построить 10 четырехугольников по I диагонали.

Упражнение 55

Построить 8 шестиугольников по II диагонали.

Упражнение 56

Построить 6 окружностей по I диагонали.

Упражнение 57

Построить 9 окружностей по II диагонали.

Упражнение 58

Построить 4 "цветка" по I диагонали, состоящих из 8 девятиугольников.

Упражнение 59

Нарисовать 3 "цветка" по II диагонали, состоящих из 6 семиугольников.

Упражнение 60

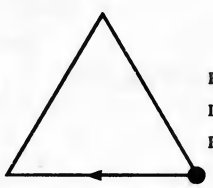
Нарисовать 5 "цветков" по I диагонали, состоящих из 16 четырехугольников.

УРОК 30

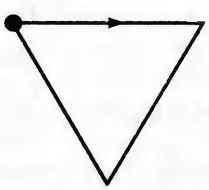
Ряды из треугольников и пятиугольников

Если мы будем рисовать ряды из треугольников или пятиугольников и при этом Черепашка находится в своем обычном исходном положении, то есть смотрит вверх, то эти фигуры будут иметь вид:

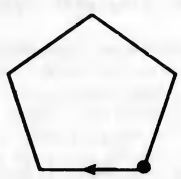
Чтобы эти ряды выглядели симпатичнее, надо, чтобы перед командой REPEAT Черепашка смотрела не вверх, как обычно, а налево или направо в зависимости от ситуации:

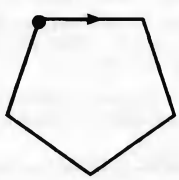
1) 

контрольная точка в нижнем правом углу фигуры, и Черепашка смотрит влево,

2) 

контрольная точка в верхнем левом углу фигуры, и Черепашка смотрит вправо.

3) 

4) 

Пример

Нарисовать "заборчик" из 8 треугольников.

$$X = 640 : 8 = 80$$

Сторона треугольника равна *X*, так как между треугольниками нет пустого места.

```
TO TRE
REPEAT 3 [FD 80 RT 120]
END
```

Чтобы получить треугольник в таком виде, необходимо, чтобы Черепашка первоначально смотрела влево, поэтому ряд будем рисовать справа налево, для чего Черепашку надо сначала невидимо переместить направо:

```
TO SUPER
CS HT
PU RT 90 FD 320 LT 180 PD
REPEAT 8 [TRE PU FD 80 PD]
END
```

Упражнение 61

Нарисовать "перевернутый" заборчик из 10 треугольников.

Упражнение 62

Нарисовать ряд из 4 таких пятиугольников по вертикали.

Упражнение 63

Нарисовать ряд из 6 таких треугольников по вертикали.

Упражнение 64

Нарисовать ряд из 9 таких треугольников по I диагонали.

Упражнение 65

Нарисовать ряд из 7 таких пятиугольников по II диагонали.

УРОК 31

Контрольная работа № 5

Цель урока: проверка освоения учащимися темы "Построение фигур по горизонтальным, вертикальным и диагональным рядам".

Варианты заданий

1. Построить по I диагонали ряд из 10 "цветков", состоящих из 15 восьмиугольников.

2. Построить по вертикали ряд из 4 "цветков", состоящий из 10 двенадцатиугольников.

3. Построить по горизонтали ряд из 5 "цветков", состоящих из 12 девятиугольников.

4. Построить по II диагонали ряд из 6 "цветков", состоящих из 16 пятиугольников.

5. Построить по вертикали ряд из 3 "цветков", состоящих из 11 семиугольников.

6. Построить по горизонтали ряд из 7 "цветков", состоящих из 9 шестиугольников.

7. Построить по I диагонали ряд из 8 "цветков", состоящих из 14 семиугольников.

8. Построить по вертикали ряд из 5 "цветков", состоящих из 22 четырехугольников.

9. Построить по II диагонали ряд из 7 "цветков", состоящих из 18 одиннадцатиугольников.

10. Построить по горизонтали ряд из 9 "цветков", состоящих из 20 пятиугольников.

11. Построить по I диагонали ряд из 6 "цветков", состоящих из 8 десятиугольников.

12. Построить по II диагонали ряд из 5 "цветков", состоящих из 13 двенадцатиугольников.

На оценку "5" можно предложить учащимся внести изменения в процедуры так, чтобы получился другой ряд, например, из горизонтального — I диагональ, из вертикального — II диагональ, из II диагонали — горизонтальный ряд и т.д.

Продолжение следует

Опыт изучения машины Поста

С.Л. Жаров,
г. Ростов-на-Дону

Введение

Среди значительного числа исполнителей алгоритмов, предлагаемых авторами различных учебников, незаслуженно забыт интересный исполнитель — "машина Поста". Являясь "младшей сестрой" другой известной алгоритмической машины — машины Тьюринга, машина Поста более подходит для знакомства школьников с основами алгоритмизации и программирования.

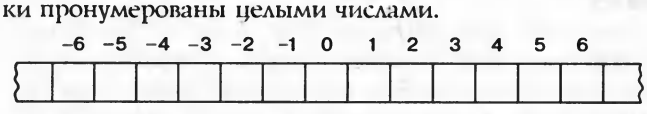
Учителя информатики, работающие со старой вычислительной техникой (например, с УКНЦ), испытывают определенные затруднения в работе, связанные с нехваткой методических и программных разработок для морально устаревшей техники. Предлагаемая работа представляет собой опыт использования исполнителя "машина Поста" при обучении информатике школьников 8-го класса. В ходе изучения темы "Алгоритмизация" ребята составляли программы для машины Поста. Для автоматической проверки программ использовалась компьютерная реализация этого исполнителя, написанная автором для компьютеров УКНЦ.

Краткая историческая справка и описание машины

В 30-х годах теперь уже прошлого столетия американский математик Эмиль Пост, работая над уточнением понятия "алгоритм", предложил достаточно простую схему для описания алгоритмов. Этот серьезный математический результат был получен практически одновременно с аналогичными результатами А.Тьюринга (машина Тьюринга) и Маркова (нормальные алгоритмы Маркова). Уточнения понятия "алгоритм", предложенные Постом и Тьюрингом, не потеряли своего значения и в наше время.

Машина Поста не есть реально существующее, сделанное кем-то устройство. Она, так же, как и машина Тьюринга, представляет собой мысленную конструкцию, существующую лишь в нашем воображении.

Машина Поста состоит из *ленты* и *каретки* (называемой также *считывающей* и *записывающей головкой*). Лента бесконечна и разделена на ячейки. Все ячейки пронумерованы целыми числами.



В ячейки бесконечной ленты можно записывать всего два знака — 0 и 1. Информация о том, какие ячейки пусты (содержат 0), а какие содержат 1 (или, как писал Пост, метку), образует *состояние ленты*. Иными словами, состояние ленты — это распределение меток по ячейкам. Состояние ленты меняется в процессе работы машины.

Вдоль ленты может передвигаться каретка. За один *шаг* каретка может сдвинуться на одну ячейку вправо или влево, может поставить или стереть метку в той ячейке, напротив которой она стоит, а также распознать, стоит или нет метка в *текущей* ячейке.

Состояние машины Поста слагается из состояния ленты и указания номера ячейки, под которой стоит каретка, то есть номера текущей ячейки.

Работа машины Поста состоит в том, что каретка передвигается вдоль ленты и печатает или стирает сетки. Эта работа происходит по инструкции, которая называется программой.

Программа машины Поста состоит из команд. Данный исполнитель — машина Поста — может выполнять только 6 действий (команд):

— сдвиг каретки вправо;
— сдвиг каретки влево;
— поставить метку в текущей ячейке;
— стереть метку в текущей ячейке;
— команда условного перехода — передача управления на ту или иную команду в зависимости от того, есть ли метка в текущей ячейке;
— команда останова.

