

В185.2

П44

Л. М. ПОДДУБНАЯ В. Ф. ШАНЬГИН

„МНЕ
НРАВИТСЯ
ПАСКАЛЬ“



Арифметическая машина осуществляет действия, которые ближе к действиям мысли, чем все, производимое животными. Но она не делает ничего такого, что указывало бы на то, что у нее есть воля, как она есть у животных.

Блез Паскаль

Действующие лица



Дока: Полиглот. Доброжелательный. Когда сидит за персональным компьютером, чувствует себя как рыба в воде.



Фока: Любознательный. Хочет овладеть секретами информатики. Стремится к общению с персональным компьютером.

ПРЕДИСЛОВИЕ

У Вас есть возможность работать на персональном компьютере? Вы хотите научиться программировать? — тогда изучайте язык Паскаль! Благодаря своей эффективности, логичности и наглядности язык получил широкое распространение во всем мире, и почти все вычислительные машины могут работать с этим языком.

В предлагаемой книге язык Паскаль представлен в популярной, доходчивой форме с большим числом разнообразных задач и иллюстраций. Каждая задача во время ее выполнения сопровождается программой вместе с информацией на экране дисплея: диалогом пользователя с компьютером, исходными данными и результатами.

Изложены все основные функции языка, при этом особое внимание уделено вводу и выводу данных, которые, как правило, вызывают большие трудности у начинающих программистов. Рассмотрены типовые алгоритмы, входящие практически в любую сложную программу.

В конце книги дано описание практической работы на персональном отечественном компьютере типа ДВК с помощью языка Паскаль.

В основу книги положен материал лекций, бесед и практических занятий, проводимых авторами в течение нескольких лет со школьниками, рабочими, инженерами, аспирантами и преподавателями.

Книга рассчитана на широкий круг читателей без ограничения возраста.

Желаем успеха!

ББК 22.18

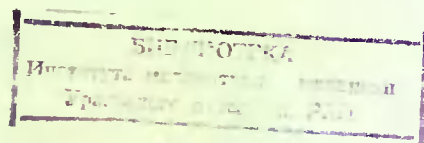
П 44

УДК 681.3.06:519.682.5

В185.25 Паскаль

Рецензенты: д-р техн. наук, профессор В. И. ВАСИЛЬЕВ,
канд. техн. наук М. П. ГРИШИН

Рисунки художников: В. И. ПОЛУХИНА,
С. А. РЕПЬЕВА



41676

Поддубная Л. М., Шаньгин В. Ф.

П 44 Мне нравится Паскаль. — М.: Радио и связь, 1992. — 160 с.:
ил. (Научно-популярная библиотека школьника)

ISBN 5-256-00636-3.

Рассматривается программирование на языке Паскаль для микроЭВМ. Стиль изложения рассчитан на школьников, язык описывается в популярной доходчивой форме. Приводится большое число разнообразных примеров от простых до сложных. Для каждого примера представлена программа и информация на экране дисплея во время ее выполнения: диалог пользователя с ЭВМ, конкретные исходные данные и результаты. Описание завершается практическим материалом для работы с языком Паскаль на ДВК.

Для школьников, учащихся ПТУ и начинающих программистов.

П 2404010000-104 132-91
046(01)-92

ББК 22.18

ISBN 5-256-00636-3

© Поддубная Л. М., Шаньгин В. Ф., 1992

Л. М. ПОДДУБНАЯ
В. Ф. ШАНЬГИН

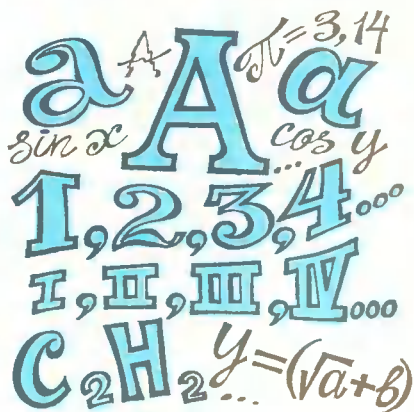
МНЕ НРАВИТСЯ ПАСКАЛЬ



Москва
"РАДИО И СВЯЗЬ"
1992

ДОКА, ФОКА И ПАСКАЛЬ

Картина 1. ВСТРЕЧА ДРУЗЕЙ



Фока. Я слышал, что ты стал полиглотом, Дока! Свободно владеешь многими языками программирования, легко общаешься с персональным компьютером. Научи и меня, пожалуйста, дружище!

Дока. С удовольствием!

Фока. По правде говоря, я даже не знаю, чем отличается язык программирования от обычного естественного языка, на котором мы говорим. Может быть, мы начнем с этого?

Дока. Ты затронул очень интересную тему. В русском алфавите, как известно, 33 буквы: А, Б, В, Г, Д, Е, Ё, Ж, З, И, Й, К, Л, М, Н, О, П, Р, С, Т, У, Ф, Х, Ц, Ч, Ш, Щ, Ъ, Ы, Ь, Э, Ю, Я.

Из этих букв можно составить огромное количество слов и предложений. В математике, физике, химии и других естественных науках помимо русского алфавита используются латинские и греческие буквы, арабские и римские цифры, круглые и фигурные скобки, специальные символы и т.д. Таким образом, возможности изложения материала на естественном языке неисчерпаемы.

Например, я хочу обратиться к тебе с просьбой, чтобы ты вычислил периметр прямоугольника со сторонами $a = 25$ и $b = 38$. Как я могу выразить свою просьбу на естественном языке? Ну, допустим:

* Вычисли, пожалуйста, периметр прямоугольника со сторонами 25 и 38.

* Если стороны прямоугольника равны 25 и 38, то чему равен его периметр?

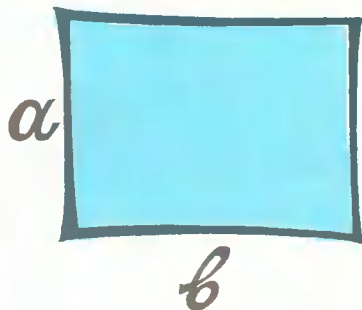
* Для любого прямоугольника периметр вычисляется по формуле $2a+2b$. Чему равен периметр со сторонами $a = 25$ и $b = 38$?

* Периметр прямоугольника определяется следующим образом: $p = 2(a+b)$. Вычисли значение p при $a = 25$ и $b = 38$.

И так далее.

Фока. А разве нельзя в такой форме обратиться к компьютеру?

Дока. К сожалению, нельзя. Хотя современный компьютер владеет многими языками программирования, он не в состоянии понять ни одну из этих просьб. Компьютер понимает и решает задачу, если ее условие записано на конкретном языке программирования, например Бейсике, Фортране, Паскале.



Любой язык программирования имеет ограниченные возможности: он содержит только определенный набор символов; запись слов и предложений подчиняется строгим правилам; каждый язык имеет свои, только ему присущие особенности.

Фока. Какой язык программирования мне изучить? Что посоветуешь, Дока?

Дока. Мне нравится Паскаль. Он разработан швейцарским профессором Никлаусом Виртом. В 1979 г. язык утвержден в качестве стандартного. Вирт назвал его в честь французского ученого Блеза Паскаля, изобретателя первой счетной машины.

ФОРТРАН
БЕЙСИК
ПАСКАЛЬ
АДА ПРОЛОГ
ПРОКАЛ



Блез Паскаль в юности
Рисунок Ж. Дюма

Язык Паскаль благодаря своей простоте, логичности и эффективности получил широкое распространение во всем мире. В настоящее время почти все вычислительные машины могут работать с этим языком.

Язык предназначен для решения самых разнообразных задач как деловых, так и игровых: вычислительных, экономических, информационных и т.д.

Фока. А какие слова используются в языке Паскаль? Русские, английские, французские или какие-то другие?

Дока. Язык Паскаль основан на английских словах.

Фока. Ясно! Если я хорошо изучу английский язык, то мне не надо будет изучать Паскаль. Я его уже буду знать!

Дока. Ты ошибаешься. Язык Паскаль — это специальный язык программирования для работы с компьютером. Знание английского в этом случае недостаточно.

Фока. А что если наоборот: я хорошо изучу язык Паскаль. Смогу ли я свободно говорить по-английски?

Дока. К сожалению, тоже нет, так как в состав языка Паскаль входят только небольшое число английских слов и существуют строгие правила их написания и использования.

Фока. Ну ты меня, Дока, заинтересовал! Любопытно посмотреть, как же все-таки выглядит программа вычисления периметра прямоугольника на языке Паскаль?

Дока. Пожалуйста, посмотри:

```
PROGRAM PRIM(INPUT,OUTPUT);
  VAR  A,B,P : INTEGER;
BEGIN
  A:=25;      (* ЗНАЧЕНИЕ СТОРОНЫ  А      *)
  B:=38;      (* ЗНАЧЕНИЕ СТОРОНЫ  В      *)
  P:=2*(A+B); (* ВЫЧИСЛЕНИЕ ПЕРИМЕТРА  Р    *)
  WRITE(P)    (* ВЫВОД НА ЭКРАН ЗНАЧЕНИЯ  Р *)
END.
```

Вот в таком виде компьютер поймет условие задачи. Конечно, на первый взгляд, здесь много непонятно. Давай разберем.

Первая строка — это стандартный заголовок программы. Любая программа начинается с заголовка. Не будем пока подробно изучать его, сделаем это позже.

Во второй строке за словом VAR перечисляются все переменные А, В, Р, которые встречаются в программе. Для каждой переменной дана характеристика, а именно: слово INTEGER означает, что переменная может принимать только целые значения (дробная часть недопустима).

Затем между словами BEGIN и END располагаются операторы. Они указывают, какие действия нужно выполнить. Первые два оператора присваивают переменным А и В их числовые значения. Третий оператор вычисляет значение $2 \cdot (A+B)$ и присваивает его переменной Р. Результат находится в памяти ЭВМ. Для вывода значения Р из памяти ЭВМ на экран дисплея используется оператор WRITE. В программе знак умножения обозначен "*", а знак присваивания ":=". В конце программы после слова END ставится точка.

