АКАДЕМИЯ НАУК СССР
НАУЧНЫЙ ЦЕНТР БИОЛОГИЧЕСКИХ ИССЛЕДОВАНИЙ
НАУЧНО-ИССЛЕДОВАТЕЛЬСКИЙ ВЫЧИСЛИТЕЛЬНЫЙ ЦЕНТР

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

Н. Л. ЛУНИНА

ФАКУЛЬТАТИВНЫЙ КУРС ПРОГРАММИРОВАНИЯ

ПАСКАЛЬ

ПУЩИНО.1989

АКАДЕМИЯ НАУК СССР
НАУЧНЫЙ ЦЕНТР БИОЛОГИЧЕСКИХ ИССЛЕДОВАНИЙ
НАУЧНО-ИССЛЕДОВАТЕЛЬСКИЙ ВЫЧИСЛИТЕЛЬНЫЙ ЦЕНТР

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

Н. Л. ЛУНИНА

ФАКУЛЬТАТИВНЫЙ КУРС ПРОГРАММИРОВАНИЯ

ПАСКАЛЬ

ПУЩИНО.1989

УДК 512.519.68

В работе изложена третья часть факультативного курса программирования. Курс рассчитан на два года и ориентирован на учащихся 7-8 классов. Общая программа курса

Основные понятия

Бэйсик

Паскаль

Фортран

Работа будет полезна старшеклассникам и преподавателям информатики и программирования.

Научный редактор - О.С.Кислюк.

Предисловие

Данная работа является третьей в цикле работ "Факультативный курс программирования". Курс рассчитан на учеников 7-8 классов при занятиях 3-4 часа в неделю. Предполагается, что первые полгода учащиеся занимаются основными понятиями программирования, а потом - достаточно независимо друг от друга - Бэйсиком, Паскалем и Фортраном. Таким образом, при изучении данной части курса предполагается, что учащиеся знакомы с понятиями переменной, цикла, ветвления и имеют навыки работы в какой-то системе программирования (по крайней мере в Е-практикуме).

По данному пособию автор занимался со школьниками на ЭВМ "Ямаха" MSX-I, используя программное обеспечение, разработанное в Лаборатории экспериментальной информатики ВЦ СО АН СССР г.Новосибирска (MSX-DOS 1.26, Сетевая версия 2).

Занятия со школьниками языком Паскаль продолжались около полугода. В конце этого срока учащиеся свободно работали в TURBO-Паскале, самостоятельно разрабатывая программы объемом в 30-40 операторов.

Описание языка в пособии является неполным. В конце работы приводится список литературы, к которой учащимся следует обращаться на разных этапах знакомства с языком Паскаль.

Автор приносит благодарность Кислюку О.С. за редактирование данного пособия.

1. Введение в Паскаль

1.1. Работа в системе TURBO-PASCAL. Редактор

Общие принципы работы системы

Наша работа на алгоритмическом языке Паскаль будет происходить в рамках системы TURBO-PASCAL. Эта система готова к работе после подготовительных операций на машине учителя, в результате чего на машинах учеников появляется номер машины в шестнадцатиричном виде и галочка, например 7> или B>.

В этот момент ученик должен набрать латинскими буквами "turbo" и нажать клавишу "RETURN".

Через некоторое время на экране появляется вопрос "INCLUDE ERROR MESSAGES (Y/N)?" (Нужно ли включать сообщения об ошибках?) Нужно ответить "Y", что будет соответствовать положительному ответу. Если ответить "N", то система тоже будет работоспособна, но работать с ней будет менее удобно. Система готова с Вами общаться, когда на экране вновь появляется галочка.

В этот момент на экране находится меню из команд:

Edit — редактировать
Compile — компилировать
Run — выполнить программу, находящуюся в памяти
Save — сохранить программу на диске
eXecute — выполнить программу, записанную на диске
Dir — просмотреть оглавление диска
Quite — закончить работу в системе
compiler Options — изменить характеристики компиляции

Выделенная буква является сокращением команды.

Нужно пояснить, что, в отличие от Бэйсика, программа на Паскале претерпевает изменения между написанием и выполнением. Сначала мы находимся в режиме редактирования и вводим текст программы в память машины. Затем выходим из режима редактирования. Исходный текст нашей программы (т.е. программа в том виде, как мы ее набрали) может быть сохранен на диске. Это полезно в тех случаях, если над программой мы собираемся работать несколько занятий. При записи на диск мы указываем имя. Информация на диске, имеющая имя, называется файлом.

Независимо от того, сохраним мы программу в файле на диске или нет, перед выполнением она должна быть откомпилирована (переведена во внутреннее представление). Результатом компиляции являются либо сообщения об ошибках, либо программа, готовая к выполнению. Откомпилированная программа может быть выполнена. Т.е. обычная последовательность команд при работе над программой — E — C — R; затем снова E. Однако можно пользоваться только командами E и R. В этом случае при выполнении команды R команда C вызывается автоматически.

Итак, для работы с программой нужно вначале выполнить функцию редактирования (нажать клавишу "E"). При первом выполнении этой функции система запросит имя файла. Предполается, что программа может находиться в файле на диске после прошлого сеанса работы; кроме того, под этим именем система предполагается сохранение нашей программы на диске в конце работы. Нужно ввести имя (без рас-

ширения ".PAS"). Тогда система поищет на диске файл <имя>.PAS и либо сообщит "NEW FILE", либо прочтет из него программу в память машины.

Редактирование

В любом случае после этого мы оказываемся в среде редактора. Перед нами чистый экран (если такого файла не было) или прочитанная программа (если он уже существовал). В верхней строке - информация о

- текущей строке (LINE);
- текущей позиции в строке (COL);
- режиме редактирования (INSERT или OVERWRITE);
- имени файла (INDENT).

Попробуйте перемещать курсор по экрану - Вы увидите, что числа в графах LINE и COL меняются.

Клавиша INS меняет режимы INSERT и OVERWRITE. Познакомьтесь с особенностями этих режимов.

Если в результате манипуляций с курсором Вы нечаянно "разрезали" строку, нажав "RETURN" в режиме INSERT в тот момент, когда курсор был в середине строки, можно "склеить" ее, нажав клавишу BS.

Еще две полезные команды

- удалить строку (CTRL/Y)
- вставить строку (CTRL/N)

Наберем простейшую программу

```
program p1;
begin
  write('ЭВМ приветствует Вас!')
end.
```

Легко понять, что это программа с именем p1, которая выводит на экран приветствие, указанное в апострофах в скобках после оператора WRITE.

Закончив редактирование, нужно нажать клавиши "HOME" и "D" (тот же эффект можно получить, нажав CTRL/K CTRL/D). На экране вновь появится галочка. И тогда следует выполнить команду RUN (нажать букву "r").

При этом сначала программа будет откомпилирована. Мы увидим некоторые сведения о компиляции, а потом либо сообщения об ошибках, либо результат работы правильной программы - фразу

ЭВМ приветствует Вас!

Если в программе есть ошибка, то мы увидим фразу вида

ERROR 1: ";" EXRECTED. PRESS <ESC>

Нужно нажать клавишу ESC, и тогда на экране появится текст программы, а курсор будет указывать возможное место ошибки.

Наиболее вероятные ошибки в нашей программе:

- одно из слов PROGRAM, BEGIN, WRITE, END написано неверно;
- в имени программы встретились русские буквы;
- после имени программы отсутствует точка с запятой;
- после оператора WRITE отсутствуют скобки (или одна из них);
- внутри скобок отсутствуют апострофы (или один из них);
- после оператора END отсутствует точка.

Ошибку нужно исправить, вновь выйти из редактора ("HOME" D или CTRL/K CTRL/D) и снова выполнить программу командой "r".

1.2. Структура программы. Простейший ввод-вывод

Рассмотренная нами ранее программа не содержала ни одной переменной. Она лишь выводила заранее определенную информацию при помощи оператора WRITE (оператора записи). Попробуем составить чуть более сложную программу: ввести с экрана два числа, а затем вычислить и сообщить их сумму.

Имена переменных и их типы должны быть описаны до начала выполнения программы. Простейшими (но не единственными) типами переменных являются целый (integer) и вещественный (real). С переменными обоих этих типов можно выполнять арифметические операции. Различие состоит в том, что переменные целого типа принимают целые значения (2, -5, 10000); вещественного типа - вещественные, или действительные значения (3.5, -2.8, 0.0015). Эти переменные по-разному хранятся в памяти машины; с ними по-разному выполняются арифметические операции. Пример описания переменных:

```
var
  a,b   : integer;
  x,y,z : real;
```

Ввод информации извне осуществляется при помощи оператора READ, после которого в скобках указываются имена переменных, куда попадет информация. Существуют разновидности операторов READ и WRITE, переводящие после выполнения курсор на новую строку: READLN и WRITELN.

Программа вычисления суммы двух чисел может быть записана так:

```
program p2;
  var a,b: integer;
begin
  writeln ('Введите 2 числа через пробел');
  readln (a,b);
  writeln ('Сумма=',a+b)
end.
```

ЗАДАНИЯ.

1. Посмотрите, как изменится выполнение программы p2, если вместо операторов READLN и WRITELN пользоваться операторами READ и WRITE.
2. Усложните эту программу, вычисляя не только сумму, но и разность, произведение, частное двух чисел.
3. Проверьте, можно ли с помощью такой программы ввести букву или слово.