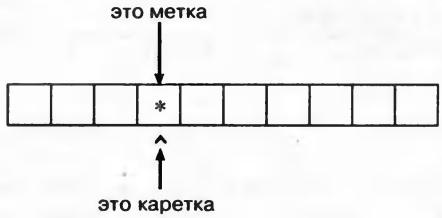
Подробное описание машины Поста содержится в [1].

Компьютерная реализация

В описываемой статье используется машина Поста, которая реализована в виде компьютерной программы. Относительно описанной в [1] машины Поста были произведены некоторые изменения в системе команд и их обозначении. Поэтому, строго говоря, был реализован другой исполнитель, а не тот, который описан в [1], и ниже дается описание именно этого исполнителя.

Что же представляет собой эта конкретная реализация машины Поста?

Имеется лента, состоящая из 39 ячеек (в варианте [1] лента бесконечна). Вдоль ленты вправо и влево может продвигаться каретка. В каждой ячейке может находиться метка. Каретка может обозревать текущую ячейку (ту, под которой она находится) и определять: есть там метка или нет. Кроме того, машина Поста может ставить или стирать метку в текущей ячейке.



Система команд состоит из 6 "стандартных" команд и двух дополнительных.

Стандартные команды:

1. → — шаг вправо,
2. ← — шаг влево,
3. ! — ставь метку,
4. # — стирай метку,
5. ? — ветвление (переход по условию),
6. stop — остановка машины.

Формат записи всех команд, кроме ветвления и остановки, следующий:

$$i \text{ <команда> } j,$$

— где i — номер строки в программе, содержащей команду (номер команды), j — номер строки, к которой происходит переход после выполнения команды.

Команда ветвления записывается следующим образом:

$$i \ ? \ n_1 \mid n_2,$$

— где i — номер команды, а n_1 и n_2 — номера строк в программе, на которые будет передано управление. Если обозреваемая ячейка пуста, то происходит переход к команде с номером n_1, иначе — к команде с номером n_2.

Команда остановки записывается так: i stop.

Дополнительные команды

При реализации машины Поста были введены две служебные команды, необходимые для задания начальных условий.

7. metka $i_1, i_2, i_3, ..., i_n$,

— где $i_1, i_2, i_3, ..., i_n$ — номера ячеек, в которые изначально будут поставлены метки,

8. start i,

— где i — номер ячейки, которую будет обозревать каретка в начале работы программы.

Тематическое планирование

№ п/п	Тема	Количество часов
1	Что такое машина Поста? Как она работает? Простейшие программы для машины Поста	1
2	Практическая работа с программной реализацией машины Поста	2
3	Запись чисел, прибавление единицы в простейшем случае	1
4	Прибавление единицы в более сложных случаях	2
5	Сложение чисел. Различные случаи	4
6	Контрольная работа	1
7*	Копирование массива меток	2
8*	Реализация вычитания, умножения и деления чисел	4
Итого:		17

Первые пять тем можно изучать со всеми учащимися, а 7-ю и 8-ю темы — только с наиболее сильными учениками или на факультативных занятиях. Далее приводятся краткие конспекты этих тем.

Тема 1

На первом уроке я даю историческую справку о машине Поста, ввожу систему команд, знакомлю с правилами записи программы (они изложены в [1]). Обязательно обращаю внимание учащихся на три возможных варианта работы машины.

1. Безрезультативная остановка. Встретилась попытка поставить метку в занятую ячейку или стереть метку в свободной ячейке.
2. Результативная остановка. Достигнута команда stop.
3. Зацикливание машины. Машина работает бесконечно.

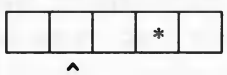
Рассмотрение этих вариантов можно провести на следующем примере.

Программа:

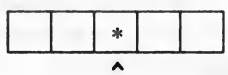
```
1 ! 4
2 stop
4 → 5
5 ? 1 | 2
```

Исходные состояния ленты:

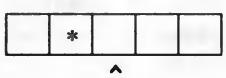
а)



б)



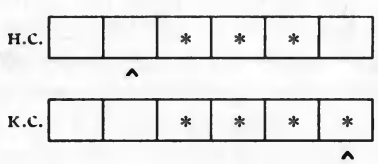
в)



В случае **а** выполняется результативная остановка, в случае **б** — безрезультативная остановка, а в случае **в** — зацикливание работы машины.

В конце урока, если останется время, можно предложить следующую задачу.

Задача. На приведенном ниже рисунке даны начальное (н.с.) и конечное (к.с.) состояния ленты. Напишите программу для машины Поста, в результате выполнения которой машина переведет начальное состояние ленты в конечное.



Тема 2

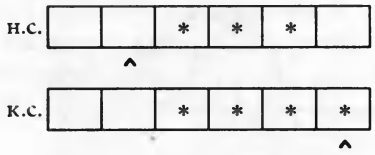
Получение практического навыка работы с компьютерной реализацией машины Поста. Предлагаемые задачи могут быть подобны приведенной выше. При решении задач нумерация команд в программе проводится наподобие нумерации в Бейсике, т.е. начиная с номера 10 и через десяток.

Темы 3, 4, 5

Написание программ для реализации на машине Поста простейших вычислений. На машине Поста можно производить арифметические действия над числами

(целыми неотрицательными), которые записываются в унарной системе счисления. Число на ленте записывается в виде последовательности подряд идущих меток. Число 0 изображается одной меткой, 1 — двумя метками и т.д.

Рассмотрение выполнения арифметических действий начинается с операции сложения данного числа с единицей. В следующем примере к числу 2 прибавляется единица.



Задача 1. Прибавление единицы в простейшем случае.

Требуется написать программу для машины Поста, обладающую следующим свойством. Каково бы ни было число n, если начальное состояние машины Поста таково, что на ленте имеется запись числа n (а в остальном лента пуста) и каретка стоит против самой левой ячейки записи, то выполнение программы должно привести к результативной остановке, после чего на ленте (в произвольном ее месте) должно быть записано число $n + 1$ (а в остальном лента должна быть пуста), причем каретка может стоять где угодно.

Идея решения задачи такова: надо прибавить к исходной последовательности меток одну метку слева или справа. Соответственно получается два варианта программы.

Вариант 1

```
10 ← 20
20 ! 30
30 stop
```

Вариант 2

```
10 → 20
20 ? 30 |10
30 ! 40
40 stop
```

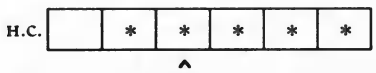
Сильным ученикам можно предложить доказать, что никакая программа из двух команд не будет решением задачи 1. Идея доказательства заключается в следующем. В программе должно быть как минимум две команды — смещение каретки и постановка метки, но формат этих команд предполагает обязательное наличие отсылки (номер команды, которая будет выполняться следующей). Отсылка во второй команде не может совпадать с номером первой или второй команды, так как в этом случае произойдет или безрезультативная остановка (в компьютерной реализации будет выдано сообщение об ошибке), или бесконечная работа машины, что не приведет нас к решению задачи. Значит, отсылка должна соответствовать третьей команде в программе. Вообще в пособии [1] приводится много задач на доказательство. Использовать их можно по усмотрению учителя при работе с сильными учениками.

Следующий шаг — прибавление единицы в более сложных условиях.

Не будем требовать, чтобы каретка в начале стояла непременно против одной из крайних ячеек последовательности. Ограничимся требованием, чтобы в начальном состоянии каретка обозревала одну из ячеек последовательности меток.