Справа от операторов даны комментарии, т.е. пояснения к программе. Каждый комментарий расположен между символами (* и *).

Вот и все! Вопросы есть?

Фока. Можно сказать, что программа понятна. Но что делать с ней? Вот передо мной компьютер, а вот программа — как их объединить?

Дока. Если эту программу ввести в компьютер, то — увы! - решение мы не получим: компьютер запомнит условие задачи и только! Он ведет себя подобно человеку, который запомнил задачу, а будет решать ее или нет, еще неизвестно.

Для того чтобы компьютер начал действовать, ему надо дать указание: "Программа готова, решай!".

После этого начнут выполняться последовательно все операторы программы и на экране появится ответ. Для нашей задачи — 126.

DO YOU SPEAK
ENGLISH?
Parlez vous
Français?
Sprechen Sie
Deutsch?

А вот как ввести программу в компьютер, как заставить его находить ошибки, как указать ему команду "Решай!", представлено в конце книги. Пока еще туда заглядывать рано.

А теперь еще раз вернемся к программе вычисления периметра прямоугольника.

Рассмотренная программа хотя и написана и верно, но обладает серьезным недостатком: периметр прямоугольника вычисляется только при конкретных значениях сторон $a=25$ и $b=38$.

Желательно программу составлять в общем виде так, чтобы она по возможности не зависела от конкретных значений и могла выполняться при любых других исходных данных.

Например, ту же самую задачу можно представить в следующем виде:

```
PROGRAM PRIM(INPUT,OUTPUT),
  VAR  A,B,P : INTEGER;
BEGIN
  WRITE('ВВЕДИТЕ ЗНАЧЕНИЯ A,B: ');
  READ(A,B);      (* ВВОД ЗНАЧЕНИЙ A,B          *)
  P:=2*(A+B);      (* ВЫЧИСЛЕНИЕ ПЕРИМЕТРА P      *)
  WRITE(P)         (* ВЫВОД НА ЭКРАН ЗНАЧЕНИЯ P  *)
END.
```

Здесь вместо конкретных значений A и B указан оператор ввода исходных данных READ (A,B).

После набора на экране дисплея всей программы и запуска ее на выполнение происходит останов машины при встрече оператора READ (A,B). Необходимо ввести с клавиатуры два числа через пробел, например 25_38. Переменная A получит значение первого числа, а B — второго. При этих значениях будет вычислен периметр прямоугольника. Если бы мы ввели другие два числа, например 14_7, то и при этих значениях был бы вычислен периметр. Программа стала универсальнее.

Для того чтобы знать, когда и какие значения нужно вводить, перед оператором ввода READ ставят оператор вывода WRITE, в котором в апострофах указывается поясняющий текст. При выполнении оператора вывода этот текст появляется на экране. Например, оператор вывода

WRITE (' ВВЕДИТЕ ЗНАЧЕНИЯ A,B:')

выводит на экран текст

ВВЕДИТЕ ЗНАЧЕНИЯ A,B:

после чего можно вводить значения A и B.

При выполнении всей программы информация на экране дисплея примет вид

```
=====
=   ВВЕДИТЕ ЗНАЧЕНИЯ  A,B: 25 38   =
=                               =
=                               126  =
=====
```

Результат вычисления периметра, равный 126, появился в новой строке. Для него автоматически отводится 13 позиций. Так как число занимает только три позиции, то перед ним расположены десять пробелов.

Для наглядности результата также можно перед выводом значения P дать поясняющий текст. Например, при использовании оператора

WRITE ('ПЕРИМЕТР = ', P)

информация на экране дисплея при выполнении программы примет вид:

```
=====
=   ВВЕДИТЕ ЗНАЧЕНИЯ  A,B: 25 38   =
=   ПЕРИМЕТР =          126        =
=====
```


Наглядность результата можно еще улучшить, если приблизить число 126 к знаку равенства "=". Это можно сделать, если в операторе вывода WRITE после переменной P указать через двосточие конкретное число позиций для изображения значения P. Например, в случае P:3 получим

$$\text{ПЕРИМЕТР} = \sqrt[3]{126},$$

а в случае P:5 получим

$$\text{ПЕРИМЕТР} = \sqrt[5]{126}.$$

Перед началом программы можно дать комментарий, тогда программа будет такой:

```
(*-----
!   ВЫЧИСЛЕНИЕ ПЕРИМЕТРА ПРЯМОУГОЛЬНИКА   !
*-----*)
PROGRAM PRIM(INPUT,OUTPUT);
  VAR A,B,P : INTEGER;
BEGIN
  WRITE('ВВЕДИТЕ ЗНАЧЕНИЯ  A,B: ');
  READ(A,B);          (* ВВОД ЗНАЧЕНИЙ  A,B          *)
  P:=2*(A+B);          (* ВЫЧИСЛЕНИЕ ПЕРИМЕТРА  P          *)
  WRITE('ПЕРИМЕТР =', P:4) (* ВЫВОД НА ЭКРАН ЗНАЧЕНИЯ  P *)
END.
```

Информация на экране дисплея во время выполнения этой программы примет вид

```
=====
=   ВВЕДИТЕ ЗНАЧЕНИЯ  A,B: 25  38   =
=   ПЕРИМЕТР = 126                     =
=====
```

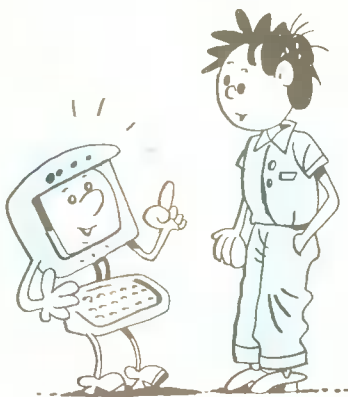
При составлении программы нужно заботиться о том, чтобы ввод исходных данных в память ЭВМ и вывод результатов на экран дисплея были наглядными и понятными.

Фока. Я принимаю это к сведению, но мне кажется, что программа получилась очень большая для такой простой задачи.

Дока. Согласен. Но!

Во-первых, по этой программе можно вычислять периметры прямоугольников, у которых стороны равны не только 25 и 38.

Во-вторых, этой программой может воспользоваться любой желающий. При этом он может и не разбирать саму программу, и даже не видеть ее. Программа хранится на диске. Достаточно запустить ее на выполнение, она запросит исходные данные и после их ввода на экране появится ответ.



Кстати, вот теперь можно обратиться к последнему разделу нашего повествования "ЗРИ В КОРЕНЬ" и поработать с языком Паскаль на персональном компьютере. И все же советую сделать это несколько позже.

Фока. По-видимому программа будет очень большой, возможно, на нескольких страницах, если понадобится вычислить периметры 25 или 50 разных прямоугольников? Устанешь составлять такую программу.

Дока. Вот здесь ты ошибаешься. В языке Паскаль можно использовать специальный оператор — оператор цикла, с помощью которого можно повторять некоторые действия. Смотри, как выглядит эта же программа для вычисления периметров восьми прямоугольников:

```
(*-----
 I  ВЫЧИСЛЕНИЕ ПЕРИМЕТРОВ ПРЯМОУГОЛЬНИКОВ  I
-----*)
PROGRAM PRIM(INPUT,OUTPUT);
  VAR  A,B,P,K : INTEGER;
BEGIN
  FOR K:=1 TO 8 DO          (* ОПЕРАТОР ЦИКЛА      *)
  BEGIN
    WRITE('ВВЕДИТЕ ЗНАЧЕНИЯ  A,B: ');
    READ(A,B);              (* ВВОД ЗНАЧЕНИЙ  A,B *)
    P:=2*(A+B);              (* ВЫЧИСЛЕНИЕ P    *)
    WRITELN('ПЕРИМЕТР =', P:4);(* ВЫВОД ЗНАЧЕНИЯ  P *)
    WRITELN('-----')      (* ВЫВОД ЛИНИИ    *)
  END
END.
```

Результаты вычислений:

```
ВВЕДИТЕ ЗНАЧЕНИЯ  A,B: 25 38
ПЕРИМЕТР = 126
-----
ВВЕДИТЕ ЗНАЧЕНИЯ  A,B: 12 4
ПЕРИМЕТР = 32
-----
ВВЕДИТЕ ЗНАЧЕНИЯ  A,B: 5 5
ПЕРИМЕТР = 20
-----
ВВЕДИТЕ ЗНАЧЕНИЯ  A,B: 1 1
ПЕРИМЕТР = 4
-----
ВВЕДИТЕ ЗНАЧЕНИЯ  A,B: 0 0
ПЕРИМЕТР = 0
-----
ВВЕДИТЕ ЗНАЧЕНИЯ  A,B: 10 20
ПЕРИМЕТР = 60
-----
ВВЕДИТЕ ЗНАЧЕНИЯ  A,B: 100 200
ПЕРИМЕТР = 600
-----
ВВЕДИТЕ ЗНАЧЕНИЯ  A,B: 12 12
ПЕРИМЕТР = 48
-----
```

В программе оператор цикла FOR K:=1 TO 8 DO повторяет 8 раз действия тех операторов, которые расположены далее между словами BEGIN и END. При этом переменная K последовательно принимает значения 1,2,3,...,8. Для вычисления периметров 25 прямоугольников нужно указать в операторе цикла вместо числа 8 число 25: FOR K:=1 TO 25 DO.

Естественно, что задачи с циклами являются более сложными, поэтому мы их рассмотрим позже.

Картина 2. ЗАБЕГАЯ ВПЕРЕД...

Дока. Мне кажется, что будет полезнее рассмотреть еще несколько программ на языке Паскаль и только после этого начинать изучать возможности языка. Впрочем, можно и не разбирать задачи и программы, если они покажутся сложными, а сразу переходить к изучению языка.

Фока. Нет! Все же интересно посмотреть, как выглядят другие программы.

Дока. Ну, тогда начнем...