1.2. Оператор присваивания. Переменные числовых типов. Арифметические выражения.

Здесь мы рассматриваем только переменные числовых типов (целые, вещественные) и арифметические выражения.

Для выполнения арифметических действий нам понадобятся знаки операций. В Паскале используются

"+"	для сложения;
"-"	для вычитания;
"*"	для умножения;
"/"	для деления;
"div"	для деления целых чисел;
"mod"	для нахождения остатка;
"sqr( )"	для возведения в квадрат;
"sqrt( )"	для извлечения квадратного корня.

В программе могут использоваться не только переменные, но и константы. Они описываются перед переменными следующим образом:

```
const
  pi = 3.1415926;
  g  = 9.8;
```

Пример вычислительной задачи:

```
program gipotenuza;
  const pi=3.1415926;
  var a,b,c: real;
```

```
begin
  writeln ('Введите длины катетов');
  read(a,b);
  c:=sqrt(sqr(a)+sqr(b));
  writeln ('Длина гипотенузы=',c);
  writeln ('Длина описанной окружности=',pi*c)
end.
```

ЗАДАНИЯ.

1. Что будет изображено при выполнении следующей программы:

```
program sr;
  var a,b,c,d: real;
begin
  a:=10;b:=15;c:=20;
  d:=(a+b+c)/3;
  writeln ('Среднее арифметическое=',d)
end.
```

2. Ввести два числа и вычислить их среднее арифметическое и среднее геометрическое.
3. Ввести радиус и вычислить длину окружности, площадь круга, площадь поверхности сферы, объем шара.
4. Ввести плотность вещества и размеры. Вычислить объем тела, имеющего форму параллелепипеда, и его массу.
5. Ввести высоту. Вычислить скорость тела при падении с такой высоты.
6. Ввести начальную скорость, ускорение, время. Вычислить путь, пройденный телом при равноускоренном движении.
7. Ввести длину стороны правильного треугольника. Вычислить его площадь; площадь поверхности тетраэдра; объем тетраэдра. Напомним, что тетраэдром называется тело, имеющее 4 грани, каждая из которых - правильный треугольник. (Для вычисления высоты тетраэдра заметьте, что она является высотой в треугольнике, где одна сторона - ребро тетраэдра, две другие - медианы боковых граней; объем тетраэдра вычисляется по формуле 1/3*S*H, где S - площадь основания, H - высота тетраэдра.)
8. Ввести 3 числа. Вычислить по теореме Герона площадь треугольника с такими сторонами.
9. Ввести 3 числа - коэффициенты квадратного уравнения, имеющего корни. Вычислить корни квадратного уравнения.
10. Средняя плотность Земли 5,52 г/см^3, средний диаметр - 12,8

тыс. км. Диаметр Марса составляет 0,53 диаметра Земли, а масса – 0,108 массы Земли. Найти среднюю плотность Марса.

1.4. Базовые типы и стандартные функции

Целый тип

Переменные этого типа описываются:

```
var a,b:integer;
```

Они принимают целые значение в диапазоне

$-maxint \le N \le maxint$;

конкретное число maxint зависит от реализации.

Для переменных целого типа определены операции

```
+
-
*
div
mod
```

Операция div соответствует целочисленному делению; операция mod – взятию остатка при целочисленном делении.

Для переменных целого типа определены функции ABS (абсолютная величина) и SQR (возведение в квадрат); результаты также являются целыми. Кроме того, определены функции SIN (синус), COS (косинус), ARCTAN (арктангенс), LN (логарифм натуральный), EXP (экспонента), SQRT (квадратный корень). Здесь результаты являются вещественными.

Все переменные целого типа упорядочены. Для них существуют функции PRED (предыдущий) и SUCC (последующий). Так, PRED(5)=4, SUCC(5)=6.

Для целых переменных определена функция ODD. Она принимает значение TRUE для нечетных переменных и FALSE для четных. Так, ODD(4)=FALSE, ODD(11)=TRUE.

Вещественный тип

Переменные этого типа описываются:

```
var a,b:real;
```

Для переменных вещественного типа определены операции

```
+
-
*
/
```

Для этих переменных определены функции ABS (абсолютная величина), SQR (возведение в квадрат), SIN (синус), COS (косинус), ARCTAN (арктангенс), LN (логарифм натуральный), EXP (экспонента), SQRT (квадратный корень). Результаты являются вещественными.

Кроме того, для переменных вещественного типа определены функции, выдающие результат округления: TRUNC – отбрасывание целой части, ROUND – округление до ближайшего целого. Результатом здесь являются целые числа.

Булевский тип

Существуют также логические (или булевские) переменные, которые могут принимать два значения – TRUE (истина) или FALSE (ложь). Они описываются следующим образом:

```
var l1,l2:boolean;
```

Они могут принимать значения операции сравнения переменных числовых (и других) типов:

```
l1:=b*b-4*a*c > 0
l2:=a <> 5
```

При этом допустимы знаки операции сравнения: >, <, >=, <=, =, <>. Над булевскими переменными определены операции AND (и), OR (или), NOT (не). Результаты этих операций вычисляются по правилам:

a	b	not a	a and b	a or b
true	true	false	true	true
true	false	false	false	true
false	true	true	false	true
false	false	true	false	false

Булевское выражение может содержать несколько операций. При этом оно расшифровывается в следующем порядке:

- сначала определяются значения всех операций сравнения;
- потом вычисляются все функции NOT;
- затем вычисляются все функции AND;
- в последнюю очередь вычисляются функции OR.

Выражение может содержать скобки, которые влияют на приоритетность вычисления значений операций.

Пример:

```
        a:=5;
        not (2 > 3) and (a=8) or (3-6>0 or 12/2>4)
            false       false      false      true
        true                                true
                    false
                              true
```

Значения булевских переменных упорядочены; считается, что FALSE < TRUE. Поэтому к ним можно применять функции PRED и SUCC; при этом PRED(TRUE)=FALSE, SUCC(FALSE)=TRUE.

Кроме того, к переменным булевского типа применима функция ORD, которая выдает значение упорядоченных переменных в их общем списке. В нашем случае ORD(false)=0, ORD(true)=1.

Символьный тип

К базовым типам относится также тип CHAR. Переменные этого типа могут принимать значение, равное любому символу на клавиатуре ЭВМ - буквы, цифры, специальные знаки.

Переменные этого типа можно сравнивать между собой в операциях сравнения:

```
var c1, c2: char;
    l1, l2: boolean;

c1:='a';
c2:='8';
l1:=c1=c2;
l2:=c1<>c2;
```

Символьные переменные упорядочены, образуя некоторый общий список; при этом английские символы упорядочены по алфавиту. Для символьных переменных используются функции PRED и SUCC; при этом PRED('b')='a', SUCC('a')='b'. Кроме того, к символьным переменным применима функция ORD - выяснение номера конкретного символа в этом списке. Так, ORD('1')=49, ORD('A')=65, ORD('a')=97.

Обратной по отношению к ORD является функция CHR. Она выдает значение символа по его номеру в списке. Так, CHR(49)='1'.

В Н И М А Н И Е - в операторах присваивания тип переменной слева от знака присваивания и тип выражения справа должны совпадать. Единственное исключение - переменной вещественного типа

можно присваивать значение выражения целого типа.

ЗАДАНИЯ.

Что будет напечатано в результате выполнения следующих программ:

1.
```
program what1;
begin
  writeln(sqrt(64)-sqr(4)>0)
end.
```

2.
```
program what2;
begin
  writeln((2=2)or(sqrt(625)-sqr(5)>0)and(pred(5)>succ(4)))
end.
```

3.
```
program what3;
begin
  writeln(((2=2)or(sqrt(625)-sqr(5)>0))and(pred(5)>succ(4)))
end.
```

4.
```
program what4;
  var a:real;
begin
  writeln('Введите число');
  readln(a);
  writeln(sin(a)>1.5)
end.
```

5.
```
program what5;
  var a:real;
begin
  writeln('Введите число');
  readln(a);
  writeln(sqr(sin(a))+sqr(cos(a))=2)
end.
```

6.
```
program what6;
  var a:integer;
begin
  writeln('Введите целое число');
  readln(a);
  writeln(odd(a+succ(a)))
```

```
end.
```

7.
```
program what7;
  var a:integer;
begin
  writeln('Введите целое число');
  readln(a);
  writeln(round((a+succ(a)) div 2)-a)
end.
```

8.
```
program what8;
  var a:real;
begin
  writeln('Введите число');
  readln(a);
  writeln(round(sqr(sin(a))+sqr(cos(a))))
end.
```

9.
```
program what9;
  var a,b:real;
begin
  writeln('Введите два числа через пробел');
  readln(a,b);
  writeln(succ((a+b)/2<sqrt(a*b)))
end.
```

10.
```
program what10;
  var a:integer;
begin
  writeln('Введите целое число');
  readln(a);
  writeln(succ((a*(a+1)*(a+2) mod 6 <>0)))
end.
```

2. Управляющие конструкции

2.1. Цикл с параметром

Паскаль допускает два варианта цикла с параметром: один – в случае возрастания параметра, второй – в случае убывания.