Задача 2. Требуется написать программу для машины Поста, обладающую следующим свойством. Каково бы ни было число n, если начальное состояние машины Поста таково, что на ленте имеется запись числа n (а в остальном лента пуста) и каретка стоит против одной из ячеек записи, то выполнение программы должно привести к результативной остановке, после чего на ленте (в произвольном ее месте) должно быть записано число $n + 1$ (а в остальном лента должна быть пуста), причем каретка может стоять где угодно.

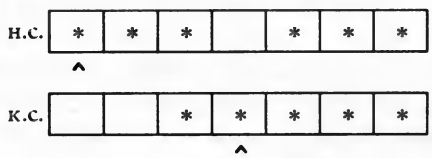
Идея решения задачи такова: надо прибавить к исходной последовательности меток одну метку слева или справа. Соответственно получаются два варианта программы.



Ученикам надо предложить не отсчитывать ячейки до ближайшей свободной, а применить цикл типа "пока" для поиска первой пустой ячейки. Один из вариантов программы может быть такой:

```
10 ← 20
20 ? 30 | 10
30 ! 40
40 stop
```

Более сложные задачи возникают при сложении чисел. Начать рассмотрение сложения чисел можно с задач с фиксированным положением каретки, в которых числа записаны на расстоянии в одну ячейку друг от друга.



Задача 3. Составить программу сложения двух чисел, записанных на расстоянии в 1 ячейку друг от друга. Идея решения заключается в следующем: стираем две метки в первом слагаемом (находится левее на ленте), находим пустую ячейку, расположенную между метками, и отмечаем ее. Ниже приводится программа, реализующая этот алгоритм.

```
10 # 20
20 → 30
30 #40
40 → 50
50 ? 60 |40
60 ! 70
70 stop
```

Наконец, более общий случай сложения чисел

Задача 4. Составить программу для сложения двух чисел, расположенных на произвольном расстоянии друг от друга. Каретка в начальный момент обозревает самую левую метку в первом слагаемом.

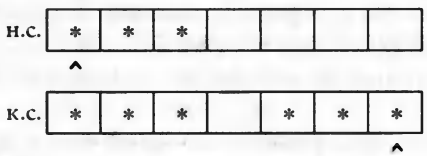
Ниже приводится один из наиболее коротких вариантов этой программы.

```
{Сложение чисел, расположенных на
произвольном расстоянии}
metka 13,14,32,33,34,35
start 13
10 # 20
20 → 25
25 ? 110 | 30
30 # 40
40 → 50
50 ? 60 | 40
60 ! 70
70 → 80
80 ? 90 | 110
90 ← 100
100 ? 20 | 90
110 stop
```

Темы 7 и 8

Изучением этих тем целесообразно заниматься в основном с сильными учениками. Задачи, предлагаемые в них, можно использовать в качестве рефератов для сдачи экзаменов. Приведем некоторые из них.

Копирование последовательности меток



Необходимо составить программу, копирующую последовательность меток на один шаг вправо. Это вспомогательная задача для умножения чисел. Вот один из возможных вариантов решения:

```
metka 13,14,15,16,17
start 13
1 → 2
2 # 3
3 → 4
4 ? 5 | 3
5 → 6
6 ? 7 | 5
7 ! 8
8 ← 9
9 ? 10 | 8
10 ← 11
11 ← 12
12 ? 17 | 13
13 ← 14
14 ? 15 | 13
15 → 16
16 # 3
17 ! 18
18 ← 19
19 ? 17 | 20
20 → 21
21 ? 22 | 20
22 → 23
23 ? 24 | 22
24 ! 25
25 → 26
26 ! 27
27 stop
```

Алгоритм заключается в удалении всех меток, кроме двух крайних, и постановке этих меток на расстоянии в одну ячейку от крайней метки. Затем к этим меткам добавляются еще две. Программа работает корректно для последовательности из трех и более меток.

Умножение чисел

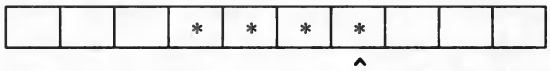
Данная программа реализует умножение чисел n и m, не равных 0 или 1. Идея алгоритма заключается в последовательном копировании второго множителя m раз и сложении полученных чисел. Копирование производится по алгоритму, изложенному выше. Исходные числа должны быть записаны на расстоянии в одну ячейку друг от друга.

```
{Произведение двух чисел. В начальном состо-
янии каретка стоит на начале первого множи-
теля. Алгоритм основан на общем способе
вычисления произведения путем замены его
сложением по формуле:
n * m = n + n + ...+ n (m раз)}
metka 2,3,4,5,6,8,9,10,11
start 2
{Удаление первой метки в первом множителе}
10 # 20
20 → 30
30 ? 40 | 20
40 → 50
{Копирование второго множителя n раз}
50 → 60
60 # 70
70 → 80
80 ? 90 | 70
90 → 100
100 ? 110 | 90
110 ! 120
120 ← 130
130 ? 140 | 120
140 ←  150
150 ← 160
160 ? 210 | 170
170 ←  180
180 ? 190 | 170
190 → 200
200 # 70
210 ! 220
220 ← 230
230 ? 210 | 240
240 → 250
250 ? 260 | 240
260 → 270
270 ? 280 | 260
280 ! 290
```

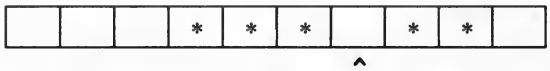
```
290 → 300
300 ! 310
{Строки 310 – 340 - перегон каретки за первую занятую ячейку}
310 ← 320
320 ? 330 | 310
330 ← 340
340 ? 350 | 310
350 → 360
360 → 370
370 # 380
380 → 390
390 → 400
400 ? 410 | 440
{Если первый множитель исчерпан, то переход на процедуру сложения к метке 580}
410 ← 420
420 # 580
{Строки 440 – 470 - перегон каретки за последнюю занятую ячейку}
440 → 450
450 ? 460 | 440
460 → 470
470 ? 480 | 440
480 ← 490
490 ← 500
500 ? 510 | 490
510 → 50
580 → 590
590 → 600
{Процедура сложения}
600 # 610
610 → 620
620 # 630
630 → 640
640 ? 650 | 630
650 ! 660
660 → 670
670 ? 680 | 700
680 ← 690
690 ? 610 | 680
700 → 710
710 ? 720 | 700
{Проверка наличия еще одного числа, расположенного правее текущего}
720 → 730
730 ? 740 | 750
740 stop
750 ← 760
760 ← 770
770 ? 780 | 760
780 → 600
```

В конце изучения темы можно предложить учащимся написать контрольную работу. Ниже приводится один из вариантов такой работы, она рассчитана на один академический час. При наличии времени можно предложить ребятам проверить свои программы на компьютере.

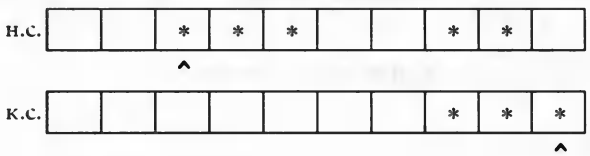
1. Прибавить единицу.



2. Сложить числа.



3. Составить программу для перевода машины Поста из начального состояния в конечное.



Литература

1. *Успенский В.А.* Машина Поста. Популярные лекции по математике, вып. 52. М.: Наука, 1988.

Материал, посвященный машине Поста, который вы только что прочитали, был прислан в редакцию учителем, обратившим внимание на следующее объявление:

ДОРОГИЕ ЧИТАТЕЛИ!