Задача 1. Хорошо бы нам покрасить дом! Но чтобы рассчитать, сколько нужно краски для этого, необходимо знать площадь окрашиваемой поверхности. Составим программу вычисления площади окрашиваемой поверхности лицевой стороны дома. Окна в покраску не входят. Все исходные данные и рисунок дома представлены в программе.



```
(*-----*)
!                ПОКРАСКА ДОМА                !
(*-----*)

PROGRAM DOM(INPUT,OUTPUT);
VAR
  A,B:REAL;      (* РАЗМЕРЫ СТЕНЫ                *)
  C,D:REAL;      (* РАЗМЕРЫ ОКНА                  *)
  P:REAL;        (* ОКРАШИВАЕМАЯ ПЛВЕРХНОСТЬ *)
BEGIN
  (*
    /-----\
   /         \
  /-----\
 I   ---   I
 I C I I   I I   I I   I
 I         I
 I   D     I
 I-----I
      B
  *)

  WRITE('ВВЕДИТЕ РАЗМЕРЫ СТЕНЫ  A,B: ');
  READ(A,B);
  WRITE('ВВЕДИТЕ РАЗМЕРЫ ОКНА   C,D: ');
  READ(C,D);
  P:=A*B-3*C*D;
  WRITE('ПЛОЩАДЬ ПОКРАСКИ =', P:7:2)

END.
=====
=
=  ВВЕДИТЕ РАЗМЕРЫ СТЕНЫ  A,B: 4.5  8.5
=  ВВЕДИТЕ РАЗМЕРЫ ОКНА   C,D: 2  1.5
=  ПЛОЩАДЬ ПОКРАСКИ = 29.25
=
=====
```



```

WRITE('ВВЕДИТЕ РАЗМЕРЫ СТЕНЫ A,B: ');
READ(A,B);
WRITE('ВВЕДИТЕ РАЗМЕРЫ ОКНА C,D: ');
READ(C,D);
WRITE('ВВЕДИТЕ КОЛИЧЕСТВО ОКОН: ');
READ(N);
P:=A*B-N*C*D;
WRITE('ПЛОЩАДЬ ПОКРАСКИ =', P:7:2)
END.
=====
=
= ВВЕДИТЕ РАЗМЕРЫ СТЕНЫ A,B: 4.5 8.5 =
= ВВЕДИТЕ РАЗМЕРЫ ОКНА C,D: 2 1.5 =
= ВВЕДИТЕ КОЛИЧЕСТВО ОКОН: 3 =
= ПЛОЩАДЬ ПОКРАСКИ = 29.25 =
=
=====

```

Результат программы не изменился, но сама программа стала универсальнее, потому что не зависит от конкретного числа окон.

Задача 2. Полторы кошки за полтора часа съедают полторы мышки. Сколько съедят мышек 100 кошек за 50 часов.

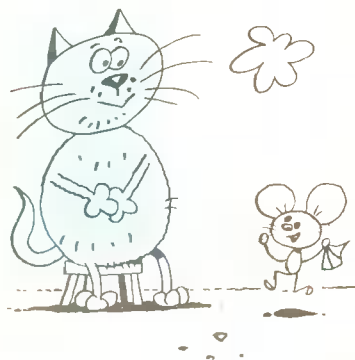
```

(*-----*)
I      " КОШКИ - МЫШКИ "      I
I      НАРОДНЫЙ ФОЛЬКЛОР      I
(*-----*)
PROGRAM КОТ(INPUT,OUTPUT);
CONST
  K=100;      (* КОЛИЧЕСТВО КОШЕК *)
  T=50;      (* КОЛИЧЕСТВО ЧАСОВ *)
  VAR M:REAL; (* КОЛИЧЕСТВО МЫШЕК *)
BEGIN
  (*-----*)
  (* ОДНА КОШКА ЗА 1,5 ЧАСА С"ЕСТ ОДНУ *)
  (* МЫШКУ, А ЗА ОДИН ЧАС 1/1,5 МЫШЕК *)
  (*-----*)

  M:= 1/1.5 * K * T;
  WRITE(' 100 КОШЕК ЗА 50 ЧАСОВ С"ЕАЯТ',
        M:9:3, ' МЫШЕК' )
END.
=====
=
= 100 КОШЕК ЗА 50 ЧАСОВ С"ЕАЯТ 3333.333 МЫШЕК =
=
=====

```

Так как в этой задаче-шутке заранее оговорено, что интерес представляют 100 кошек и 50 часов, то эти величины заданы в разделе описания констант CONST. Здесь K и T — константы и они не могут принимать другие значения в программе. Исходные данные заданы с дробной частью: 1,5 кошки, мышки, время. Поэтому все вычисления должны быть действительного типа, а значит, и конечный результат M — тоже действительного типа. Оператор ввода не использован, а конкретные значения указаны непосредственно при вычислении значения M.



Оператор WRITE сначала выводит строку текста, затем значение M, после чего опять строку текста. Условие задачи записано в комментариях в разделе операторов.

При вычислении количества мышек M операция деления обозначена наклонной чертой "/".

В изображении числа дробная часть отделяется от целой не запятой, а точкой.

Задача 3. Лист бумаги разрезают пополам. Одну из полученных половин снова делят пополам и так далее. Сколько понадобится делений, чтобы получить частицы размером с атом?

Примем для определенности, что бумажный лист имеет массу 1 грамм, а атом $1/10^{24} = 10^{-24}$ граммов.

Сколько же будет таких делений? Может быть сначала посмотреть на ответ, а потом разбирать задачу? Пожалуй, так и сделаем!

```
(*-----*)
I      ДЕЛЕНИЕ ЛИСТА БУМАГИ      I
I      НА ДВЕ ЧАСТИ              I
(*-----*)
PROGRAM LIST (INPUT,OUTPUT);
VAR
  P:REAL;      (* МАССА ЛИСТА БУМАГИ *)
  I:INTEGER;    (* ЧИСЛО ДЕЛЕНИЙ *)
BEGIN
  I:=0; P:=1;   (* НАЧАЛЬНЫЕ ЗНАЧЕНИЯ *)
  WHILE P > 1E-24 DO
    BEGIN
      P:=P/2;
      I:=I+1;
    END;
    WRITELN(' ВСЕГО ПОЛУЧИТСЯ', I:4, ' ДЕЛЕНИЙ !');
    WRITELN;
    WRITELN('ВЫ НЕ ВЕРИТЕ? ВЫ ОЖИДАЛИ МИЛЛИАРДЫ...');
    WRITELN('ПРОВЕРЬТЕ ОПЫТНЫМ ПУТЕМ, ЖЕЛАЮ УДАЧИ !')
  END.
=====
=
=      ВСЕГО ПОЛУЧИТСЯ      80      ДЕЛЕНИЙ !      =
=
=      ВЫ НЕ ВЕРИТЕ? ВЫ ОЖИДАЛИ МИЛЛИАРДЫ...      =
=      ПРОВЕРЬТЕ ОПЫТНЫМ ПУТЕМ, ЖЕЛАЮ УДАЧИ !      =
=
=====
```

Число 10^{-24} в языке Паскаль записывается в виде 1E-24. Так как процесс деления листа бумаги многократно повторяется от 1 до 10^{-24} граммов, то он отражен оператором повторения WHILE: пока вес бумаги P больше чем 10^{-24} , выполняются операторы, расположенные далее между словами BEGIN и END. Новое значение веса листа всегда в 2 раза меньше предыдущего значения. Это отражается оператором P:=P/2. Переменная I выполняет роль счетчика делений. Его значение при каждом делении увеличивается на единицу I:=I+1. Как только получится $P \leq 10^{-24}$, процесс повторения заканчивается и происходит вывод результата вместе с пояснениями на экран дисплея.

Задача 4. Условие этой задачи записано в программе. А программа составлена таким образом, что устанавливается диалог между пользователем и компьютером. После того как программа начала выполняться, на экране дис-

появляется условие задачи и вопрос. Нужно решить задачу устно и ввести ответ. Машина сравнит этот результат с тем, что она вычислила сама и ответит, правильно или нет решена задача.

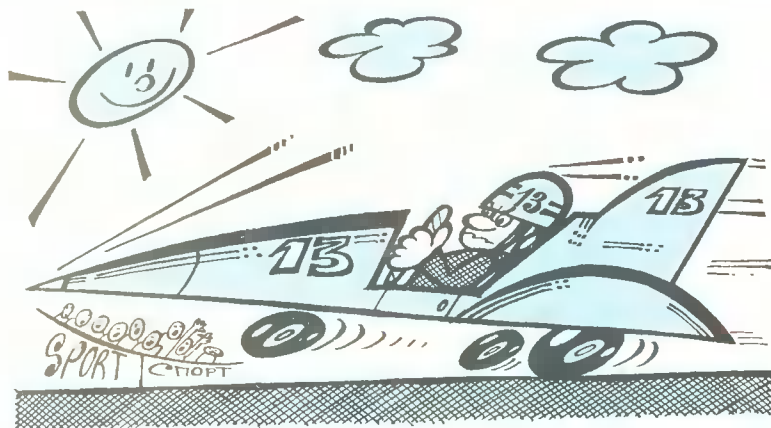
```

(*-----*
I      ЗАДАЧА ПО ФИЗИКЕ      I
*-----*)
PROGRAM AT(INPUT,OUTPUT);
VAR
  F  : INTEGER;
  F1 : INTEGER;
  M  : INTEGER;
  A  : INTEGER;
BEGIN
  WRITELN('АВТОМОБИЛЬ ДВИЖЕТСЯ С УСКОРЕНИЕМ A=2 (М/С**2) ');
  WRITELN('С КАКОЙ СИЛОЙ F ЧЕЛОВЕК МАССОЙ M=70 КГ ДАВИТ ');
  WRITELN('НА СПИНКУ СИДЕНЬЯ АВТОМОБИЛЯ ? ');
  WRITELN('***** ');
  WRITELN('ОТВЕТ ВВЕДИТЕ В НЬЮТОНАХ : ');
  READLN(F);
  A:= 2;
  M:= 70;
  F1:= A*M;
  IF F=F1 THEN WRITE(' МОЛОДЕЦ! ЗАДАЧА РЕШЕНА ВЕРНО! ')
    ELSE WRITE(' НЕВЕРНО ');
END.
=====
=
= АВТОМОБИЛЬ ДВИЖЕТСЯ С УСКОРЕНИЕМ A=2 (М/С**2) =
= С КАКОЙ СИЛОЙ F ЧЕЛОВЕК МАССОЙ M=70 КГ ДАВИТ =
= НА СПИНКУ СИДЕНЬЯ АВТОМОБИЛЯ ? =
= ***** =
= ОТВЕТ ВВЕДИТЕ В НЬЮТОНАХ : =
= 140 =
= МОЛОДЕЦ! ЗАДАЧА РЕШЕНА ВЕРНО! =
=====

```

В программе приняты следующие обозначения: F1 — сила, значение которой вычисляет машина: F — сила, значение которой вычисляется устно.