Задача. Напечатать четные числа от 2 до 20:

```
program chet1;
  var k:integer;
begin
  for k:=1 to 10 do
    writeln(2*k)
end.
```

Задача. Напечатать четные числа от 20 до 2:

```
program chet2;
  var k:integer;
begin
  for k:=10 downto 1 do
    writeln(2*k)
end.
```

Замечание 1. В отличие от многих языков программирования, не допускается шаг изменения параметра, отличный от 1.

Замечание 2. После DO может находиться лишь один оператор. Если есть необходимость выполнить в цикле несколько операторов, то они должны быть взяты в "операторные скобки" BEGIN-END:

```
program chet3;
  var k,l:integer;
begin
  for k:=1 to 10 do
    begin
      l:=2*k;
      writeln(l)
    end
end.
```

Такая конструкция - несколько операторов, взятых в "скобки" BEGIN-END - называется составным оператором. Она имеет все права обыкновенного оператора, т.е. может появляться везде, где по синтаксису необходим оператор (в операторе ветвления после THEN и ELSE, в операторе WHILE после DO и т.п.)

Рассмотрим некоторые возможности работы с экраном. Его можно чистить в начале работы оператором CLRSCR (либо WRITE(#12)). Кроме того, перед оператором WRITE можно устанавливать курсор в

любое нужное место экрана оператором GOTOXY(a,b). При этом параметр a соответствует x-координате и может меняться от 1 до 80, b - y-координате и меняется от 1 до 21. Так, в результате работы программы

```
program zv;
begin
  clrscr;
  gotoxy(1,1);write('*');
  gotoxy(80,1);write('*');
  gotoxy(1,21);write('*');
  gotoxy(80,21);write('*')
end.
```

будут напечатаны 4 звездочки по углам экрана.

Провести диагональ из левого верхнего угла экрана в правый нижний можно, например, таким образом:

```
program dia;
  var a:integer;
begin
  clrscr;
  for a:=1 to 20 do
    begin
      gotoxy(a*4,a);write('*')
    end
end.
```

ЗАДАНИЯ.

1. Что будет изображено в результате работы следующей программы:

```
program what;
  var a:integer;
begin
  clrscr;
  for a:=1 to 20 do
    begin
      gotoxy(83-a*4,a);write('*')
    end
end.
```

2. Нарисовать рамку из звездочек по краям экрана.

3. Нарисовать "снежинку" - прямой и косой крестик из звездочек во весь экран.
4. Нарисовать контур ромба из звездочек во весь экран.
5. Нарисовать равнобедренный треугольник из звездочек во весь экран.
6. Подсчитать сумму чисел от 1 до 100.
7. Факториалом целого числа называется произведение всех целых чисел, не превосходящих его: $n!=1*2*3...*(n-1)*n$. Вычислить факториал введенного числа.
8. Проверить справедливость формулы $H=gt^2/2$, вычислив мгновенную скорость в каждую секунду падения по формуле $v=v0+gt$ и подсчитав высоту падения по формуле $s=vt$ в течение 50 сек.
9. Построить параболу из звездочек.
10. Построить синусоиду из звездочек.

2.2. Ветвления

Оператор ветвления в Паскале выглядит традиционно:

```
if <условие>
  then <оператор1>
  else <оператор2>
```

Как обычно, часть ELSE может отсутствовать.
Под "условием" здесь понимается логическая переменная (типа boolean) или булевское выражение. После THEN и ELSE может находиться любой оператор, в том числе составной оператор или другой оператор ветвления. Если при расшифровке вложенного оператора ветвления возникают трудности, то нужно помнить, что ELSE всегда относится к БЛИЖАЙШЕМУ незакрытому THEN:

```
if dlina>10
  then if shirina<5
         then writeln('длинный и узкий')
         else writeln('длинный и широкий')
  else if shirina>5
         then writeln('короткий и широкий')
         else writeln('короткий и узкий')
```

При dlina=8, shirina=4 будет напечатано "короткий и узкий", а при dlina=12, shirina=7 - "длинный и широкий".

Важным примером функции, используемой в условиях, является функция EOF (End Of File). Она принимает значение "истина", когда при вводе данных встречается конец. Достигается это путем пустого ввода – нажатия клавиши "RETURN" без ввода данных. Пример использования функции EOF:

```
read(a);
if eof
   then writeln('Не введено ни одного числа')
   else writeln('Введено одно число');
```

ЗАДАНИЯ.

1. Что будет изображено в результате работы следующей программы:

```
program what;
  var a:integer;
begin
  read(a);
  a:=pred(a)+succ(a);
  if odd(a)
    then writeln('каракатица')
    else writeln('осьминог')
end.
```

2. Имеются 7 чисел – результаты измерений температуры за неделю. Холодными днями считаются дни с отрицательной температурой; теплыми – с температурой от 0 до 18; жаркими – выше 18. Подсчитать количество холодных, теплых, жарких дней за неделю.
3. Заполнить экран следующими символами: если сумма координат X и Y является четной, то символом "0", нечетной – "1".
4. Поставьте точку в середине экрана. Потом командами "Ю", "С", "З", "В" передвигайте точку по экрану следующим образом: по команде "Ю" передвигайте точку на юг, т.е. увеличивайте Y-координату, по команде "С" – на север и т.п.
5. Ввести 3 числа. Проверить, могут ли они быть длинами сторон прямоугольного треугольника.
6. Ввести 3 числа. Если отрезки с такими длинами образуют треугольник, то напечатать его площадь, иначе напечатать "Это не треугольник".
7. Ввести 2 числа и напечатать меньшее из них.
8. Ввести 3 числа и напечатать их в порядке возрастания.
9. Ввести 5 чисел. Напечатать "Все нечетные" либо "Имеются чет-

ные" в зависимости от четности введенных чисел.

Возможное решение:

```
program proverka;
  var a,k:integer; l:boolean;
begin
  l:=true;
  for k:=1 to 5 do
    begin
      read (a);
      l:=l and odd(a)
    end;
  writeln;
  if l
     then writeln('Все нечетные')
     else writeln('Имеются четные')
end.
```

Найдите решение, которое не требует введения булевской переменной.

10. Ввести 5 чисел. Напечатать "Все четные" либо "Имеются нечетные" в зависимости от четности введенных чисел.

2.3. Циклы с пред- и пост-условием

Цикл с параметром используется тогда, когда необходимо организовать заранее известное число повторений части программы. Если же число повторений заранее не известно, то используют другие циклы - повторение по условию. В Паскале существует 2 разновидности таких циклов: цикл с предусловием

```
while <условие> do
  <оператор>;
```

и цикл с пост-условием

```
repeat
  <оператор1>;
   ...
  <операторN>
until <условие>;
```

При выполнении цикла с предусловием сначала происходит проверка условия. Если выражение, используемое в качестве условия, принимает значение TRUE, то выполняется оператор (который, напомним, может быть и составным оператором, т.е. последовательностью операторов, заключенных в скобки BEGIN-END). Затем вновь проверяется условие, и т.д. Выполнение цикла заканчивается, когда выражение в условии принимает значение FALSE. Если первая же проверка дает такое значение, то оператор не выполняется ни разу.

При выполнении цикла с постусловием сначала выполняются операторы, записанные между REPEAT и UNTIL, а затем происходит проверка условия. Если условие ложно, происходит возврат к выполнению последовательности <оператор 1>; ...; <оператор N>. Если условие истинно, цикл считается выполненным. Заметим, что, в отличие от конструкции WHILE, если условие истинно с самого начала, то один раз операторы между REPEAT и UNTIL все же выполняются.

Приведем пример использования этих циклов при решении задачи: напечатать все степени двойки, не превосходящие 1000:

```
program step1;
  var a:integer;
begin
  a:=1;
  while a<1000 do
    begin
      writeln(a);
      a:=a*2;
    end
end.

program step2;
  var a:integer;
begin
  a:=1;
  repeat
    writeln(a);
    a:=a*2
  until a<1000
end.
```

При использовании этих конструкций нужно внимательно следить, чтобы условие, указанное после WHILE или UNTIL, обязательно достигалось. Не рекомендуется использовать проверки на равенство

двух вещественных чисел.

Проверьте, что произойдет при выполнении программы

```
program one;
  var a,d:real;
begin
  a:=0; d:=0.1;
  while a<>1   do
    begin
      a:=a+d;
      writeln(a);
    end
end.
```

Ситуация, возникающая при выполнении этой программы, называется зацикливанием - невозможностью завершения цикла (поскольку условие никогда не выполняется). Чтобы прекратить выполнение такой программы, нужно нажать CTRL/S, а затем CTRL/STOP.

Очень часто в качестве условия окончания цикла используется функция EOF. Приведенная здесь программа вводит с экрана несколько (заранее неизвестно, сколько) чисел и печатает их количество и сумму. При вводе после каждого числа следует нажимать "RETURN"; признак конца ввода - нажатие "RETURN" два раза подряд.

```
program summa;
  var a,s:real; n:integer;
begin
  s:=0; n:=0;
  while not eof do
    begin
      readln (a);
      if not eof then
         begin
           s:=s+a;
           n:=n+1
         end
    end
  writeln('Введено ',n,' чисел');
  writeln('их сумма= ',s)
end.
```

ЗАДАНИЯ.