Возможно, некоторые из вас даже не подозревают о том, что стать автором "Информатики" (а это значит "заработать" лишнюю, а для многих совсем не лишнюю публикацию и получить за это совсем не маленький гонорар) совсем просто. Для этого надо всего лишь отправить нам (любым способом, хотя посредством электронной почты быстрее и предпочтительнее) материал, который вы предлагаете для опубликования в нашей газете. Мы не накладываем никаких ограничений на тематику материалов (конечно, они должны иметь отношение к информатике). После получения нами материалов они поступят опытным и доброжелательным рецензентам и редакторам, которые, возможно, свяжутся с вами (поэтому, пожалуйста, указывайте свои адреса и телефоны). Вы всегда можете поинтересоваться "судьбой" своих материалов, но делать это имеет смысл не ранее чем через две-три недели после их поступления в редакцию (это минимальное время, за которое мы можем принять какое-то решение).

В какой форме должны быть представлены материалы?

Учитывая, сколь разные технические возможности имеют наши авторы, мы не предъявляем жестких требований к форме представления материалов и готовы обсудить этот вопрос в каждом конкретном случае.

Поэтому ответим на другой вопрос: **в какой форме нам удобно получать материалы?**

Для проведения цикла редакционной обработки нам удобно, чтобы вы представляли текст статьи в формате DOC-файла (WinWord 6.0). Все иллюстрации и программы должны быть помещены в текст и, кроме того, приложены в виде отдельных файлов (программы — в виде текстовых файлов). Редакторы, как правило, проверяют все программы; конечно, они могут "извлечь" их из DOC-файла, но удобнее иметь их отдельно, это же касается и иллюстраций. Наиболее удобный для нас формат файлов растровых иллюстраций — TIFF (150 dpi), векторных — WMF. Но не пытайтесь (особенно в "сложных" случаях) предоставить нам иллюстрации высокого качества, не тратьте на это свое время. Обычно нам достаточно иметь эскиз, над которым в дальнейшем поработают наши художники. Проще говоря, *нам важно точно знать, что именно вы хотите изобразить*. Отдельный вопрос — снимки экранов. Их мы редактировать и перерисовывать, конечно, не можем, они должны быть представлены именно в том виде, в котором и будут помещены в газете. Нам удобно, чтобы снимки экранов делались в разрешении не менее 800 × 600 × 256 цветов. Но это опять же лишь пожелание.

Присылайте нам свои материалы, заметки, статьи. Мы вместе поработаем над тем, чтобы они были опубликованы на страницах "Информатики".

Все наши адреса, по которым следует направлять материалы, указаны на последней странице каждого номера.

"Назаметки" Сидорова

А.А. Дуванов,
г. Переславль-Залесский

Продолжение. См. № 6, 7, 8/2001

"Назаметка" 7. Схема навигации

В "Кухне Сидорова" использована простейшая схема навигации.

Меню текстовых ссылок — для перехода к "потомкам".

Кнопки [◄] и [►] — для движения по страницам-"сестрам". Кнопка [●] — для перехода к "родителю".

Кнопка [⌂] — для перехода к "основателю рода".

Когда сайт небольшой и состоит из нескольких не подчиненных друг другу страниц, можно использовать навигационную схему, содержащую одно и то же меню на каждой странице. Для большего удобства это меню повторяется дважды: в начале и в конце страницы.

логотип сайта **название сайта**

начало | страница1 | страница2 | страница3 | страница4
название страницы

содержимое страницы

начало | страница1 | страница2 | страница3 | страница4
автор сайта и его электронный адрес

Следует отметить следующее:

— На всех страницах меню должно иметь один и тот же вид, за исключением того, что раздел, указывающий на текущую страницу, не должен быть ссылкой. Но он обязательно должен присутствовать. Его состояние (все другие разделы являются ссылками, а он нет) дополнительно показывает пользователю местоположение на сайте. Присутствие в меню записи о текущей странице говорит и о месте текущей страницы по отношению к другим.

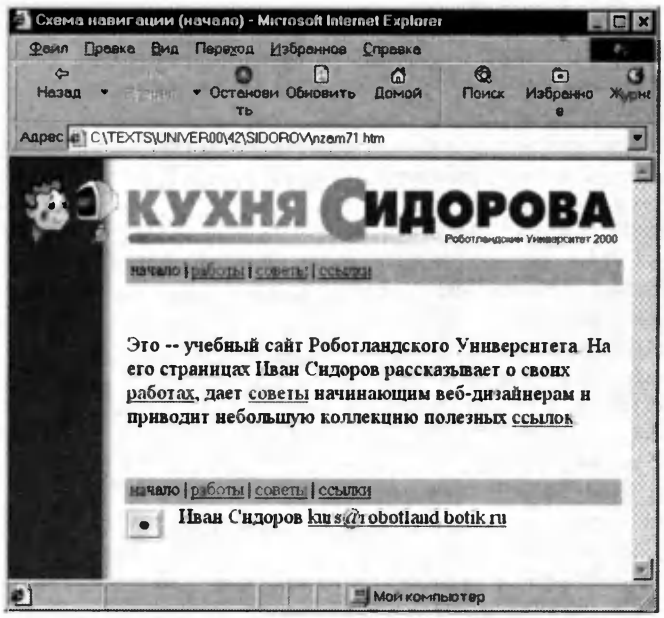
— Меню на каждой странице должно содержать одни и те же разделы, они должны следовать в одном и том же порядке и иметь одинаковое визуальное представление.

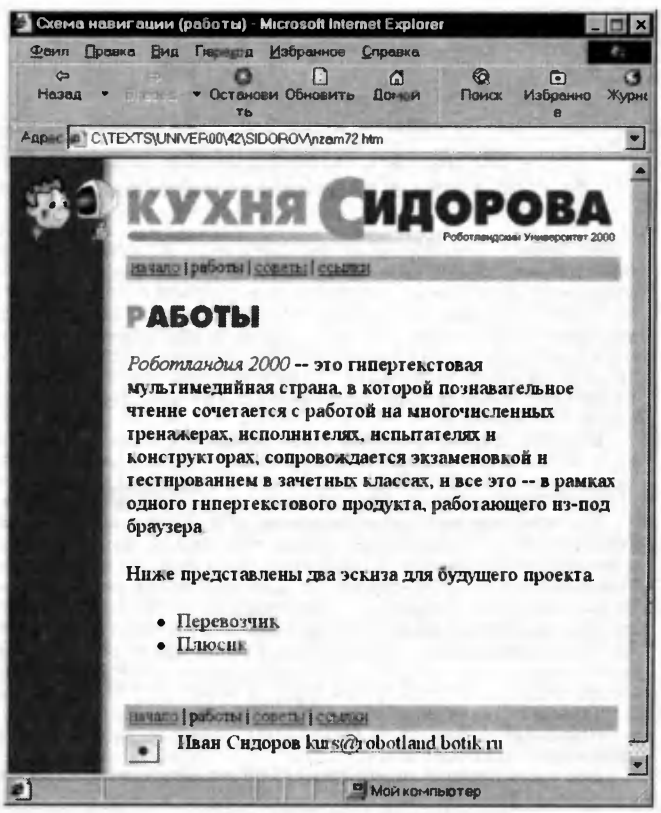
— Логотип сайта на всех страницах, кроме начальной, должен быть ссылкой на начальную страницу.

— Меню в начале уместно. Пользователь решает, куда ему перейти после загрузки страницы. Меню в конце страницы не менее уместно. Пользователь дочитал страницу до конца. Что же теперь, прокручивать ее наверх, чтобы выбрать другой раздел? Неудобно! Хорошо, когда меню под рукой там, где оно требуется.

Все эти правила преследуют цель облегчить пользователю навигацию по сайту и сделать ее максимально комфортной.