Для сравнения результатов вычисления введен новый оператор IF. В нем проверяется условие: если $F = F1$, тогда на экран дисплея выводится текст о правильном решении задачи, иначе — о неправильном решении.



Картина 3. КИРПИЧИКИ, ИЗ КОТОРЫХ СТРОИТСЯ ЗДАНИЕ

Фока. В задачах, которые мы рассмотрели, используются то русские слова, то английские, а то и отдельные буквы, цифры, символы. Мне непонятно, какими символами и словами можно пользоваться при составлении программы?

Дока. Давай разберем этот вопрос.

Любой естественный язык (русский, английский, французский и др.) состоит из нескольких основных элементов: символов, слов и предложений. В языке программирования имеются аналогичные структурные элементы: символы, слова и операторы. При этом слово образуется из последовательности символов, а оператор — из комбинации слов. Рассмотрим подробно элементы языка Паскаль.

Символы языка. Набор символов, которыми можно пользоваться при составлении программы на языке Паскаль, ограничен. Основными символами языка являются:

1) 26 букв латинского алфавита

A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z;

2) арабские цифры

0, 1, 2, 3, 4, 5, 6, 7, 8, 9 (чтобы отличить цифру ноль 0 от буквы O, цифру иногда подчеркивают косой чертой 0);

3) 32 буквы русского алфавита (для отечественных ЭВМ)

A, Б, В, Г, Д, Е, Ж, З, И, Й, К, Л, М, Н, О, П, Р, С, Т, У, Ф, Х, Ц, Ч, Ш, Щ, Ъ, Ы, Ь, Э, Ю, Я;

4) специальные символы

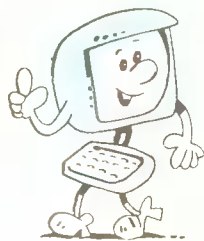
Символ	Название	Символ	Название
+	Плюс	:	Двосточие
—	Минус	,	Апостроф
*	Звездочка	<	Меньше
/	Наклонная черта	>	Больше
=	Равно	()	Скобки круглые
.	Точка	[]	Скобки квадратные
,	Запятая		Пробел (пустая позиция, условно обозначим символом _)
;	Точка с запятой		



Все эти символы имеются на клавиатуре. Пример клавиатуры, наиболее распространенной для отечественных компьютеров, приведен в Приложении 1. Следует обратить внимание на то, что клавиатура содержит значительно больше символов, чем можно использовать в языке Паскаль, например символы восклицательного "!" и вопросительного "?" знаков. Однако эти символы и многие другие можно использовать в комментариях.

Некоторые буквы русского алфавита могут быть запрещены (чаще всего Щ, Ъ, Ч, Ю).

Во многих версиях языка допускается использование строчных и прописных букв как русского, так и латинского алфавитов.



Вся совокупность символов клавиатуры упорядочена, т.е. каждый символ имеет свой порядковый номер (см. Приложение 2), при этом латинские буквы упорядочены в соответствии с алфавитом, а русские буквы расположены в ином порядке.

При изложении материала примем следующее решение:

в программах используются только заглавные буквы русского и латинского алфавитов;

строчными русскими буквами будем пояснять текст какой-либо конструкции языка Паскаль.

В программе на месте этого текста должны быть элементы языка Паскаль.

Следующие определения будем пояснять на примере программы 1 вычисления периметра прямоугольника:

Программа 1

```
(*-----*  
!   ВЫЧИСЛЕНИЕ ПЕРИМЕТРА ПРЯМОУГОЛЬНИКА   !  
-----*)  
PROGRAM PRIM(INPUT,OUTPUT);  
  VAR  A,B,P : INTEGER;  
BEGIN  
  WRITE('ВВЕДИТЕ ЗНАЧЕНИЯ  A,B: ');  
  READ(A,B);          (* ВВОД ЗНАЧЕНИЙ  A,B          *)  
  P:=2*(A+B);          (* ВЫЧИСЛЕНИЕ ПЕРИМЕТРА  P          *)  
  WRITE('ПЕРИМЕТР =', P:4) (* ВЫВОД НА ЭКРАН ЗНАЧЕНИЯ  P *)  
END.
```

Слово. Из отдельных символов языка образуются слова, которые имеют определенный смысл в программе. Слова в тексте разделяются пробелами или другими специальными символами.

Примеры слов в программе:

PROGRAM, PRIM, INPUT, OUTPUT, VAR, INTEGER, BEGIN, WRITE, END.

Если между словами стоит какой-нибудь разделитель, например, знаки "+", "-", ":", ";", ")" и другие, то пробел можно ставить до и после разделителя, но можно его и не ставить. Там, где допускается один пробел, можно ставить любое количество пробелов.

В зависимости от назначения различают служебные слова и имена:

Служебное слово. Оно имеет в языке Паскаль определенное назначение и его нельзя использовать для других целей.

В программе 1 служебными словами являются: PROGRAM, VAR, BEGIN, END. Полный перечень служебных слов приведен в Приложении 3.

Имя. В языке Паскаль различают два вида имен: стандартные и даваемые программистом.

Стандартные имена уже заложены в языке для обозначения стандартных (встроенных) объектов. Так, в программе 1 используются встроенные программы ввода и вывода, которые имеют стандартные имена READ и WRITE. Полный перечень стандартных имен приведен в Приложении 4.

Какие имена может давать программист? Любые, но...

ИМЯ — последовательность букв и цифр, начинающаяся с буквы

Примеры имен: K, A16, ELENA, ЕЛЕНА, MAX, ИВАНБОРИСОВИЧ, SUMMA, СКОРОСТЬ, A4R, ALPHA, MOS15A, СКОРОСТЬДВИЖЕНИЯ.

Следует заметить, что в качестве имени нельзя давать служебные слова и стандартные имена. Составные имена пишутся слитно, потому что пробел, тире, черточка не являются буквой или цифрой, а значит, не должны входить в состав имени. В некоторых версиях языка принято, чтобы имена отличались друг от друга, по крайней мере, одним из первых восьми символов.

Примеры неправильных имен:

(МЕТР) — круглые скобки;
END — служебное слово;
4A — начинается с цифры;
км/час — наклонная черта;
N 824 — пробел;
ВЕГА-5 — тире;

В программе периметр обозначен именем Р. Вместо него можно было бы указать имя Р1, PERIM, П, ПЕРИМЕТР и др. Стороны обозначены именами А и В, их также можно обозначить иначе.

Комментарий. Комментарий служит для пояснения программы или отдельных ее частей. Наличие комментариев делает программу более понятной и удобной для чтения. Комментарий языка Паскаль — это последовательность символов, ограниченная слева парой символов "(*" и справа парой символов "**)". Вместо круглых скобок допускается наклонная черта. Примеры комментариев:

(* ВЫЧИСЛЕНИЕ СУММЫ ЭЛЕМЕНТОВ *)

или

/* ЗНАЧЕНИЕ ПЕРИМЕТРА */

В программе даны комментарии ко всем операторам.

При выполнении программы комментарии игнорируются и не влияют на решение задачи.

Комментарии можно свободно распределять по всей программе. Они могут быть вставлены в любое место, где используется пробел.

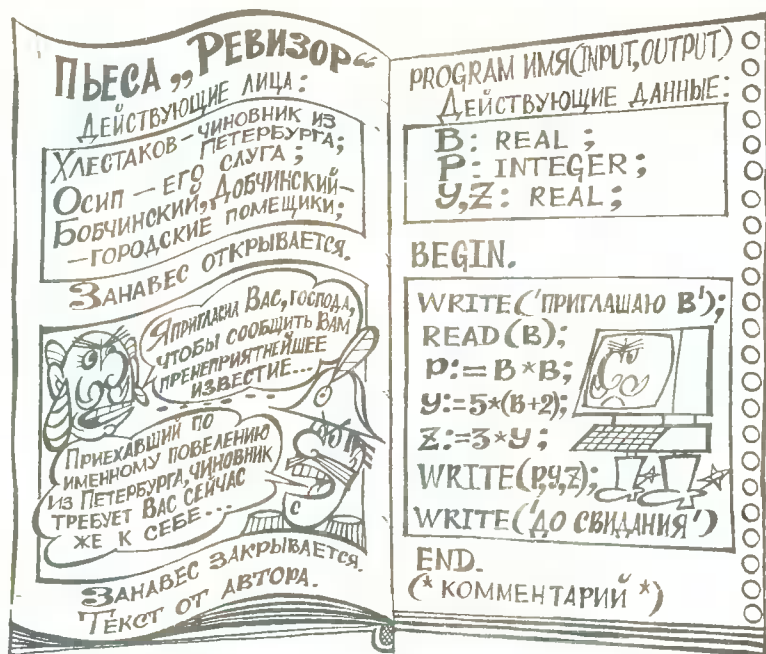
Как правило, допускается вставлять комментарий и до заголовка программы. Кроме того, в комментариях разрешается использовать не только символы языка Паскаль, но и другие символы, имеющиеся на клавиатуре (см. Приложения 1 и 2).

Если комментарий занимает несколько строк, то достаточно указать начало комментария перед первой строкой и конец — после последней строки. В то же время признаки начала и конца комментария можно давать в каждой строке.

Картина 4. ТЕАТРАЛЬНАЯ АФИША

Дока. Мы рассмотрели символы и слова языка Паскаль. Но как показать структуру программы в общем виде? Что-то я загрудняюсь это сделать. С чем бы ее сравнить? Придумал! Программа имеет точно такой же общий





вид, что и пьеса: сначала дается название пьесы (заголовок программы), затем указываются действующие лица и их характеристики (описание данных), после чего идут сами действия (операторы).