1. Что будет изображено в результате работы следующей программы:

```
program stepeni;
  var a:real; k:integer;
begin
  k:=0;a:=1;
  while a<maxint do
    begin
      k:=k+1;
      a:=a*2
    end;
  writeln(k,a)
end.
```

2. Вычислить и напечатать среднее арифметическое введенных чисел.
3. Из нескольких введенных чисел выбрать наименьшее.
4. Подсчитать количество четных и нечетных чисел во введенных с экрана числах.
5. Вводить с экрана числа до тех пор, пока сумма их не превысит 100. Количество введенных чисел напечатать.
6. Вводить с экрана символы до тех пор, пока не будут введены 4 буквы 'a'.
7. Ввести число. Напечатать все числа, квадрат которых не превосходит введенного числа.
8. Вводить с экрана возрастающие целые числа до тех пор, пока не будет нарушен порядок, т.е. пока следующее число не окажется меньше предыдущего.
9. С клавиатуры вводятся первые буквы названий животных в живом уголке. Известно, что в уголке есть собака, медвежонок, лиса (4 ноги), голубь, ворона (2 ноги), жук (6 ног), паук (8 ног), рак и краб (10 ног), причем некоторых животных имеется по нескольку представителей. Подсчитать количество ног у введенных животных.
10. Запрограммируйте следующую игру: машина выбирает случайным образом число от 1 до 100 (a:=random(100)). Играющий вводит числа, а машина сообщает "Задуманное число больше", "Задуманное число меньше" или "Вы угадали". Игра заканчивается, когда число угадано или количество неудачных попыток достигло 7.

2.4. Конструкция выбора

Когда в программе нужно предусмотреть обработку нескольких альтернативных возможностей, применяют конструкцию выбора:

```
case N of
  1: <оператор>;
  2: <оператор>;
  . . .
  M: <оператор>
end;
```

При этом допускается и такая запись:

```
case N of
  1,2,3: <оператор>;
  . . .
  M,K  : <оператор>
end;
```

(т.е. в двух или нескольких случаях нужно выполнять одно и то же действие).

Если в некоторых случаях нужно выполнить несколько операторов, их заключают в скобки BEGIN-END, образуя составной оператор.

Вообще говоря, в операторе CASE может встречаться не только переменная целого типа. Но такой вид оператора мы сейчас рассматривать не будем.

Задача. Среди символов, вводимых с клавиатуры, подсчитать количество символов 'a', 'b', 'c'.

```
program simvoly;
  var k0,k1,k2,n:integer;
      a         :char;
begin
  k0:=0; k1:=0; k2:=0;
  while not eof do
    begin
      readln (a);
      if not eof then
         begin
           n:=ord(a)-ord('a');
```

```
            case n of
              0: k0:=k0+1;
              1: k1:=k1+1;
              2: k2:=k2+1;
            end
          end
      end;
    writeln('Символов "a" - ',k0);
    writeln('Символов "b" - ',k1);
    writeln('Символов "c" - ',k2)
  end.
```

Во многих задачах бывает полезно управление каким-либо объектом на экране с помощью стрелок. Для этого нужно уметь опрашивать клавиатуру и осуществлять ветвление на 4 возможности в зависимости от кода нажатой стрелки. Эти коды такие: стрелка вправо - 28, влево - 29, вверх - 30, вниз - 31. Чтение специальных символов, которое трудно осуществить при помощи обычных операторов READ или READLN, может быть осуществлено при помощи чтения из буфера клавиатуры; этому соответствует специальный оператор READ(KBD, ...).

Задача. Поставить звездочку в середине экрана. Стрелками перемещать звездочку по экрану, оставляя след перемещения.

```
program upravlenie;
 var x,y,n:integer;
     ch :char;
     eoj:boolean;
begin
 eoj:=false;
 clrscr;
 x:=40;y:=10;gotoxy(x,y);write('*');
 while not eoj do
 begin
  if keypressed then
   begin
    read(kbd,ch);
    if ord(ch)=13
     then eoj:=true
     else
       begin
         n:=ord(ch)-27;
         case n of
```

```
                1: x:=x+1;
                2: x:=x-1;
                3: y:=y-1;
                4: y:=y+1
              end;  {case}
              gotoxy(x,y);write('*');
            end     {else}
        end
      end
    end.
```

ЗАДАНИЯ.

1. Что будет изображено в результате работы следующей программы:

```
program cifry;
  var k1,k2,k3,n,m:integer;
begin
  m:=1989;
  k1:=0; k2:=0; k3:=0;
  while m<>0 do
    begin
      n:=m mod 10;m:=m div 10;
      n:=n-6;
      case n of
        1: k1:=k1+1;
        2: k2:=k2+1;
        3: k3:=k3+1;
      end
    end;
  writeln(k1,k2,k3)
end.
```

2. Измените программу UPRAVLENIE так, чтобы звездочка не могла выйти за пределы экрана.
3. Среди введенных с экрана чисел подсчитать количество тех, которые делятся на три; при делении на 3 дают остаток 1; остаток 2.
4. Написать программу, которая вводит прилагательное в единственном числе, определяет его род и печатает во всех падежах. Ветвление на 3 возможности осуществить с помощью оператора CASE.
5. Имеется устройство, которое понимает команды З, В, Ю, С и дви-

гается по этим командам на запад, восток, юг, север на 1 единицу длины. Ввести несколько таких команд и вычислить, на сколько сместится устройство относительно начального положения при таком движении.

3. Сложные типы и организация данных

3.1. Строки

Для удобства работы с текстовой информацией в Паскале есть возможность объявить переменную СТРОКОЙ определенной размерности:

```
var a:string[10];
```

Со строками можно делать следующее:

1. Присваивать одной строке значения другой. При этом, если делается попытка присвоить более длинную строку более короткой, то происходит "обрезание" правых символов:

```
var a:string[10];
    b:string[3];
. . .
a:='пароход';
b:=a;
```

В этом случае b получит значение 'пар'.

2. Приписывать друг к другу:

```
a:='лед';
b:='о';
c:='ход';
d:=a+b+c;        {переменная d получила значение 'ледоход'}
```

3. Выделять элемент строки:

```
a:='самолет';
b:=a[1];         {переменная b получила значение 'с'}
```

4. Определять длину строки:

```
a:='клавиатура';
k:=length(a);   {переменная k получила значение 10}
```

5. Сравнивать между собой две строки в качестве условия:

```
a:='мир';
b:='mir';
if a=b then ... else ...
```

Строка из одного символа эквивалентна переменной типа char. Их можно сравнивать; можно выполнять операторы присваивания, где такие переменные стоят по разные стороны от знака присваивания.

ЗАДАНИЯ.

1. Что будет изображено в результате работы следующей программы:

```
program what;
  var a,b:string[20];
        k:integer;
begin
  a:='головоломка';b:='';
  for k:=1 to lenght(a) do
    if a[k]<>'o'
      then b:=b+a[k]
      else b:=b+'a';
   writeln(b)
end.
```

2. Ввести строку и напечатать ее первый символ; ее последний символ.
3. Ввести слово и напечатать его по одной букве в строке.
4. Ввести слово. Напечатать его в обратном порядке.
5. Ввести строку и подсчитать количество букв 'а' в ней.
6. Ввести слово и подсчитать количество гласных букв в нем.
7. Ввести три слова и напечатать слово максимальной длины.
8. Ввести сложное слово, состоящее из двух корней одинаковой длины. Напечатать отдельно каждый корень.
9. Ввести слово. Напечатать сначала все буквы, стоящие на четных местах, потом - на нечетных.
10. Вводить по два слова и определять, составляют ли они рифму. Считается, что два слова образуют рифму, если все буквы, начиная от последней гласной буквы в слове, совпадают.

3.2. Массивы

Массивом называется такой способ хранения однотипных данных, когда к элементу возможен доступ по номеру. При описании массива указывается его имя, тип элементов и размерность:

```
var x:array[1..80] of integer;
```

При использовании элемента массива указывается имя массива и номер элемента:

```
a:=x[15];
```

Если нам нужно задать массив констант, мы можем сделать это следующим образом:

```
const dm:array[1..12] of integer=(31,28,31,30,
  31,30,31,31,30,31,30,31,30,31,30,31);
```

Обычно для одной и той же обработки всех элементов массива используется цикл с параметром.

Задача. В массивах A[1..5] и B[1..5] содержится длина и ширина каждого из пяти прямоугольников. Вычислить площади прямоугольников.

```
program sq;
  const a:array[1..5] of real=(3.5,4.7,7.6,6.9,12.7);
        b:array[1..5] of real=(8.5,9.7,3.6,2.9,18.7);
  var s:real;
      k:integer;
begin
  for k:=1 to 5 do
    begin
      s:=a[k]*b[k];
      writeln(s)
    end
end.
```

ЗАДАНИЯ.