На следующих двух рисунках представлены страницы "Кухни Сидорова" с такой схемой навигации.





"Назаметка" 8. Универсальная схема навигации

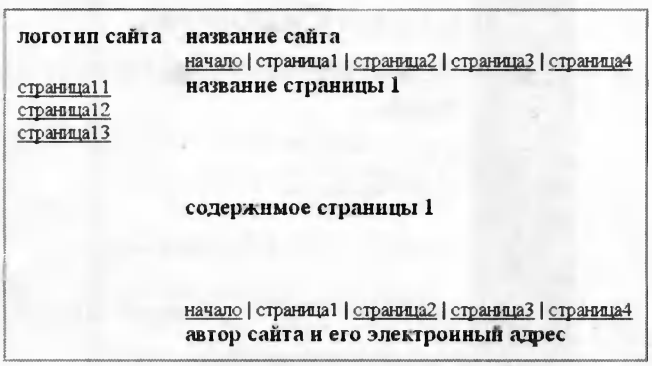
Как организовать навигацию, если сайт состоит из нескольких страниц, каждая из которых содержит подчиненные страницы?

```
начало
страница1
  страница11
  страница12
  страница13
страница2
  страница21
  страница22
  страница23
страница3
  страница31
  страница32
  страница33
страница4
  страница41
  страница42
  страница43
```

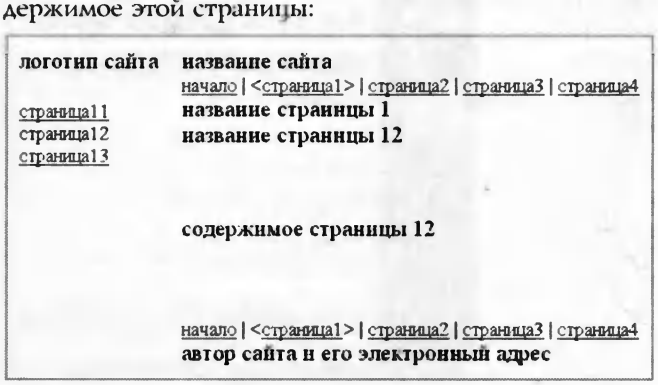
Двухуровневая навигация

Первое, что приходит в голову, — это взять навигационную схему, описанную в предыдущей "назаметке" 7, но в каждой позиции меню при указании на нее курсором мыши показывать ниспадающий список — подменю. Такая технология требует использования скриптов, но главное, она не слишком хороша: подменю пользователь видит не постоянно, и ему сложнее понять, где он находится и куда может пойти.

Предлагается следующая навигационная схема:



На левой вертикальной полоске отображается меню текущей станицы. Если, например, пользователь щелкает по ссылке страница12, он видит на экране содержимое этой страницы:

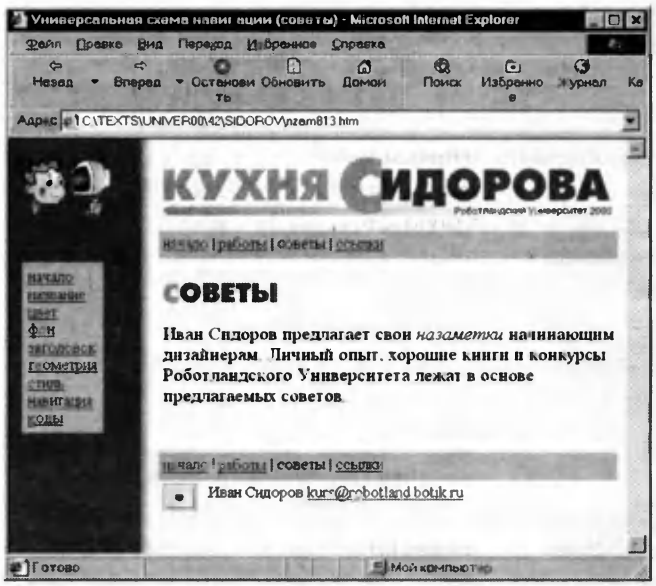


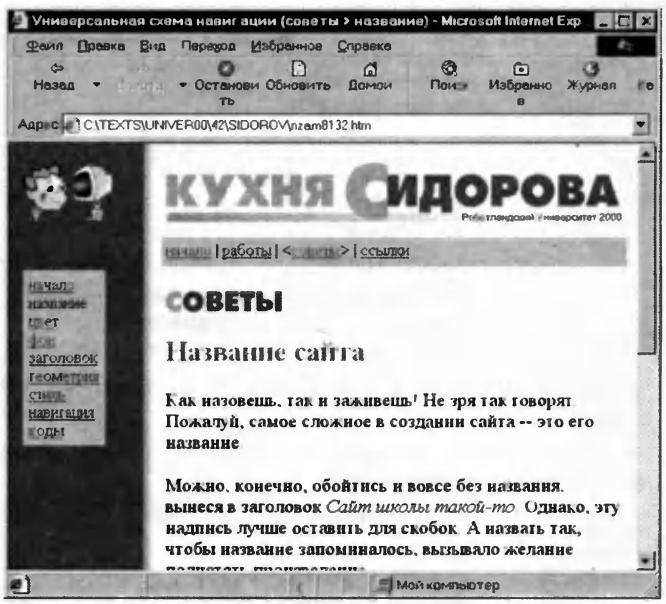
Такая схема хорошо подходит для двухуровневой иерархии.

Меню первого уровня располагается сверху и снизу страницы. Страница, которая является родителем текущей, выделяется (на схеме выделение выполнено при помощи угловых скобок).

Меню второго уровня располагается на вертикальной полоске слева. Раздел, соответствующий текущей странице, присутствует, но не является ссылкой.

На рисунках представлены страницы "Кухни Сидорова" с такой схемой навигации.





Универсальная навигация

Представить себе сайт, содержащий только два уровня иерархии, трудно, обычно уровней гораздо больше. Как в этом случае организовать навигацию?

Сделаем так. В левой полоске будем отображать меню текущей страницы, а чтобы пользователь не запутался, дополнительно к главному горизонтальному меню бу-

дем строить путь к текущей странице от первого этажа гипертекстовой иерархии.

Пусть сайт имеет следующую структуру:

```
начало
страница1
  страница11
  страница12
    страница121
    страница122
    страница123
  страница13
страница2
страница3
страница4
```

На странице 1 пользователь видит такую картинку:

логотип сайта	**название сайта**
	начало | страница1 | страница2 | страница3 | страница4
страница11	страница1
страница12	**название страницы 1**
страница13	
	содержимое страницы 1
	страница1
	начало | страница1 | страница2 | страница3 | страница4
	автор сайта и его электронный адрес

Щелчок по ссылке страница12 меняет экран так:

логотип сайта	**название сайта**
	начало | <страница1> | страница2 | страница3 | страница4
страница121	страница1 > страница12
страница122	**название страницы 1**
страница123	**название страницы 12**
	содержимое страницы 12
	страница1 > страница12
	начало | <страница1> | страница2 | страница3 | страница4
	автор сайта и его электронный адрес

Если теперь щелкнуть по ссылке страница121, то экран примет вид:

логотип сайта	**название сайта**
	начало | <страница1> | страница2 | страница3 | страница4
страница121	страница1 > страница12 > страница121
страница122	**название страницы 1**
страница123	**название страницы 12**
	название страницы 121
	содержимое страницы 121
	страница1 > страница12 > страница121
	начало | <страница1> | страница2 | страница3 | страница4
	автор сайта и его электронный адрес