Программа на языке Паскаль состоит из заголовка, раздела описаний и раздела операторов (рис. 1).

Заголовок содержит служебное слово PROGRAM, имя программы и в круглых скобках параметры INPUT и OUTPUT для связи микроЭВМ с внешними устройствами (стандартным внешним устройством ввода является клавиатура, выводом — дисплей; на них и указывают соответственно параметры INPUT и OUTPUT).

Имя программы задает программист — разработчик данной программы. Оно может быть любым, но, как правило, имеет длину не более шести символов. Заголовок заканчивается символом точки с запятой.

Раздел описаний предназначен для объявления всех встречающихся в программе данных и их характеристик (имена данных, их тип, возможные значения и др.).

Этот раздел в общем случае содержит в себе другие разделы: описание меток, констант, типов, переменных, а также процедур и функций. Рекомендуется располагать их в названном порядке. Каждое описание заканчивается точкой с запятой. В программе необязательно наличие всех этих разделов. Так в простой программе может быть только один раздел — описание переменных и даже ни одного, допускается вообще отсутствие раздела описаний.

PROGRAM — ПРОГРАММА
BEGIN — НАЧАЛО
END — КОНЕЦ

Раздел операторов заключается в операторные скобки BEGIN и END, при этом после END ставится точка. В разделе операторов записывается последовательность операторов. Каждый оператор выражает действия, которые необходимо выполнить. Между операторами ставится символ точки с запятой ";". После слова BEGIN и перед словом END точка с запятой не ставится.

Структура программы на языке Паскаль представлена в общем виде на рис.1.

```

PROGRAM ИМЯ(INPUT,OUTPUT);

  LABEL - РАЗДЕЛ МЕТОК;

  CONST - РАЗДЕЛ КОНСТАНТ;

  TYPE - РАЗДЕЛ ТИПОВ;

  VAR - РАЗДЕЛ ПЕРЕМЕННЫХ;

  PROCEDURE, FUNCTION - РАЗДЕЛ
                        ПРОЦЕДУР И ФУНКЦИЙ;

BEGIN
  -----
  I      I
  I      РАЗДЕЛ ОПЕРАТОРОВ      I
  I      I
  I      I
  -----
END.

```

Рис.1. Структура программы

Конечно, многое здесь не понятно, но нашей целью как раз и является постепенное изложение и пояснение всех элементов программы. Сейчас же самое главное — "увидеть" всю структуру программы в целом.

Сама же программа записывается в свободной форме, операторы не привязаны к определенной позиции строки в отличие от многих других языков программирования.

В одной строке можно указывать несколько описаний или операторов.

Допускается перенос с одной строки на другую отдельных частей описаний или операторов, но специального символа переноса нет. В то же время не разрешается разделять слова, числа, составные символы. Хороший стиль программирования — это написать программу так, чтобы.

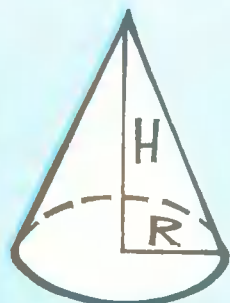
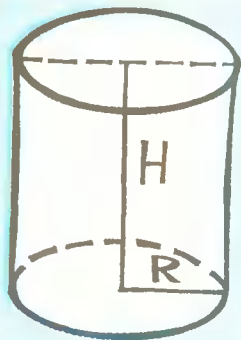
*ее было легко читать и понимать,
легко было использовать,
она повышала настроение!*



Для этого в программе необходимо широко использовать комментарии, пробелы и пустые строки. Желательно смысловые части выделять одинаковым отступлением от начала строки. Так, для выделения заголовка программы, раздела описаний и раздела операторов удобно записывать слова PROGRAM, BEGIN, END с одной позиции строки. По отношению к ним соответствующие описания или операторы сдвигаются вправо. Желательно сдвиги делать на одинаковое число позиций по отношению к предыдущему сдвигу.

Задача 5. Составить программу вычисления объемов цилиндра и конуса, которые имеют одинаковую высоту H и одинаковый радиус основания R .

Объем цилиндра вычисляется по формуле $V = \pi R^2 H$, а объем конуса — по формуле $V = 1/3 \pi R^2 H$, где $\pi = 3,14$. Поскольку символа π нет в языке Паскаль, заменим его на PI. Если значение π в программе не изменяется, то его можно вынести в раздел констант



```
(*-----
I   ВЫЧИСЛЕНИЕ ОБЪЕМОВ ЦИЛИНДРА И КОНУСА   I
-----*)
PROGRAM VOL(INPUT,OUTPUT);
  CONST  PI = 3.14;
  VAR
    R   : REAL; (* РАДИУС ОСНОВАНИЯ *)
    H   : REAL; (* ВЫСОТА              *)
    V1  : REAL; (* ОБЪЕМ ЦИЛИНДРА      *)
    V2  : REAL; (* ОБЪЕМ КОНУСА        *)
  BEGIN
    WRITE('ВВЕДИТЕ ЗНАЧЕНИЯ R,H : ');
    READ( R,H );
    V1:= PI*R*R*H;
    V2:= V1/3;
    WRITELN('*****');
    WRITELN('ОБЪЕМ ЦИЛИНДРА =', V1:10:4 );
    WRITE('ОБЪЕМ КОНУСА =', V2:10:4 )
  END.
```

```
=====
*
=   ВВЕДИТЕ ЗНАЧЕНИЯ R,H : 2.5  10   =
=   *****                          =
=   ОБЪЕМ ЦИЛИНДРА =  193.2500      =
=   ОБЪЕМ КОНУСА  =   65.4167      =
=                                     =
=====
```



```
PROGRAM VOL(INPUT,OUTPUT);CONST PI 3.14;  
VAR R:REAL;H:REAL;V1:REAL;V2:REAL;BEGIN  
READ(R,H);V1:=PI*R*R*H;V2:=V1/3;Writeln  
(V1);WRITE(V2)END.
```

```
2.5 10  
1.962500E+02  
6.541667E+01
```

Картина 5. ДЕЙСТВУЮЩИЕ ДАННЫЕ

Фока. Всем понятно, кто такие действующие лица в пьесе. Но в программе?.. Может быть это я — главный герой в программе или ты, Дока?

Дока. Нет, дружище! Основными "действующими лицами" в программе являются константы и переменные.

Константа не изменяет своего значения в процессе выполнения программы. Она может быть задана явно своим значением или обозначена именем. Так, в программе при вычислении объема конуса явно задана константа 3.

Другая константа 3,14 обозначена именем PI и задана в специальном разделе.

Если константа обозначается именем, то она должна быть описана в разделе констант. Описание начинается со служебного слова CONST и имеет следующую форму записи:

CONST _ имя = значение;

Например,

CONST N = 15;

В одном разделе допускается описывать несколько констант. Каждое описание заканчивается символом точки с запятой, а слово CONST для наглядности программы желательно выделить в отдельную строку, например:

```
CONST  
L1 = 100;  
N = 0;  
НОМЕР = 17;
```

В программе ни одна из этих констант не может принимать другое значение.

Переменная может изменять свое значение в ходе выполнения программы. В программе вычисления объемов переменными являются R и H, а также V1 и V2.

Любая переменная, встречающаяся в программе, должна быть описана в разделе пере-

CONST.
— КОНСТАНТА



VAR.
— ПЕРЕМЕННАЯ



менных. Описание начинается со слова VAR и имеет следующую форму записи:

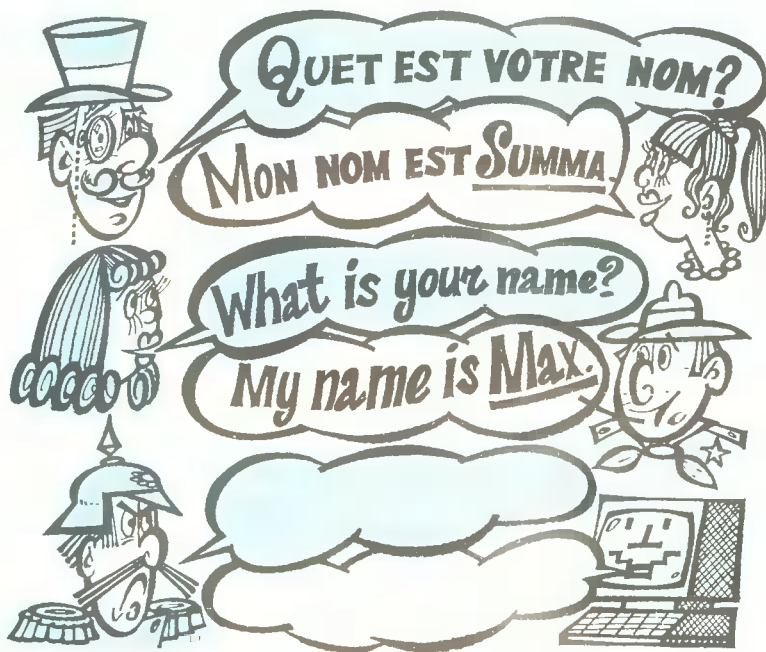
```
VAR _ имя : тип;
```

В одном разделе допускается описание нескольких переменных. В этом случае слово VAR желательно выделить в отдельную строку, например:

```
VAR  
B1 : INTEGER;  
C  : REAL;  
SUMMA : REAL;
```

Если несколько переменных имеют одинаковый тип, то их можно объединять в список. Под списком понимается последовательность элементов (в нашем случае — переменных), разделенных запятыми, например:

```
VAR  
C25, MAX, SUMMA : REAL;  
K, NUMBER, TOM  : INTEGER;
```



ДАЙТЕ МНЕ ТОЧКУ ОПОРЫ

Картина 1. КАК НУЖНО ПРАВИЛЬНО ВЫРАЖАТЬСЯ?

Дока. Сейчас мы приступим к изучению выражений. Слово "выражение" в русском языке имеет несколько значений. Фока, посмотри их, пожалуйста в словаре.