1. Что будет результатом работы следующей программы:

```
program kvadraty;
  var a,b:array[1..5] of integer;
      s,k:integer;
begin
  for k:=1 to 5 do
    a[k]:=k;
  for k:=1 to 5 do
    writeln(sqr(a[k]))
end.
```

2. В массивах X1[1..4], Y1[1..4] находятся координаты левых концов отрезков, в массивах X2[1..4], Y2[1..4] – правых. Найти длины четырех отрезков.
3. В массивах X[1..6], Y[1..6] находятся координаты шести точек. Каждую из шести точек можно соединить с любыми пятью; при этом получится 6*5=30 отрезков. Из них различных будет только 15 (понятно, что отрезок от второй точки до третьей совпадает с отрезком от третьей точки до второй). Найти длины этих 15 отрезков.
4. В массиве CH[1..8] находятся 8 символов. Найти, у какого символа номер минимален (номер – функция ORD).
5. В массиве GL[1..9] находятся гласные буквы. Для каждого вводимого слова подсчитать количество гласных букв в нем.
6. В массиве SI[1..5] находятся 5 символов. Преобразовать вводимые слова следующим образом: каждый из этих символов удваивается, остальные остаются без изменения. (Например, если в массиве SI находятся символы 'а', 'в', 'в', 'г', 'д', то слово 'вагон' должно превратиться в 'ввааггон'.)
7. В массиве A[1..10] находится рост 10 девочек, в массиве B[1..10] – 10 мальчиков. Определить, кто в среднем выше – девочки или мальчики.
8. В массиве A[1..20] находится 20 целых чисел. Определить, каких чисел больше – четных или нечетных.
9. В массиве A[1..10] находится 10 целых чисел. Определить, имеется ли среди них хотя бы одно четное число, используя оператор IF только один раз.
10. В массиве A[1..10] находится 10 целых чисел. Определить, имеется ли среди них хотя бы одно нечетное число, используя оператор IF только один раз.

3.4. Перечисление. Интервал

Переменные целого, вещественного, логического, символьного типа можно встретить во всех языках программирования. Однако есть типы, характерные именно для Паскаля. К ним относится, например, тип перечисление (другое название - скалярный тип). Переменные этого типа могут принимать только значения, явно перечисленные при указании этого типа, например:

```
type day=(mondey, tuesday,wednesday,thursday,
          friday,saturday,sunday);
var weekday:day;
```

или

```
type pmonth=(28,29,30,31);
var kday:pmonth;
```

Переменным такого типа

1) можно присваивать значения из указанного списка:

```
weekday:=friday;
kday:=30;
```

2) можно отношения, использующие такие переменные, использовать в качестве условий в операторе ветвления:

```
if wday=sunday
  then ...     ;
```

3) можно использовать номер элемента в списке; нумерация начинается с нуля. Например, при выполнении оператора

```
writeln(ord(monday));
```

будет напечатан 0.

4) значения, содержащиеся в списке, можно использовать в качестве границ в операторе цикла:

```
for d:=monday to friday do ...
(если переменная d имеет тип day)
```

Другой тип данных, характерный для Паскаля - ограниченный,

или интервальный тип. Этим типом пользуются тогда, когда заранее по смыслу задачи известен диапазон изменения переменных. При этом в качестве границ могут быть указаны константы целого, символьного либо скалярного типа.
Примеры:

```
type nday=1..31;
type workday=monday..friday;
type ball=1..5;
type let='a'..'k';
```

Преимущество использования такого типа данных - дополнительный контроль. Конечно, наши программы не перестанут работать, если мы используем вместо описанных здесь типов тип INTEGER вместо NDAY и BALL, тип CHAR вместо LET и тип DAY вместо WORKDAY. Более того, могут возникнуть сложности в операторах ввода-вывода для переменных таких типов. Но использование ограниченного типа позволит нам уменьшить число ошибок в программе.

ЗАДАНИЯ.

1. Что будет изображено при работе следующей программы (в нужном месте подставьте свой номер машины):

```
program period;
  const dm:array[0..11] of integer=(31,28,31,30,
    31,30,31,31,30,31,30,31,30,31,30,31);

  type month=(january,february,march,april,may,june,july,
              august,september,october,november,december);

  var mo:month;
      n,s:integer;

begin
  n:=        {номер машины};
  s:=0;
  case n of
    1: for mo:=january to march do
       s:=s+dm[ord(mo)];
    2: for mo:=february to june do
       s:=s+dm[ord(mo)];
    3: for mo:=march to october do
```

```
        s:=s+dm[ord(mo)];
      4: for mo:=april to november do
        s:=s+dm[ord(mo)];
      5: for mo:=march to september do
        s:=s+dm[ord(mo)];
      6: for mo:=february to july do
        s:=s+dm[ord(mo)];
      7: for mo:=january to august do
        s:=s+dm[ord(mo)];
      8: for mo:=february to december do
        s:=s+dm[ord(mo)];
      9: for mo:=june to november do
        s:=s+dm[ord(mo)];
      10: for mo:=may to october do
        s:=s+dm[ord(mo)];
      11: for mo:=march to august do
        s:=s+dm[ord(mo)];
      12: for mo:=april to october do
        s:=s+dm[ord(mo)];
      13: for mo:=january to may do
        s:=s+dm[ord(mo)];
      14: for mo:=march to november do
        s:=s+dm[ord(mo)];
      15: for mo:=april to september do
        s:=s+dm[ord(mo)]
    end;
    writeln(n);
    writeln('В указанном периоде ',s,' дней');
  end.
```

2. Подсчитайте длину рабочей недели с использованием типов ДЕНЬ_НЕДЕЛИ (перечисление) и РАБОЧИЙ_ДЕНЬ (интервал).
3. Подсчитайте длину летних школьных каникул с использованием констант, типов и переменных из задания 1.

4. Технология программирования

4.1. Процедуры

При разработке сложных программ в Паскале, как и в других языках, принято использовать самостоятельные программные единицы. В Паскале это процедуры и функции.

Примеры описания процедур – без параметров; только с входными параметрами; с входными и выходными параметрами:

```
procedure kvadrat;
procedure kvadr(x,y:integer,a:real);
procedure kv(x,y:integer;a:real;var u,t:integer;b:real);
```

Служебное слово VAR стоит перед теми переменными, которые могут изменяться в процессе выполнения процедуры – перед выходными параметрами.

Обращения к этим процедурам и функциям могут выглядеть так:

```
kvadrat;
kvadr(3,4,2.5);
kv(10,20,3.14,k,l,s);
```

В отличие от Е-практикума, процедуры и функции в Паскале описываются до их использования.

Программа с использованием процедуры имеет такую структуру:

```
program qq;
  var . . .

  procedure tt(a,b:integer;var c:integer);
    var . . .
  begin  {tt}
      . . .
  end;   {tt}

begin {qq}
  . . .
  tt(10,20,x);
  write(x)
  . . .
end.  {qq}
```

Задача. Нарисовать детскую пирамидку из четырех прямоугольников. Оформим рисование одного прямоугольника в виде процедуры с четырьмя входными параметрами – координатами двух противоположных углов прямоугольника.

```
program piramida;

  procedure pr(x1,y1,x2,y2:integer);
    var w,k,l:integer;

    begin
      if x2<x1 then begin w:=x1;x1:=x2;x2:=w end;
      if y2<y1 then begin w:=y1;y1:=y2;y2:=w end;
      for k:=x1 to x2 do
        for l:=y1 to y2 do
          begin
            gotoxy(k,l);write('■')
          end
    end;      {pr}

  begin  {piramida}
    clrscr;
    pr(36,2,44,7);
    pr(30,8,50,11);
    pr(20,12,60,15);
    pr(10,16,70,20)
  end.
```

Заметим, что переменные w,k,l описаны в процедуре pr и недоступны для использования за ее пределами (это локальные переменные). В то же время переменные, описанные в головной программе (глобальные переменные), доступны для использования внутри процедуры даже в тех случаях, когда их значения не передаются через параметры.

Задача. Нарисовать 4 треугольника из звездочек и перенумеровать их:

```
program risunok1;
  var k:integer;

  procedure treug;
  begin
    writeln('  *');
    writeln(' ***');
    writeln('*****');
    writeln('  ',k);
    writeln
  end;    {treug}
```

```
begin {risunok1}
  clrscr;
  for k:=1 to 4 do
    treug;
end.   {risunok1}
```

Переменная k здесь получает значение вне процедуры, а печатается внутри процедуры. Под треугольниками из трех рядов звездочек будут напечатаны их номера - 1,2,3,4.

В Н И М А Н И Е - если мы внутри процедуры опишем переменную с тем же именем, что и в головной программе, то реально использоваться будут две переменные - одна внутри процедуры, а другая вне ее.

```
program risunok2;
  var k:integer;

  procedure treug;
    var k:integer;
  begin
    writeln('  *');
    writeln(' ***');
    writeln('*****');
    k:=1;
    writeln('  ',k);
    writeln
  end;    {treug}

begin {risunok2}
  clrscr;
  for k:=1 to 4 do
    treug;
end.   {risunok2}
```

В результате выполнения программы risunok2 под всеми треугольниками будет стоять цифра "1". Но треугольников, тем не менее, будет ровно 4, поскольку значение параметра цикла в самой программе портиться не будет.