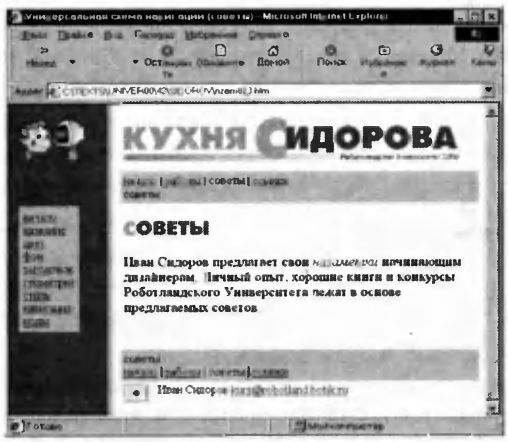
Строка:

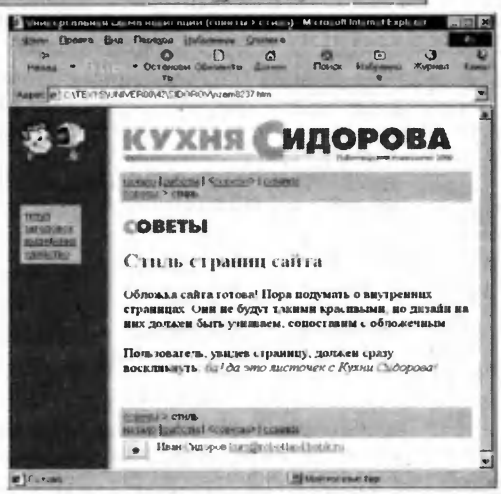
страница1 > страница12 > страница121

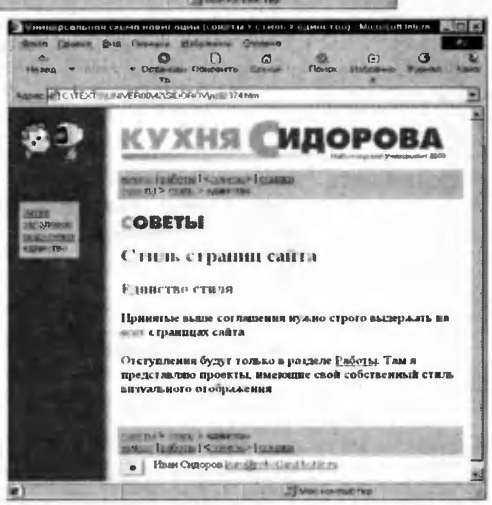
показывает, что сейчас на экране страница121 (эта запись не является ссылкой), что верхние иерархические этажи этой страницы страница12 и страница1 и мы можем напрямую попасть на любой уровень в этой цепочке.

Очень важно в иерархической схеме иметь ориентированный значок типа ">" — он показывает направление движения от корня вглубь иерархии. Заменять этот значок на безликий знак типа "—" не рекомендуется.

На рисунках, приведенных ниже, представлены страницы "Кухни Сидорова" с такой схемой навигации.







Holos + ...графия

Голография (от греч. *holos* — весь, полный и graphō — пишу) — способ "получения объемного изображения предмета, основанный на интерференции[1] двух лучей света — от источника и от предмета (см. *голограмма*); при освещении голограммы светом той же длины волны, что и у опорного луча, в результате дифракции[2] света возникает объемное (при определенных условиях цветное) изображение предмета; кроме оптической, существует акустическая голография".

Картина интерференции

Голограмма — "полученная, зарегистрированная на фотопластинке картина интерференции двух лучей света от одного источника (лазера): луча, отраженного предметом, и луча, непосредственно идущего от источника света (опорного луча)" [1].

В 1947 году было сделано научное открытие, к которому сначала отнеслись просто как к очередному доказательству волновых свойств света, но впоследствии выяснилось, что оно является значительно более важным [2]. А известно об этом открытии стало спустя примерно год, когда английский физик венгерского происхождения Деннис Габор сообщил о разработанном им методе получения объемных изображений, названном голографией. В отличие от фотографии, которая фиксирует только интенсивность света и формирует плоское изображение объекта, голография регистрирует так называемый волновой фронт светового луча и воспроизводит трехмерное изображение предмета.

Д. Габор

Еще через полтора десятка лет американские физики Эммет Лейт и Юрис Упатниекс получили первые голограммы с помощью лазерного луча, причем эти исследователи до некоторой степени изменили схему Габора. Теперь при создании голограмм использовалось разделение исходного светового луча на два с помощью полупрозрачного зеркала. Одна часть света отражалась от объекта, другая попадала прямо на фотопластинку. Там световые лучи накладываются друг на друга (интерферируют) и, поскольку они различаются по фазе, возникает сложная интерференционная картина, которая записывается на светочувствительную эмульсию. Это и есть голограмма. На ней можно увидеть как бы орнамент из черточек, точек, пятнышек. Голограмма в какой-то степени похожа на испорченный негатив, однако обладает необычными свойствами. Если теперь направить на голограмму первоначальный, опорный пучок света (под тем же углом, как и при записи), то она будет играть роль дифракционной решетки[3] и восстановит волновой фронт луча, отраженного от предмета, в результате чего возникнет его трехмерное изображение.

Интересно, что если разделить голограмму на части, то каждая из них также позволяет восстанавливать изображение, хотя качество его будет более низким. Нечто похожее происходит и в человеческом мозгу: при некоторых его поражениях память не теряется полностью, а лишь ухудшается.

При получении первых голограмм Деннис Габор (удостоенный в 1971 году Нобелевской премии по физике за создание голографии) столкнулся с большими трудностями, поскольку для работы требовались специальные источники света, которых тогда не было. Появление лазеров дало голографии новую жизнь.

Существенный вклад в развитие голографии внес Юрий Николаевич Денисюк [3, 4, 5, 6, 7]. В 1962 году он предложил один из самых интересных способов получения голограмм, предполагающий их запись в трехмерной среде. При использовании данного метода изображение предмета удается воспроизвести с помощью обычного источника света.

Ю.Н. Денисюк

Голограмма может быть изготовлена и с помощью компьютера (цифровая голография). Такой способ употребляется, к примеру, для формирования объемных изображений не существующих еще объектов [3, 4].

Голография нашла применение в экспериментальной физике, изобразительном искусстве, технике распознавания и других областях. Возможности использования голографических методов велики и изучены далеко не полностью [3, 7]. И очень важно, что эти методы позволяют записывать, хранить и очень быстро преобразовывать громадное количество информации...

Литература

1. Словарь иностранных слов. 15-е изд., испр. М.: Русский язык, 1988.

2. *Чолаков В.* Нобелевские премии. Ученые и открытия: Пер. с болг. М.: Мир, 1986.

3. *Федоров Б.Ф., Цибулькин Л.М.* Голография. М.: Радио и связь, 1989.

4. Физический энциклопедический словарь. М.: Советская энциклопедия, 1984.

5. *Храмов Ю.А.* Физики. Биографический справочник. 2-е изд. М.: Наука, Гл. редакция физико-математической литературы, 1983.

6. Энциклопедический словарь юного физика. 2-е изд. М.: Педагогика, 1991.

7. *Дикарев В.И.* Справочник изобретателя. СПб.: Лань, 1999.