Фока. Ого! Их действительно много:

- а) внешнее проявление чего-нибудь, например "читай с выражением";
- б) внешний вид, например, "Дока, у тебя веселое выражение лица";
- в) оборот речи, например, "Не стесняйся в выражениях, Дока!";
- г) формула, отражающая математические отношения.

Дока. Вот оно! Именно последнее нам и нужно. В языке Паскаль вместо слова "формула" говорят "выражение". Выражение по сравнению с математическим понятием имеет свои особенности:

Выражение строится из констант, переменных, функций и операций над ними. Допускается использование круглых скобок. Частным случаем выражения являются просто константа, переменная или функция.

При составлении выражений необходимо выполнять следующие правила.

1. Записывать все выражение в строку. Двухэтажные выражения, а также верхние и нижние индексы не допускаются. Например, формула

$$\frac{a_1x_1 + a_2x_2}{x_1 - x_2}$$

должна быть записана в виде выражения

$$(A1 \cdot X1 + A2 \cdot X2) / (X1 - X2).$$

2. Использовать скобки только одного вида — круглые. Применение квадратных и фигурных скобок запрещается, так как они имеют другое назначение. Поэтому алгебраической записи

$$a\{b+c[d+e(f+g)]\}$$

в языке Паскаль соответствует выражение

$$A \cdot (B + C \cdot (D + E \cdot (F + G))).$$

В правильно записанном выражении число открывающих скобок всегда должно равняться числу закрывающих. В сложных выражениях и во всех сомнительных случаях рекомендуется ставить скобки. Лишние скобки (правильно поставленные) не влияют на выполнение задачи, но делают ее более наглядной.

3. Нельзя записывать подряд два знака операций. Например, выражение $A+B/-C$ следует записать в виде $A+B/(-C)$.

4. Большие и сложные выражения нужно разбивать на несколько простых, а повторяющиеся действия вычислять отдельно. Так, выражение $(a+bx) - 2(a+bx) + c(a+bx)$ лучше записать следующим образом:

$$Y = A + B \cdot X;$$

$$Z = Y - 2 \cdot Y + C \cdot Y.$$

В языке Паскаль выражение может быть арифметическим и логическим. От этого зависит приоритет выполнения операций. Но об этом мы поговорим позже.

Картина 2. ЗАЧЕМ ИЗОБРЕТАТЬ ВЕЛОСИПЕД?

Фока. В математике приходится часто пользоваться тригонометрическими функциями, например синусом или косинусом. А как их представить в программе?

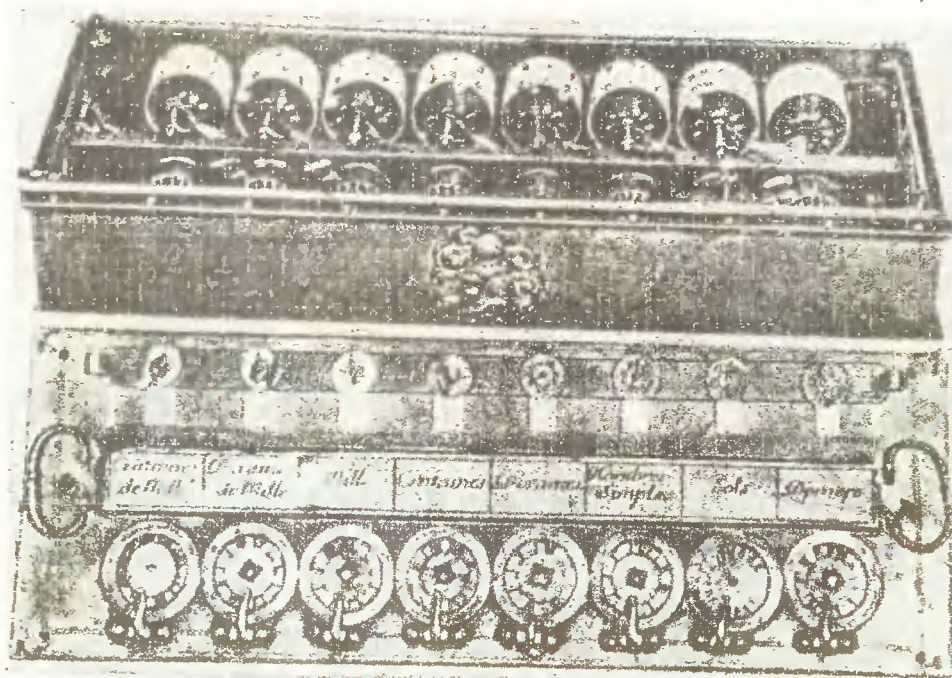
Дока. В языке Паскаль имеется много готовых встроенных функций и не только тригонометрических. Ими можно пользоваться, а не изобретать заново. Такие встроенные функции называются стандартными. Программисту достаточно знать, как записывается функция и аргумент. Аргумент функции заключается в круглые скобки. Аргументом может быть константа, переменная или выражение. Следует заметить, что в тригонометрических функциях синуса и косинуса аргумент должен быть задан только в радианной мере. Если аргумент x задан в градусах, то для его перевода в радианы используется формула $x\pi/180$.

Примеры некоторых встроенных функций:

$SIN(X)$ — вычисляет синус аргумента X , что соответствует математической записи $\sin x$;

$COS(X)$ — вычисляет косинус аргумента X , что соответствует математической записи $\cos x$;

$SQRT(X)$ — вычисляет корень квадратный из аргумента X , что соответствует математической записи \sqrt{x} , например $SQRT(49)$ дает значение 7;



Общий вид арифметической машины Паскаля

SQR(X) — вычисляет квадрат аргумента X, что соответствует математической записи x^2 , например SQR(5) дает значение 25;

TRUNC — определяет целую часть аргумента X, например TRUNC (6.8) и TRUNC(6.1) дают в результате число 6;

ROUND(X) — округляет значение аргумента X до ближайшего целого, например ROUND(6.8) дает значение 7, а ROUND(6.1) значение 6;

TIME — функция без аргумента. При включенном таймере дает значение времени.

Встроенные функции можно использовать в выражениях, например

$$\text{SQR}(X) + \text{SIN}(X-\text{PI}/2).$$

Основной набор встроенных функций представлен в Приложении 5.

Картина 3. ТОЧКА ОПОРЫ

Дока. Исполняемые действия в языке Паскаль записываются в виде операторов. Основными являются следующие: оператор присваивания, операторы ввода и вывода.

Практически любая программа содержит оператор присваивания. С помощью его можно задать переменной некоторое значение. Общая форма записи оператора присваивания имеет вид:

переменная := выражение

Оператор вычисляет значение выражения, стоящего справа от знака присваивания ":", и присваивает полученное значение переменной, стоящей слева. Частным случаем выражения является константа или переменная.

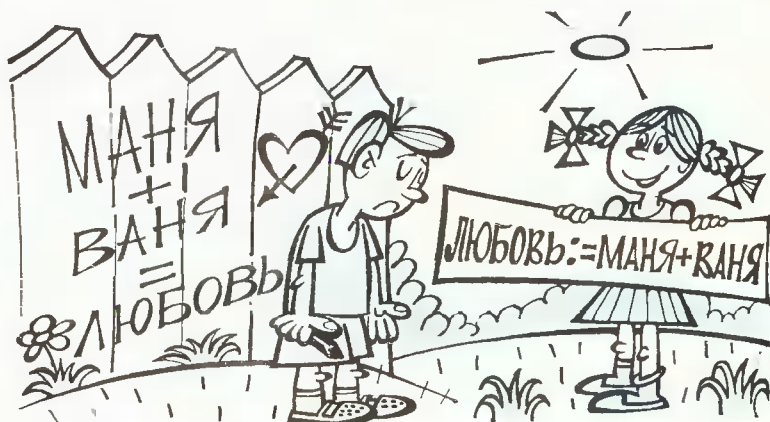
Примеры операторов присваивания:

A:= 5.63; V:= СКОРОСТЬ; ELEMENT:=K;

Y:= SIN(OMEGA*T); R:= 2*(X+Y+Z);

MAX:= M1+M2; N15:=-15.

Фока. Правильно ли я думаю, что все переменные, стоящие в правой части оператора присваивания, должны иметь значения к моменту его выполнения? Например, должны быть определены значения OMEGA и T при вычислении Y или M1 и M2 при вычислении MAX.



Дока. Конечно! Если значения переменных в правой части оператора неизвестны, то и невозможно получить правильный результат.

Оператор присваивания не представляет особой трудности. Однако же подумай, Фока, чему будут равны значения A и SUMMA в результате выполнения следующих операторов:

- | | |
|-----------|---------------------|
| 1) A:=1; | 2) X:=5; |
| B:=-A; | SUMMA:=0; |
| A:=18; | SUMMA:=SUMMA+X; |
| A:=A+2*B; | SUMMA:=SUMMA+X; |
| | SUMMA:=SUMMA+SUMMA; |

Фока. В первом примере сначала будем иметь A=1 и B=-1. Затем переменная A получит новое значение 18, при котором вычисляется выражение A+2·B. Полученное значение вновь присваивается переменной A. Окончательно имеем A=18+2·(-1)=16.

Во втором примере я затрудняюсь дать ответ.

Дока. Твои рассуждения верны и ответ правильный. Что же тебя смущает во втором примере? Рассуждаем точно так же: сначала вычисляется правая часть третьего оператора

SUMMA:=SUMMA+X,

в результате чего получаем SUMMA=0+5=5. С этим значением вычисляется правая часть четвертого оператора

SUMMA:=SUMMA+X.

Таким образом, имеем SUMMA = 5+5=10. Наконец, выполняется правая часть последнего оператора, в результате чего получаем

SUMMA:=10+10=20.

Картина 4. ТО ГУСТО, ТО ПУСТО

Дока. В языке Паскаль можно объединять несколько операторов в одну группу. Такая группа операторов называется составным оператором. Общая форма записи составного оператора имеет вид:

```
BEGIN
  оператор 1;
  оператор 2;
  ...
  оператор n-1;
  оператор n
END
```

В этой конструкции служебные слова BEGIN (начало) и END (конец) называются операторными скобками. Слово BEGIN выполняет функцию открывающей скобки, слово END — закрывающей. Составной оператор представляется как единый оператор. Его можно вставлять в любое место программы, где допускается один оператор.