Приведем пример использования процедуры в программах вычислительного характера. Пусть для поступающих с экрана троек чисел

нужно определять площади треугольников, образованных такими сторонами; из всех площадей выбрать и напечатать максимальную.

```
program p1;
  var a,b,c,s,sm:real;
      l       :boolean;

  procedure geron(var s:real);
    var p:real;
  begin  {geron}
    p:=(a+b+c)/2;
    s:=sqrt(p*(p-a)*(p-b)*(p-c));
  end;   {geron}

begin  {p1}
  sm:=0; readln(a,b,c);
  while not eof do
    begin
      l:=(a+b>c) and (a+c>b) and (b+c>a);
      if l then begin
                  geron (s);
                  if s>sm then sm:=s
                end
           else writeln('Это не треугольник');
      readln(a,b,c)
    end;   {while}
  writeln('Максимальная площадь=',sm)
end.   {p1}
```

Заметим, что переменные a,b,c,s,sm,l описаны в головной программе и поэтому доступны всем процедурам. Переменная p описана в процедуре GERON и является локальной переменной этой процедуры. Попытка использовать ее в другом месте программы вызовет ошибку. Переменная s для процедуры GERON является выходной, и поэтому ее значение доступно за пределами этой процедуры.

Процедуры могут быть вложенными. При этом к процедуре P2, описанной внутри процедуры P1, возможен доступ только из процедуры P1 и невозможен доступ из программы или из других процедур. Структура программы при этом становится такой:

```
program qq;
  var . . .

  procedure tt(a,b:integer;var c:integer);
    var . . .

    procedure pp;
      var . . .
    begin  {pp}
      . . .
    end;   {pp}

  begin  {tt}
    . . .
    pp;
    . . .
  end;   {tt}

begin  {qq}
  . . .
  tt(10,20,x);
  write(x)
  . . .
end.   {qq}
```

ЗАДАНИЯ.

1. Что будет изображено в результате работы следующей программы:

```
program goroda;

  procedure p1;

    procedure p2;
    begin
      writeln('Бологое')
    end;

  begin
    p2;
    writeln('Ленинград');
    p2
  end;
```

```
procedure p3;

  procedure p4;
  begin
    writeln('Вильнюс')
  end;

begin
  p4;
  writeln('Каунас');
  p4
end;

begin
  p1;
  p3
end.
```

2. Написать процедуру рисования контура прямоугольника с четырьмя входными параметрами – координатами противоположных вершин.
3. Написать процедуру рисования линии с четырьмя входными параметрами – координатами противоположных концов.
4. Написать процедуру рисования контура треугольника с шестью входными параметрами – координатами вершин.
5. Написать процедуру рисования закрашенного треугольника с шестью входными параметрами – координатами вершин (сложная задача).
6. Написать программу с использованием процедуры GIPOTENUZA, которая имеет два входных параметра – длины катетов – и один выходной – длину гипотенузы.
7. Написать программу с использованием процедуры DLINA, которая имеет четыре входных параметра – координаты двух точек – и один выходной – длину отрезка между ними.
8. Написать программу, которая запрашивает координаты трех вершин треугольника и выдает координаты оснований трех медиан. Используйте процедуру SEREDINA с четырьмя входными и двумя выходными параметрами (входные – координаты двух концов отрезка, выходные – координаты середины).
9. Напишите программу с использованием процедуры KORNI для решения корней квадратного уравнения. Используйте три входных параметра (коэффициенты) и три выходных (два корня и одна логи-

ческая переменная - имеются ли корни).

10. Напишите программу с использованием процедуры SHAR, у которой один входной параметр - радиус - и два выходных - площадь поверхности сферы и объем шара.

4.2. Функции

Одной из форм самостоятельных частей программы являются функции. Функция всегда принимает некоторое значение; обычно она зависит от некоторых параметров. Значение функции может быть разного типа: числового, логического, символьного.

Примеры описания функций:

```
function cub:real;
function cub2(a,b:integer):real;
function sim(a,b:char):char;
```

Примеры вызова этих функций:

```
a:=cub;
c:=cub2(3,4);
f:=sim('m','a');
```

Задача 1. Для тройки чисел определить их среднее арифметическое.

Такая задача может быть решена с помощью функции вещественного типа, которая имеет три входных параметра вещественного типа.

```
program srednee;
  var a,b,c : real;

    function m(a,b,c:real):real;
    begin
      m:=(a+b+c)/3;
    end;  {m}

begin {srednee}
  writeln(m(1,2,3));
  writeln(m(1.5,2,2.5))
end.  {srednee}
```

Задача 2. Для каждой введенной шестерки чисел установить, мо-

жет ли существовать треугольная пирамида с ребрами такой длины.

Обозначим верхние ребра a,b,c, нижние – x,y,z. Проверим, образуют ли треугольники тройки чисел

- a, b, c;
- b, c, y;
- a, c, z;
- x, y, z;

а также, превышает ли сумма площадей верхних треугольников площадь нижнего треугольника.

Введем в рассмотрение логическую функцию CHECK от трех вещественных аргументов, принимающую значение TRUE, когда числа могут быть длинами сторон треугольника, и FALSE в противном случае. Введем также вещественную функцию S от трех вещественных аргументов, равную площади соответствующего треугольника.

```
program tetra;
  var a,b,c,x,y,z : real;
      l           : boolean;

    function check(a,b,c:real):boolean;
    begin
      check:=(a+b>c) and (b+c>a) and (a+c>b)
    end; {check}

    function s(a,b,c:real):real;
      var p:real;
    begin
      p:=(a+b+c)/2;
      s:=sqrt(p*(p-a)*(p-b)*(p-c))
    end;  {s}

begin {tetra}
  while not eof do
    begin
      read(a,b,c,x,y,z);
      if not eof then
         begin
           l:=check(a,b,x) and check(b,c,y) and
              check(a,c,z) and check(x,y,z);
           if not l
              then writeln('Это не треугольники')
              else
```

```
                    if s(a,b,x)+s(b,c,y)+s(a,c,z)>s(x,y,z)
                       then writeln('Это пирамида')
                       else writeln('Это не пирамида')
            end   {if}
        end       {while}
    end.          {tetra}
```

В Паскале имеется ряд встроенных функций. К ним относится функция RANDOM(a), которая выдает случайное целое число от 0 до a. "Разогнать" датчик случайных чисел можно процедурой RANDOMIZE; она не имеет входных параметров.

ЗАДАНИЯ.

1. Что будет изображено в результате работы следующей программы:

```
program what;

  function sim(a,b:char):char;
  begin
    if ord(a)<ord(b)
      then sim:=a
      else sim:=b
  end;

begin
  writeln(sim('k','l'))
end.
```

2. Нарисовать звездное небо с помощью функции RANDOM.
3. Написать программу вычисления площади поверхности прямоугольного параллелепипеда по трем ребрам. Использовать функцию – вычисление площади прямоугольника – с двумя входными параметрами.
4. Напишите программу с использованием функции GIPOTENUZA, которая по двух входным параметрам – длинам катетов – вычисляет длину гипотенузы.
5. Напишите программу с использованием функции MINKOREN, которая по трем входным параметрам определяет минимальный корень квадратного уравнения.
6. Напишите программу с использованием функции KTOCHEK, которая определяет количество точек пересечения параболы с осью OX. Парабола задается уравнением $y=ax^2+bx+c$. Коэффициенты a, b, c

сделать входными параметрами процедуры.

7. Вводятся пары чисел, которые соответствуют (x,y)-координатам точек. После завершения ввода указать расстояние до максимально удаленной от начала координат точки. Использовать функцию RASST – расстояние от точки до начала координат – с двумя входными параметрами – координатами точки.
8. Вводятся пары чисел, которые соответствуют (x,y)-координатам точек. Потом все точки соединяются отрезками – каждая с каждой. Требуется определить отрезок максимальной длины. Использовать функцию DLINA с четырьмя входными параметрами – координатами двух точек.
9. Напишите программу с использованием функции S для вычисления пути при равноускоренном движении. Скорость, ускорение и время используйте как параметры.
10. С экрана вводятся слова. Требуется определить слово, в котором наибольшее число раз встретилась буква 'a'. Используйте функцию – количество букв 'a' в данном слове. Учтите, что нельзя передавать в качестве параметра переменные типа STRING, поэтому соответствующую переменную сделайте глобальной.

4.3. Рекурсия

Факториалом целого положительного числа n называется произведение целых чисел от 1 до n:

$$n! = 1*2*3*...*(n-1)*n$$

При вычислении этой функции можно поступить двумя способами.
Способ 1. В цикле n раз повторить одинаковое действие – уже накопленное к текущему моменту произведение умножить на очередное число.

```
program factorial1;
  var n,p,k:integer;

begin
  readln(n);
  p:=1;            {начальное значение произведения}
  for k:=1 to n do
    p:=p*k;        {произведение умножить на текущее число}
  writeln('n!=',p);
end.
```

Способ 2. Заметим, что

$$n! = n*(n-1)*...>3*2*1 = n*(n-1)! \text{ , если } n>1$$
$$n! = 1 \text{ , если } n=1$$

При этом мы вычисляет значение функции ФАКТОРИАЛ от n через значение этой функции от n-1. Поскольку значение функции от n=1 известно, это гарантирует нам, что последовательность таких действий не будет бесконечной. Напишем программу в соответствии с таким способом вычисления n!:

```
program factorial2;
{$A-}
  var n:integer;

  function fact(n:integer):integer;
    var k:integer;
  begin {function}
    if n=1
      then fact:=1
      else fact:=n*fact(n-1)
  end; {function}

begin {program}
  readln(n);
  writeln('n!=',fact(n))
end.
```

Такой способ действий, когда функция или процедура вызывает себя, но с другими параметрами, называется рекурсивным. Рекурсия допустима не во всех языках программирования. Она требует некоторых дополнительных ресурсов; способом запросить эти ресурсы является указание {$A-} после оператора PROGRAM. Другой способ вычисления подобных выражений (в нашем случае это способ 1) называется итерационным; он обеспечивается во всех языках программирования посредством цикла с параметром.