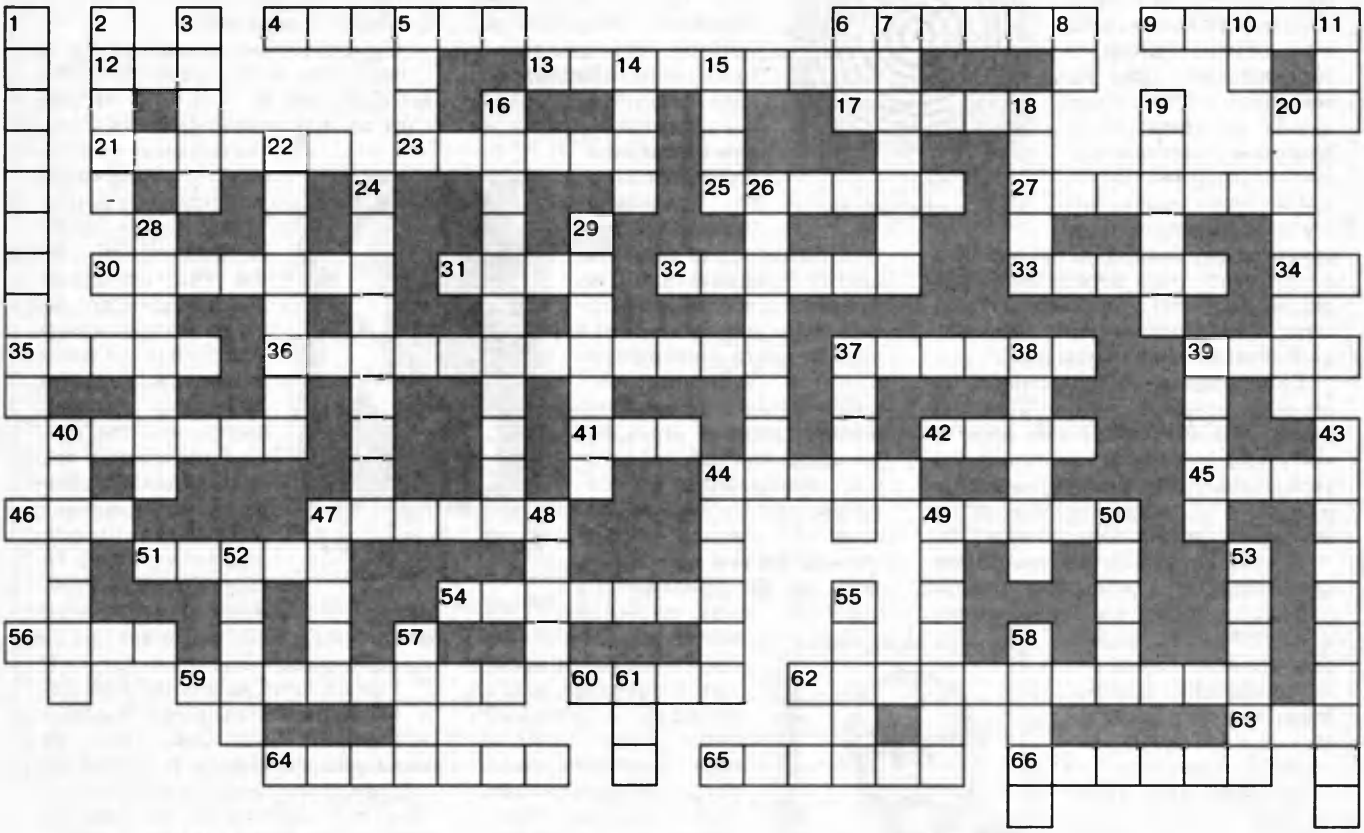
[1] Интерференция волн — сложение в пространстве двух (или нескольких) волн, при котором в разных его точках получается усиление или ослабление амплитуды результирующей волны.

[2] Дифракция (в узком, наиболее употребительном смысле) — огибание волнами (световыми, звуковыми и др.) препятствий.

[3] Дифракционная решетка — оптический прибор, представляющий собой, например, совокупность большого числа параллельных щелей в каком-либо непрозрачном экране.

Кроссворд "Текстовый редактор Microsoft Word"

Д.М. Златопольский,
Москва



По горизонтали:

4. Примечание к тексту, размещаемое в нижней части страницы или в конце документа.
6. Смещение текста абзаца по отношению к границам полей.
9. Единица измерения высоты шрифта.
12. Старинная французская мера длины, упоминаемая в романе Ж.Верна.
13. Часть таблицы.
17. Часть текста, набранная до нажатия клавиши Enter.
20. Название буквы φ греческого алфавита.
21. Поколение, вариант Microsoft Word.
23. Служебный файл Microsoft Word с расширением DOT, содержащий информацию о структуре и оформлении документа.
25. Наклонный шрифт.
27. Выражение, оформляемое с помощью Microsoft Equation.
30. Изображение на инструменте, выполняющем сохранение документа.
31. Судьба, участь.
32. Пункт главного (системного) меню Microsoft Word.
33. Разработчик документа или программы.
34. Название буквы μ греческого алфавита.
35. Программа, обладающая способностью к самовоспроизведению.
36. Действие над абзацем.
37. Пункт главного (системного) меню Microsoft Word.
39. Затемненное место вокруг рамки данного текста.
40. Промежуток между словами.
41. Линия ровно расположенных однородных предметов.

42. Графический элемент, который "обтекается" текстом.
44. Размер (высота) шрифта.
45. Часть слова.
46. Пункт главного (системного) меню Microsoft Word.
47. Часть документа, границу которого можно вставить с помощью разрыва (пункт меню "Вставка").
49. Знак, входящий в алфавит.
51. Услуга, предоставляемая телекоммуникационными сетями, с помощью которой можно переслать созданный документ.
54. Отношение размеров элементов документа на экране к натуральным размерам.
56. Элемент окна Microsoft Word.
59. Нумеруемая часть документа, параметры которой устанавливаются с помощью пункта меню "Файл".
60. Носитель, на котором хранится созданный документ.
62. Действие, проводимое с фрагментом текста.
63. Карточный термин (на него ставят ставку).
64. Часть окна текстового редактора, используемая для установки полей, отступов и т.п.
65. ... прокрутки.
66. Ввоз компьютеров и программ из-за рубежа.

По вертикали:

1. Число экземпляров документа.
2. Последовательность букв и цифр, ограниченная с обоих концов пробелами, запятыми, точками, дефисами и т.п.
3. Сторона грани многогранника.
5. Название символа "/".
7. Данные, расположенные по графам.
8. Название буквы греческого алфавита, используемой в геометрии.
10. Название буквы ν греческого алфавита.
11. Название цифры.
14. Часть экрана, занимаемая документом.
15. Отблеск света на экране монитора.
16. Действие, проводимое над ячейками таблицы.
18. Знак, обозначающий число.
19. Буква, цифра, знак препинания.
20. Часть текста.
22. Характеристика абзаца.
24. Графическое изображение цифровой информации.
26. Действие над фрагментом текста.
28. Графический объект в текстовом документе.
29. Набор, используемый при проверке орфографии.
37. Совокупность характеристик символа или абзаца.
38. Символ уменьшенного размера, размещаемый справа внизу или справа вверху другого символа.
39. Последовательность символов.
40. Изображение на инструменте, выполняющем печать документа.
43. Перечень глав и пунктов документа.
44. Смотанные шариком нитки.
47. Прямоугольник, обрамляющий абзац.
48. Кусок бумаги.
49. Большая буква в начале абзаца, оформленная отличным от основного текста образом.
50. Первая буква греческого алфавита.
52. Доля чего-то целого, например, абзаца, документа и т.п.
55. Металл белого цвета, используемый для пайки, в т.ч. при ремонте компьютера.
57. Знак препинания.
58. Второй экземпляр документа.
61. Обозначение документа.

Ответы:

По горизонтали: 4. Сноска. 6. Отступ. 9. Пункт. 12. Лье. 13. Столбец. 17. Абзац. 20. Фи. 21. Версия. 23. Шаблон. 25. Курсив. 27. Формула. 30. Дискета. 31. Удел. 32. Правка. 33. Автор. 34. Мю. 35. Вирус. 36. Выравнивание. 37. Сервис. 39. Тень. 40. Пробел. 41. Ряд. 42. Кадр. 44. Кегль. 45. Слог. 46. Вид. 47. Раздел. 49. Буква. 51. Почта. 54. Масштаб. 56. Меню. 59. Страница. 60. Диск. 62. Копирование. 63. Кон. 64. Линейка. 65. Полоса. 66. Импорт.