Любой из операторов составного оператора тоже в свою очередь может быть составным.

Пустой оператор — это оператор, не выполняющий никакого действия. Пустой оператор соответствует отсутствию записи на том месте, где по правилам должен быть какой-нибудь оператор. Например:

```
A:=5;  
;  
B:=7;;;
```

Наличие символа ";" во второй строке и в конце третьей указывает на пустые операторы.

Фока. Я окончательно запутался: где же обязательно нужно ставить точку с запятой, а где можно и не ставить? Поясни, пожалуйста, подробнее.

Дока. У начинающих программистов часто возникает сомнение, где правильно поставить знак точки с запятой? Чтобы ответить на этот вопрос, обратимся к обычному естественному языку. В любом перечне элементов ставится запятая между элементами, например:

A,B,C,D

Если эти элементы объединить в одну группу, заключив их в круглые скобки

(A,B,C,D)

то запятая ставится опять-таки между элементами: после открывающей скобки и перед закрывающей запятая не указывается.

Если эта группа элементов входит в состав другой группы, то запятая ставится и между группами, например:

((A,B,C,D),(K,L,M),E,(R,S))

Подобная же система введена и в языке Паскаль, только роль круглых скобок выполняют операторные скобки BEGIN—END, вместо запятой ставится точка с запятой, а вместо элементов — операторы.

Вот почему после BEGIN и перед END точка с запятой не ставится.

Последняя рассмотренная конструкция естественного языка примет следующую форму записи на языке Паскаль:

```
BEGIN  
  BEGIN  
    оператор A;  
    оператор B;  
    оператор C;  
    оператор D  
  END;  
  BEGIN  
    оператор K;  
    оператор L;  
    оператор M  
  END;  
  оператор E;  
  BEGIN  
    оператор R;  
    оператор S  
  END  
END
```


Во многих версиях языка Паскаль не является ошибкой наличие знака ";" после окончания каждого оператора, в том числе и после END. Это равносильно наличию пустого оператора. Некоторые программисты широко этим пользуются, например:

```
BEGIN
    оператор R;
    оператор S;
END;
```

В дальнейшем мы увидим, что роль операторных скобок могут выполнять не только слова BEGIN и END, но и другие служебные слова.

Картина 5. ДИАЛОГ С КОМПЬЮТЕРОМ

Фока. Вот мы все время что-то вычисляем, присваиваем переменным их значения, а мне бы хотелось просто побеседовать с компьютером. Как это сделать?

Дока. О, это интересный вопрос! Для того чтобы иметь возможность беседовать с компьютером, нужно научиться программировать диалог, т.е. научиться обмениваться информацией с компьютером.

С помощью оператора ввода READ мы передаем исходные данные компьютеру во время выполнения программы. С помощью оператора вывода WRITE компьютер беседует с нами: запрашивает данные и выдает результаты выполнения программы.

Сейчас мы подробно рассмотрим операторы ввода и вывода. Но если тебе, Фока, очень уж не терпится, то посмотри сначала задачи в конце этого раздела, а потом изучай особенности этих операторов.

Следует заметить, что ввод и вывод данных зависят от конкретной версии языка Паскаль и от конкретного типа персонального компьютера.

Мы рассмотрим наиболее распространенный ввод и вывод данных, который совпадает с вводом и выводом на ДВК.

Оператор ввода. Для ввода данных в память ЭВМ предназначен оператор READ, с помощью которого переменные могут получать значения во время выполнения программы.

Оператор ввода имеет вид

READ (a₁, a₂, ..., a_n)

где параметр a₁, a₂, ..., a_n — это переменные, которые последовательно принимают значения, вводимые с клавиатуры (и отражаемые на экране дисплея).

Следует обратить особое внимание: данные вводятся после набора на экране всей программы и запуска ее на выполнение (см. раздел в конце книги "Зри в корень").





Данные могут быть разбиты на строки. Признаком конца строки (конца ввода) является нажатие клавиши ВК.

Если вводятся числа, то они отделяются друг от друга пробелом или несколькими пробелами. Пусть, например, переменные А, В, С должны получать во время выполнения программы следующие значения: А = 5, В = 17, С = -6. Оператор ввода примет вид

READ (A,B,C)

а численные значения можно ввести так:

5 _ 17 _ - 6 ВК

Если вновь повторить запуск программы на выполнение, то можно ввести другие значения, например:

24 _ 3 _ 8 ВК

При этом переменные получают значения

А = 24, В = 3, С = 8

и при этих значениях продолжится выполнение программы.

Один оператор ввода

READ (A,B,C)

полностью эквивалентен трем операторам

READ(A); READ(B); READ(C);

Значения А, В, С вводятся точно так же в одной строке.

Во многих версиях языка Паскаль оператор ввода имеет три назначения:

а) READ (a₁, a₂, ..., a_n) — переменные a₁, a₂, ..., a_n последовательно получают вводимые значения;

б) READLN (a₁, a₂, ..., a_n) — переменные a₁, a₂, ..., a_n последовательно получают вводимые значения, после чего происходит переход на новую строку (если за этим оператором следует снова оператор ввода, то он будет ждать данные в новой строке);

в) READLN — переход на новую строку при вводе данных (поскольку вводимых значений нет, то нужно просто нажать клавишу ВК при выполнении этого оператора).

Последовательно расположенные операторы а) и в) эквивалентны одному оператору б).

Примеры ввода числовых данных:

1		1		1
1	READLN(A,B);	1	READLN(A);	1
1	READ(C);	1	READLN(B);	1
1	1	READ(C);	1
1		1	1
1	24 3	1		1
1	8	1	24	1
1		1	3	1
1		1	8	1

где переменные получают значения $A = 24$, $B = 3$, $C = 8$.

Ввод значений в одной строке

24 _ 3 _ 8 **ВК**

вызовет ошибку. Переменная C не получит значения в первом примере, и переменные B , C — во втором.

В некоторых версиях языка Паскаль допускается вводить числа, разделяя их запятыми или только нажатием клавиши **ВК**.

В языке Паскаль существуют различные типы данных, а не только целые числа. Все они имеют особенности ввода, которые и будут рассмотрены в соответствующих разделах.

Оператор вывода. Для вывода данных из памяти ЭВМ на экран дисплея предназначен оператор вывода, который имеет следующую формулу записи:

WRITE (b_1 , b_2 , ..., b_n)

где параметры b_1 , b_2 , ..., b_n являются переменными, константами, выражениями или строками символов, заключенными в апострофы.

Например, оператор

WRITE ('СУММА ЭЛЕМЕНТОВ =', SUM)

выводит на экран дисплея текст

СУММА ЭЛЕМЕНТОВ =

а затем в этой же строке после знака равенства — значение переменной SUM .

В языке Паскаль допускается использование и других операторов вывода. Оператор вывода без параметров

WRITELN

осуществляет переход на новую строку экрана дисплея. Последующий оператор вывода с параметрами будет выводить данные на новую строку экрана. Оператор вывода без параметров часто используется для пропуска пустых строк.

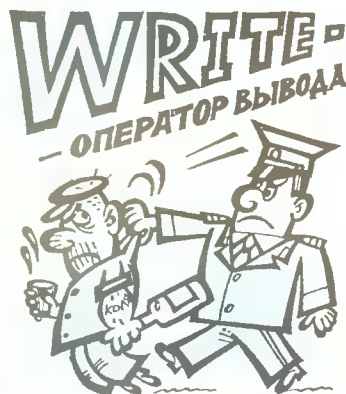
Оператор вывода

WRITELN (b_1 , b_2 , ..., b_n)

осуществляет сначала вывод на экран дисплея значений b_1 , b_2 , ..., b_n , а затем — переход на новую строку. Таким образом, этот оператор эквивалентен двум операторам

WRITE (b_1 , b_2 , ..., b_n);
WRITELN

Оператор **WRITE(A,B,C)** действует точно так же, как три последовательных оператора **WRITE(A)**, **WRITE(B)**, **WRITE(C)**.



Пример использования оператора вывода:

```
-----
I      РЕЗУЛЬТАТОМ ВЫПОЛНЕНИЯ ОПЕРАТОРОВ      I
I
I      WRITE('А. С. ПУШКИН: ');
I      WRITE('ЕВГЕНИЙ ОНЕГИН');
I
I      ЯВЛЯЕТСЯ ОДНА СТРОКА:
I-----
I      А. С. ПУШКИН: ЕВГЕНИЙ ОНЕГИН
I-----
```

```
-----
I      РЕЗУЛЬТАТОМ ВЫПОЛНЕНИЯ ОПЕРАТОРОВ      I
I
I      WRITELN('А. С. ПУШКИН: ');
I      WRITE('ЕВГЕНИЙ ОНЕГИН');
I
I      ЯВЛЯЕТСЯ ДВЕ СТРОКИ:
I-----
I      А. С. ПУШКИН:
I      ЕВГЕНИЙ ОНЕГИН
I-----
```



Для того чтобы результаты выполнения программы были наглядными, удобными и "красивыми" в операторах вывода предусмотрены форматы. Форматы зависят от типа данных и будут рассмотрены в разделах, где они представлены.

Перед вводом данных с помощью оператора *READ* рекомендуется давать поясняющий текст, используя оператор *WRITE*. Этим самым устанавливается полный диалог между пользователем и компьютером. Например, при вводе значений *A, B, C* желательно указать

WRITE ('ВВЕДИТЕ ЗНАЧЕНИЯ A,B,C');

READ(A,B,C)

Перед вводом числовых значений на экране появится сообщение

ВВЕДИТЕ ЗНАЧЕНИЯ A,B,C

после чего можно вводить числа.