Разберем еще одну задачу с использованием этих двух подходов. Это задача определения суммы цифр многозначного числа. Заметим, что последняя цифра - это остаток числа от деления на 10 (число по модулю 10); число, образованное отбрасыванием последней цифры - это результат целочисленного деления первоначального числа на

10.

Способ 1 - итерационный - поочередное прибавление последней цифры:

```
program sum1;
 var n,s:integer;

begin
  readln(n);
  s:=n mod 10;      {последняя цифра исходного числа}
  while n div 10 <> 0 do
    begin
     n:=n div 10;  {отбрасываем последнюю цифру}
     s:=s+n mod 10 {прибавляем очередную правую цифру}
    end;
  writeln('s=',s)
end.
```

Способ 2 - рекурсивный. Замечаем, что сумма цифр числа - это последняя цифра + сумма цифр уменьшенного в 10 раз числа:

```
program sum2;
{$A-}
  var n:integer;

  function summa(n:integer):integer;

  begin
    if n div 10 = 0
      then summa:=n
      else summa:=summa(n div 10)+n mod 10
  end;

begin
  readln(n);
  writeln('s=',summa(n))
end.
```

Примером задачи, где рекурсия используется не для вычисления, является задача о ханойских башнях, разобранная во многих пособиях (см. [2]). Там решение задачи для башни из 64 дисков сводится к решению задачи для башни из 63 дисков.

ЗАДАНИЯ.

1. Что будет изображено в результате выполнения следующей программы:

```
program hanoy;
{$A-}
  var disks:integer;

  procedure moveb(kd,fromb,tob,workb:integer);

    procedure moved(n1,n2:integer);
    begin
      writeln(n1,'->',n2)
    end; {moved}

  begin {moveb}
    if kd>0
      then
        begin
          moveb(kd-1,fromb,workb,tob);
          moved(fromb,tob);
          moveb(kd-1,workb,tob,fromb)
        end
  end;  {moveb}

begin {hanoy}
  disks:=3;
  moveb(disks,1,3,2)
end.  {hanoy}
```

2. Написать программу вычисления целой степени числа с использованием итерационного и рекурсивного подходов.

5. Работа с внешними устройствами

5.1. Файлы

До сих пор мы писали такие программы, где переменные получали значения либо в результате операторов присваивания, либо путем ввода с клавиатуры. Но эти способы хороши только тогда, когда данных немного. Скажем, обработать список фамилий учеников одного класса так уже неудобно. Большие массивы данных хранятся на ма-

шинных носителях, сейчас все больше на магнитных носителях (времена перфокарт и перфолент проходят), в нашем случае - на магнитной дискете. При занесении данных на дискету они получают имя, например "CLASS8A". Группа данных на магнитном носителе, имеющая имя, называется файлом. Мы будем работать только с текстовыми файлами. Допустим, нам нужно разделить учащихся на две группы; фамилии первой группы нужно записать на диск под именем "GRUPPA1". Для работы с файлами необходимо:

1. Объявить, что в программе используется один или несколько файлов:

```
var inpfile, outfile: text;
```

2. Связать программные имена файлов с именами на диске:

```
assign(inpfile,'class8a');
assign(outfile,'gruppa1');
```

3. Открыть файлы для записи или для чтения:

```
reset(inpfile);        { открыть для чтения }
rewrite(outfile);      { открыть для записи }
```

4. При чтении данных из файла указывать его имя:

```
read(inpfile,stroka);
readln(inpfile,stroka);
```

5. При записи данных в файл указывать его имя:

```
write(outfile,stroka);
writeln(outfile,stroka);
```

6. После завершения работы с файлом закрыть его:

```
close(inpfile);
close(outfile);
```

Задача. В файле "CLASS8A" хранится список учеников всего класса. Получить в файле "GRUPPA1" первую половину этого списка.

Заметим, что можно поступить двумя способами: прочитать весь список, подсчитывая количество учеников; затем вычислить половину

этого количества, закрыть файл, снова открыть его и прочитать половину списка.

```
program gruppa1;
  var inpfile, outfile:text;
      stroka          :string[15];
      k,n             :integer;
begin
  assign(inpfile,'class8a');
  assign(outfile,'gruppa1');
  reset(inpfile);
  rewrite(outfile);
  n:=0;
  while not eof(inpfile) do
    begin
      n:=n+1;
      readln(inpfile,stroka)
    end;  {n-количество учеников в классе}
  close(inpfile);
  reset(inpfile);
  n:=n div 2;
  for k:=1 to n do
    begin
      readln(inpfile,stroka);
      writeln(outfile,stroka)
    end;
  close(inpfile);
  close(outfile)
end.
```

При другом подходе можно организовать массив строк и читать фамилии из файла в этот массив; дочитав файл до конца, определить количество фамилий в списке, а затем напечатать нужное количество элементов массива. При таком подходе с файлом работаем только один раз; это несколько быстрее, но зато нам требуется дополнительная память.

```
program gruppa2;
  var inpfile, outfile:text;
      stroka          :array[1..50] of string[15];
      k,n             :integer;
begin
```

```
    assign(inpfile,'class8a');
    assign(outfile,'gruppa1');
    reset(inpfile);
    rewrite(outfile);
    n:=0;
    while not eof(inpfile) do
      begin
        n:=n+1;
        readln(inpfile,stroka[n])
      end;  {n-количество учеников в классе}
    n:=n div 2;
    for k:=1 to n do
      writeln(outfile,stroka[k]);
    close(inpfile);
    close(outfile)
  end.
```

Здесь условие NOT EOF(INPFILE) означает "пока не поступил конец файла при чтении из файла INPFILE". Существует аналогичное условие "пока не поступил конец строки при чтении из файла ..."; это записывается так:

```
while not eoln(inpfile) do ...
```

ЗАДАНИЯ.

1. Прочитать и вывести на экран данные из файла.
2. Скопировать один файл в другой без изменений.
3. Прочитать данные и вывести на экран первую букву из каждой строки файла.
4. Прочитать данные и вывести на экран первую и последнюю букву из каждой строки файла.
5. Прочитать данные и вывести на экран первое слово из каждой строки файла.
6. В некотором файле данные организованы так: сначала фамилия, потом, в следующей строке - несколько оценок этого ученика по интересующему нас предмету; количество оценок заранее неизвестно, известно лишь, что все они помещаются в одной строке. В следующей строке - фамилия очередного ученика, и т.д. Прочитать файл и выдать на экран фамилии и средние баллы всех учеников класса.
7. В одном файле содержатся несколько (не более 10) прилагательных, в другом - несколько (не более 6) существительных. Считая, что все они одного рода, напечатать все возможные сочета-

ния прилагательное - существительное.

8. В одном файле содержатся несколько (до 10) прилагательных мужского, женского и среднего рода, в другом - несколько (до 10) существительных второго и третьего склонения. Напечатать все возможные сочетания прилагательное - существительное.
9. В двух файлах находятся списки учеников 8-А и 8-Б классов. Получить в третьем файле общий список, в котором сначала будут содержаться фамилии из первого списка, а затем - из второго.
10. В двух файлах находятся два списка фамилий, причем каждый список упорядочен по алфавиту. Требуется получить общий упорядоченный про алфавиту список. Для простоты предположим, что в общем списке нет фамилий, начинающихся на одну и ту же букву.

5.2. Записи

Запись является сложным типом данных. Как и массив, запись объединяет группу данных. Но, в отличие от массива, в записи могут быть объединены данные разных типов. Компоненты записи называются полями. Пример - список участников городских соревнований, включающий фамилию ученика, класс, школу. Пусть фамилилия содержит не более 15 символов, класс записывается в виде 10-Б и содержит, таким образом, не более 4 символов; номер школы - целое число. Тогда запись может быть описана в программе следующим образом:

```
type pupil=record
              family:string[15];
              class :string[4];
              school:integer
            end;
var a:pupil;
```

При этом a является именем записи, family, class, school - именами полей записи.

Приведем пример использования записи в программе. Пусть нам нужно записать одну строку в файл, где хранится список участников соревнований:

```
program spisok1;
  type pupil=record
               family:string[15];
               class :string[4];
```

```
                    school:integer
                  end;
  var a   :pupil;
      outf:text;
begin
  assign(outf,'spisok');
  rewrite(outf);
  a.family:='Иванов';
  a.class :='10-B';
  a.school:=3;
  writeln(outf,a.family,a.class,a.school);
  close(outf)
end.
```

Мы видим, что, для того чтобы использовать какую-то компоненту записи, нужно указывать имя записи и имя поля. Можно несколько упростить программу, используя оператор присоединения WITH. Так, наша программа эквивалентна следующей:

```
program spisok2;
  type pupil=record
                family:string[15];
                class :string[4];
                school:integer
              end;
  var a   :pupil;
      outf:text;
begin
  assign(outf,'spisok');
  rewrite(outf);
  with a do
    begin
      family:='Иванов'; class :='10-B'; school:=3;
      writeln(outf,family,class,school)
    end
  close(outf)
end.
```

Приведем теперь программу чтения из файла данных, организованных в виде записи. Пусть нам нужно прочитать и напечатать весь список участников:

```
program spisok3;
  type pupil=record
                family:string[15];
                class :string[4];
                school:integer
              end;
  var a  :pupil;
      inf:text;
begin
  assign(inf,'spisok');
  reset(inf);
  while not eof(inf) do
    with a do
      begin
        readln(inf,family,class,school);
        writeln(family,class,school)
      end;
  close(inf)
end.
```

ЗАДАНИЯ.