По вертикали: 1. Количество. 2. Слово. 3. Ребро. 5. "Слэш". 7. Таблица. 8. Пи. 10. Ню. 11. Три. 14. Окно. 15. Блик. 16. Объединение. 18. Цифра. 19. Символ. 20. Фрагмент. 22. Интервал. 24. Диаграмма. 26. Удаление. 28. Рисунок. 29. Словарь. 37. Стиль. 38. Индекс. 39. Текст. 40. Принтер. 43. Оглавление. 44. Клубок. 47. Рамка. 48. Лист. 49. Буквица. 50. Альфа. 52. Часть. 53. Объект. 55. Олово. 57. Тире. 58. Копия. 61. Имя.

Георг Ом и его закон

Окончание. Начало на с. 1

Ом создал источник тока в виде термопары, образованной висмутовым стержнем и замыкающей его концы медной проволокой. [Вообще термопара представляет собой датчик температуры, состоящий из двух соединенных между собой разнородных электропроводящих элементов. Если контакты (обычно спаи) проводящих элементов, образующих термопару (их часто называют термоэлектродами), находятся при разных температурах, то в цепи термопары возникает электродвижущая сила, величина которой однозначно определяется температурой горячего и холодного контактов и природой материалов, используемых в качестве термоэлектродов.] Это была достаточно причудливая конструкция, в которой предусматривался даже оптический визир для регистрации угла отклонения магнитной стрелки (в качестве "контрольно-измерительной аппаратуры" Ом употреблял магнитную стрелку и электроскоп — прибор для обнаружения и измерения электрических зарядов) [3, 4]. Один из концов стержня помещался в кипящую воду, а другой — в мелко наколотый лед. Вскоре ученый пришел к выводу, что между электрическим током и равномерным водным потоком в русле существует аналогия. Чем больше перепад уровней в русле и свободнее путь, тем значительней расход. Нечто подобное происходит и с электрическим током: сила тока будет тем больше, чем большей электродвижущей силой обладает источник и чем меньше сопротивление току на его пути.

"Причудливая конструкция", которую использовал Ом

Немного позже Ом обосновал этот закон теоретически (для участка и полной цепи), ввел понятия *электродвижущей силы*, *падения напряжения* и *проводимости*, а в 1830 году выполнил первые измерения электродвижущей силы источника тока [5].

Закон Ома является центральным в электродинамике, он послужил основой для установления других законов. При этом соответствующее соотношение справедливо и для цепей переменного тока.

С установлением количественного соотношения между основными параметрами электрической цепи появлялись широкие возможности для изучения электрических явлений. Однако новый закон долгое время не получал признания. И только когда такие ученые, как Эмилий Христианович Ленц, Борис Семенович Якоби, Карл Фридрих Гаусс, Густав Роберт Кирхгоф, а также некоторые другие положили открытый закон в основу своих исследований, его значение стало очевидным [1].

Последние десятилетия своей жизни Ом посвятил работам, главным образом, в области акустики [1, 4, 5, 6]. В 1843 году он показал, что простейшее слуховое ощущение вызывается гармоническими колебаниями, на которые ухо разлагает сложные звуки. Позже акустический закон Ома был использован его соотечественником Германом Людвигом Фердинандом Гельмгольцем в качестве основы для резонансной теории слуха.

Кроме того, Ом вел исследования в области оптики и кристаллооптики.

В 1881 году, через 27 лет после смерти ученого, на Международном конгрессе электриков его именем была названа единица сопротивления (*ом*). Ом равен сопротивлению проводника, между концами которого при силе тока 1 А возникает напряжение 1 В.

Литература

1. Ом // Большая советская энциклопедия. 2-е изд. М.: Гл. науч. изд-во "Большая советская энциклопедия", 1955. Т. 35.

2. Физический энциклопедический словарь. М.: Советская энциклопедия, 1984.

3. Вольт, ампер, ом и другие. Единицы физических величин в технике связи. М.: Радио и связь, 1988.

4. *Смирнов О.А., Майорова Т.С., Власова И.Г.* 100 великих имен в математике, физике и географии. М.: Филологическое общество "СЛОВО", 1998.

5. *Храмов Ю.А.* Физики. Биографический справочник. 2-е изд. М.: Наука, Гл. редакция физико-математической литературы, 1983.

6. Энциклопедический словарь юного физика. 2-е изд. М.: Педагогика, 1991.

Гл. редактор
С.Л. Островский
Зам. гл. редактора
И.Н. Фалина
Редакция:
Е.В. Андреева
Н.Л. Беленькая
Л.Н. Картвелишвили
Н.П. Медведева
Дизайн и верстка:
Н.И. Пронская
Корректоры:
Е.Л. Володина,
С.М. Подберезина

Адрес редакции и издателя:
121165, Киевская, 24
тел. 249-48-96
Отдел рекламы
тел. 249-98-70

Учредитель: ООО "Чистые пруды"
Зарегистрировано в Комитете РФ по печати 20.03.1997.
Регистрационный номер 015873.
Отпечатано в типографии ОАО ПО "Пресса-1".
125865, ГСП, Москва, ул. "Правды", 24.
Тираж 6600 экз.
Срок подписания в печать по графику 28.02.2001.
Номер подписан 28.02.2001.
Заказ № 11422 Цена свободная

ИНДЕКС ПОДПИСКИ
для индивидуальных подписчиков 32291
комплекта приложений 32744

Тел.: (095)249-31-38, 249-33-86. Факс (095)249-31-84

Internet: inf@1september.ru
WWW: http://www.1september.ru

Номера "Информатики" (в ограниченном количестве) **бесплатно** распространяются в офисе ведущей российской антивирусной фирмы "ДиалогНаука".
Адрес: Москва, ул. Вавилова, д. 40, офис 103. Тел. 137-01-50.

Комплект приложений, главный редактор — А.С. Соловейчик

Английский язык (Е.В. Громушкина), индекс подписки — 32025; **Библиотека в школе** (О.К. Громова), индекс подписки — 33376; **Биология** (Н.Г. Иванова), индекс подписки — 32026; **Воскресная школа** (монах Киприан (Ященко), индекс подписки — 32742; **География** (О.Н. Коротова), индекс подписки — 32027; **Здоровье детей** (А.У. Лекманов), индекс подписки — 32033; **Информатика** (С.Л. Островский), индекс подписки — 32291; **Искусство** (Н.Х. Исмаилова), индекс подписки — 32584; **История** (А.Ю. Головатенко), индекс подписки — 32028; **Литература** (Г.Г. Красухин), индекс подписки — 32029; **Математика** (И.Л. Соловейчик), индекс подписки — 32030; **Начальная школа** (М.В. Соловейчик), индекс подписки — 32031; **Немецкий язык** (М.Д. Бузоева), индекс подписки — 32292; **Русский язык** (Л.А. Гончар), индекс подписки — 32383; **Спорт в школе** (Н.В. Школьникова), индекс подписки — 32384; **Управление школой** (А.И. Адамский), индекс подписки — 32652; **Физика** (Н.Д. Козлова), индекс подписки — 32032; **Французский язык** (Г.А. Чесновицкая), индекс подписки — 33371; **Химия** (О.Г. Блохина), индекс подписки — 32034; **Школьный психолог** (М.Н. Сартан), индекс подписки — 32898.