1. Прочитать список и подсчитать количество Ивановых, принимавших участие в соревнованиях.
2. Прочитать и напечатать всех участников из третьей школы.
3. Прочитать и напечатать всех участников из девятого класса.
4. Прочитать и напечатать всех участников из десятого класса.
5. Данные в файле организованы в виде записи: фамилия - номер школы. Известно, что в городе 3 школы. Определить, учеников из какой школы больше.
6. Данные в файле организованы в виде записи: фамилия - номер школы. Определить количество школ, ученики которых внесены в список.
7. Данные в файле организованы в виде записи: фамилия - пол (д/м) - год рождения. Определить самую юную девочку; самого старшего мальчика.
8. В некотором отчете находятся данные, организованные в следующие записи: номер школы; число участников летнего трудового лагеря; общий заработок. Определить школу, в которой заработок одного ученика был максимальным.
9. Для подготовки к экзамену составлен список в таком виде: поэт - название стихотворения. Определить поэта, у которого в этом списке больше всего стихотворений.

10. Имеется список в таком виде: название планеты, радиус, масса. Определить планеты с минимальной и максимальной плотностью.

6. Задачи

1. Ввести n,R. Вычислить периметр и площадь вписанного и описанного n-угольников.
2. Имеется банка с квадратным дном. Сторона квадрата – a, высота – 2a. Банка наполнена водой до половины. В банку опускают тяжелые куб со стороной a/2, тетраэдр со стороной a/2 и шар с диаметром a/2. На сколько поднимется вода в банке?
3. Вводится число от 1 до 18. Считая, что это порядковый номер элемента, указать его группу и период.
4. Вычислить сумму цифр введенного числа.
5. Вычислить количество цифр введенного числа.
6. Напечатать все четырехзначные числа, которые одинаково читаются и слева направо, и справа налево.
7. Напечатать все трехзначные числа такие, у которых все цифры различны.
8. В книге 1000 страниц. На страницах обычно 40 строк по 60 символов. На каждой 10-й странице есть картинка на четверть странице, а на каждой 25-ой, кроме того, – картинка на полстранице. Примерно каждая четвертая буква – гласная, каждая пятая гласная буква – это буква "а". Подсчитать количество буква "а" в книге.
9. Напечатать 10 троек чисел таких, которые могут быть длинами сторон прямоугольного треугольника.
10. Для введенного числа n напечатать все его делители (т.е. такие числа, на которые n делится без остатка).
11. Числа 1,25,49 являются квадратами и образуют арифметическую прогрессию. Найдите еще 3 таких тройки.
12. Треугольником Паскаля называется такой треугольник из чисел, где по краям стоят единицы, а остальные числа равны сумме двух верхних. Напечатать n строк треугольника Паскаля.

```
        1
       1 1
      1 2 1
     1 3 3 1
    1 4 6 4 1        – 5-ая строка
   ...........
```

13. Написать программу, которая по введенному прилагательному единственного числа определяет его род.

14. Написать программу, которая по введенному прилагательному единственного числа печатает его множественное число.
15. Написать программу, которая вводит глагол в неопределенной форме и определяет его спряжение.
16. Написать программу, которая вводит прилагательное и изменяет его, добавляя суффикс -еньк- (хороший-хорошенький, малый-маленький)
17. Написать программу, которая вводит слово и заменяет в нем все буквы "а" на букву "о".
18. Ввести строку, содержащую несколько слов, разделенных запятыми (не более 10). Напечатать каждое слово с новой строки.
19. Ввести арифметическое выражение, содержащее однозначные числа и знаки "+" и "-". Напечатать значение выражения.
20. Данный текст зашифровать так, чтобы буквы, равноотстоящие от концов алфавита, кодировали друг друга.
21. Введенное слово разбить на слоги.
22. Дан сонет. Написать схему его рифм (например, схема 'аабб вгвг деед жж' означает, что в первом четверостишии рифма смежная, во втором - перекрестная, в третьем - опоясывающая).
23. Ввести дату в виде "23.05" и напечатать в виде "23 мая".
24. Даны три числа - число, месяц, год. Получить аналогичные три числа для следующего дня.
25. Используя функцию RANDOM, нарисовать иллюстрацию звездного неба. При этом точки правой верхней части экрана изображать знаком "." (точка), нижней левой - знаком "*" (звездочка).
26. В массиве A[1..80, 1..20] даны показатели кислотности почвы на поле 80*20 единиц. Требуется найти диапазон изменения pH, разбить этот диапазон на 3 интервала и отразить картину на экране следующим образом: в точку с координатами (X,Y) такими, что A(X,Y) принадлежит меньшему интервалу, вывести букву "К" (кислая реакция); среднему - "Н" (нормальная), большему - "Щ" (щелочная). Подписать, какому интервалу соответствует какая буква.
27. Имеется список книг в виде автор-название. Напечатать сведения о всех книгах, в названии которых встречается строка "компьютер" (заметим, что в слове "компьютеризация" строка "компьютер" тоже встречается).
28. Имеется список в виде название города - население в 1987г. - население в 1988г. Напечатать город, в котором был максимальным а) абсолютный прирост населения; б) относительный прирост населения.
29. В файле находятся 365 чисел - результаты ежедневных измерений

температуры в течение года. Напечатать минимальную и максимальную температуру за каждый месяц.

30. В файле находятся 365 строк вида "юг", "северо-запад", "северо-восток" - информация о направлении ветра в течение года. Напечатать, какое направление ветра было преобладающим в каждом месяце.

31. В файле находятся слова (не более 100). Для вводимых с экрана слов подобрать рифму из этих 100 слов либо сообщить, что такого слова нет.

32. В файле содержатся несколько имен (женских и мужских). Напечатать все возможные сочетания "имя и отчество", полученные из этих имен.

33. В файле хранится некоторое стихотворение. Требуется напечатать его так, чтобы каждая строчка находилась посередине экрана (т.е. чтобы количество пробелов слева от букв строки было равно количеству пробелов справа).

34. В файле содержится некоторый текст. Определите:

- количество слов;
- максимальную длину слова;
- минимальную длину слова;
- среднюю длину слова.

35. Определите, какая гласная в тексте встречается чаще всего.

36. Вывести в середину экрана слово и передвигать его по экрану следующим образом: стрелки изменяют координату первой буквы; вторая буква занимает бывшее положение первой, и т.д.

37. В файле хранится информация в следующем виде: автор - название - издательство - год издания. Напечатать: а) все книги указанного автора; б) все книги, выпущенные после указанного года; в) все книги двух указанных издательств.

38. Нарисовать таблицу 4*4, оставив в клетках место для двузначного числа. Расположите в клетках числа от 0 до 15 следующим образом: в очередную клетку выведите "?"; затем введите с клавиатуры число от 0 до 15; если такое число еще не встречалось, поместите его в указанную клетку; клетку с числом 0 оставьте пустой. (Вы получите стартовую позицию для игры в "пятнадцать".)

39. Запрограммировать игру в "пятнадцать" следующим образом: 1) расположить 15 чисел в матрице 4*4 случайным образом; 2) принимать от пользователя число и стрелку; если указанное число можно передвинуть в указанном направлении, то перерисовать числа на экране.

40. В некотором файле содержатся вопросы и ответы на них: в первой

строке вопрос, во второй ответ и т.д. Написать программу, которая выводит вопрос, принимает от пользователя ответ и сравнивает его с правильным ответом, а затем печатает, ответ верен или нет. После каждого ответа выводить в правый верхний угол экрана счетчик верных и неверных ответов.

Литература

1. Йенсен К., Вирт Н. Паскаль - М.:Финансы и статистика, 1982.
2. П.Грогоно. Программирование на языке Паскаль - М.:Мир, 1982.
3. С.А.Абрамов, Е.В.Зима. Начала программирования на языке паскаль. - М.: Наука, 1987.
4. Г.Л.Семашко, А.И.Салтыков. Программирование на языке паскаль. М.: Наука, 1988.
5. Н.И.Вьюкова, В.А.Галатенко, А.Б.Ходулев. Систематический подход к программированию - М.:Наука, 1988.
6. К.Боон. Паскаль для всех. - М.:Энергоатомиздат, 1988.

Содержание

Подписано к печати 8.09.89 г. TI338I. Тираж 660 экз.
Зак. 2I59P. Уч.-изд. л. 2,3. Усл.печ.л. 3,5. Формат 60x90/I6.
Бумага офсетная. Цена 50 к. Изд.№ 288.

Отпечатано с оригинала-макета на ротапринте в Отделе научно-технической информации Научного центра биологических исследований АН СССР в Пущине.