Задача 6. Как организовать диалог? Только с помощью операторов ввода и вывода. Вот образец:


```
(-----
1           Д И А Л О Г           1
-----*)
```

```
PROGRAM DI(INPUT,OUTPUT);
```

```
VAR
```

```
  A,B,C : INTEGER;
```

```
BEGIN
```

```
  WRITELN('      ДОБРЫЙ ДЕНЬ, ФОКА ! ');
```

```
  WRITELN('ВВЕДИ, ПОЖАЛУЙСТА, В ОДНОЙ СТРОКЕ ');
```

```
  WRITE('ЧЕРЕЗ ПРОБЕЛ ТРИ ЦЕЛЫХ ЧИСЛА : ');
```

```
  READ(A,B,C);
```

```
  WRITELN;
```

```
  WRITELN('ЧТО ПРОИСХОДИТ ДАЛЕЕ? ');
```

```
    'ЭТИ ЧИСЛА ВЫВЕДЕНЫ В СТОЛБЕЦ ');
```

```
  WRITELN(A:3);
```

```
  WRITELN(B:3);
```

```
  WRITELN(C:3);
```

```
  WRITELN('КАК НУЖНО ИЗМЕНИТЬ ОПЕРАТОРЫ ВЫВОДА,');
```

```
  WRITELN('ЧТОБЫ В ПЕРВОЙ СТРОКЕ БЫЛО ДВА ЧИСЛА,');
```

```
  WRITELN('А ВО ВТОРОЙ - ОДНО ? ');
```

```
  WRITELN;
```

```
  WRITELN('ЖЕЛАЮ УСПЕХА! ДО СВИДАНИЯ !');
```

```
  WRITE('      ТВОЙ ДРУГ ДОКА.')
```

```
END.
```

```
=====
=
=      ДОБРЫЙ ДЕНЬ, ФОКА !
=      ВВЕДИ, ПОЖАЛУЙСТА, В ОДНОЙ СТРОКЕ
=      ЧЕРЕЗ ПРОБЕЛ ТРИ ЦЕЛЫХ ЧИСЛА : 3 5 7
=
=      ЧТО ПРОИСХОДИТ ДАЛЕЕ? ЭТИ ЧИСЛА ВЫВЕДЕНЫ В СТОЛБЕЦ ,
=      3
=      5
=      7
=      КАК НУЖНО ИЗМЕНИТЬ ОПЕРАТОРЫ ВЫВОДА,
=      ЧТОБЫ В ПЕРВОЙ СТРОКЕ БЫЛО ДВА ЧИСЛА,
=      А ВО ВТОРОЙ - ОДНО ?
=
=      ЖЕЛАЮ УСПЕХА! ДО СВИДАНИЯ !
=      ТВОЙ ДРУГ ДОКА.
=====
```



Задача 7. Если устанешь сидеть за дисплеем...

```

(*-----*)
!           СПОРТИВНАЯ ЗАРЯДКА           !
(*-----*)

PROGRAM SPORT(INPUT,OUTPUT);
VAR
  T1,T2,T:REAL;    (* ВРЕМЕНА *)
BEGIN
  READLN;
  WRITELN('*****');
  WRITELN('*      ВКЛЮЧИТЕ ТАЙМЕР И НАЖМИТЕ КЛАВИШУ <ВК>      *');
  WRITELN('*      *');
  WRITELN('*      СДЕЛАЙТЕ НЕСКОЛЬКО ПРИСЕДАНИЙ      *');
  WRITELN('*****');
  READLN;
  T1:=TIME;
  WRITELN('      ПО ОКОНЧАНИИ ЗАРЯДКИ НАЖМИТЕ КЛАВИШУ <ВК>');
  READLN;
  T2:=TIME;
  T:=(T2-T1)*3600;
  WRITELN('*****');
  WRITELN('*      ВЫ ПРИСЕДАЛИ      ',T:9:2,'      СЕКУНД      *');
  WRITELN('*****');
END.

```

Здесь представлен диалог компьютера и пользователя. На экране дисплея появляется информация, которая указана в операторах вывода. Стандартная функция TIME определяет текущее время. T1 — время до начала зарядки, T2 — после ее окончания, T — время выполнения зарядки, выраженное в секундах.

Первый оператор READLN нужен только для компьютеров типа ДБК. При встрече следующего оператора без параметров READLN происходит останов машины. После нажатия клавиши ВК будет продолжено выполнение программы.

ВИЗИТНАЯ КАРТОЧКА ДАННЫХ

Картина 1. ХОРОШИЙ ИЛИ ПЛОХОЙ ХАРАКТЕР У ДАННЫХ?

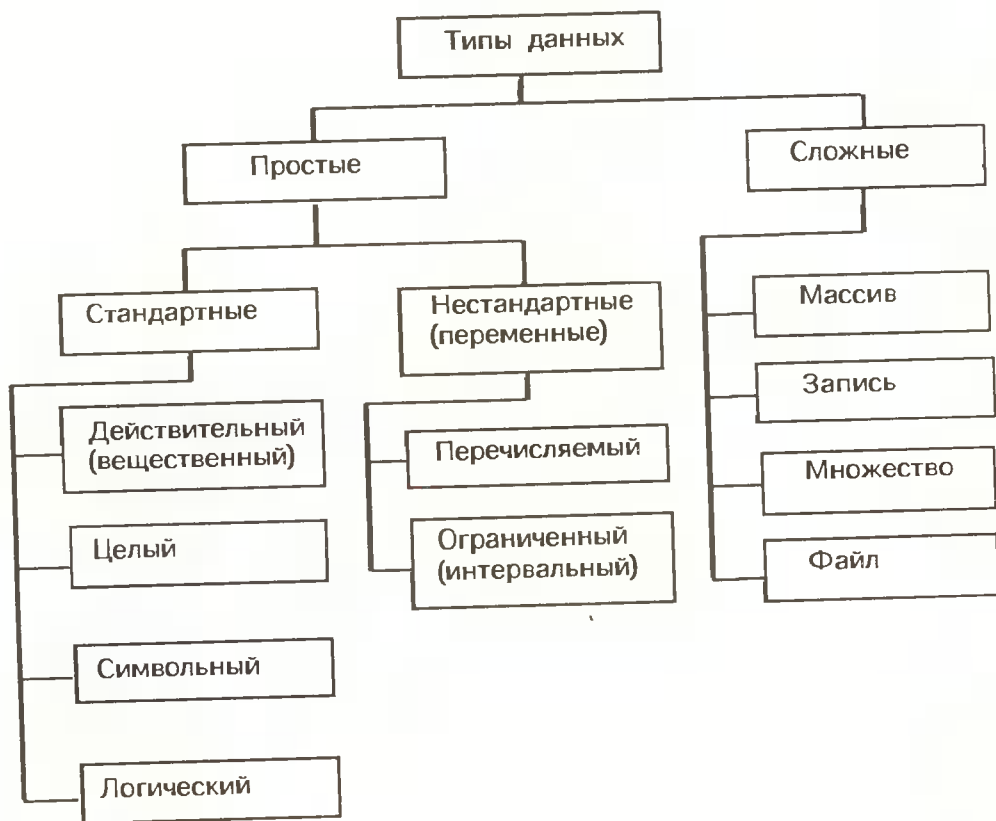
Дока. Когда мы рассматривали программу на языке Паскаль в общем виде, то сравнили ее с пьесой. После заголовка пьесы представляются действующие лица и даются особенности их внешности, характера, которые влияют на поведение. Подобно этому в программе после заголовка указываются все действующие данные и их характеры.

Фока. Получается, что если я указал N:REAL, то REAL — это характер переменной N и он тоже влияет на ее поведение?

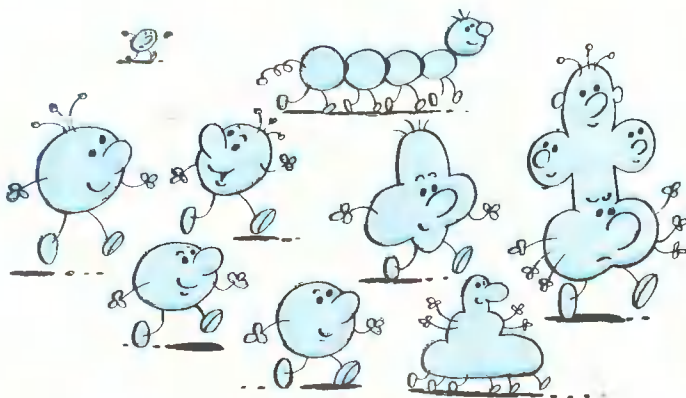
Дока. Именно так! Только в пьесе характеры действующих лиц могут быть самыми разнообразными, а в языке Паскаль имеется определенный набор типов характеров, говорят — типов данных.

Все типы данных можно разделить на простые и сложные.

Типы данных в языке Паскаль:



Тип данных определяет возможные значения констант, переменных, функций, выражений, а также операции, которые могут выполняться над ними.



Картина 2. ДАННЫЕ ЦЕЛОГО ТИПА

Дока. Данные целого типа используются в тех случаях, когда какую-то величину нужно представить абсолютно точно, например число живых существ или количество предметов, элементов. К данным целого типа относятся константы и переменные.

Константа целого типа — это десятичное число, записанное без точки. Если константа отрицательная, то перед ней должен стоять знак "-", а если положительная, то знак "+" можно не указывать.

Примеры констант целого типа: 6, -140, +357, 0, 9999, 32765.

Мы уже говорили о том, что константы могут быть обозначены именем. В этом случае они задаются в разделе CONST, например:

```
CONST
  K = 15;
  NULL = 0;
  P1 = -46;   P2 := +46;
  ВИСОТА = 120;
```

Переменная целого типа принимает значение целого десятичного числа. Описание переменной имеет тип INTEGER (целый), например:

```
VAR
  A : INTEGER;
  SUMMA : INTEGER;
  B1, B2, B3 : INTEGER;
```

В разделе операторов указанные переменные должны принимать значения целых десятичных чисел, например:

```
A:=25; SUMMA:=0;
```



Данные целого типа принимают значения в диапазоне от -32768 до +32767. В памяти машины значения представлены точно.

Операции. Над данными целого типа можно выполнять следующие арифметические операции:

- +** сложение (результат сложения — целое число);
- вычитание (результат вычитания — целое число);
- *** умножение (результат умножения — целое число);
- DIV** деление с отбрасыванием дробной части (получение целого частного при делении целого данного на целое);
- MOD** получение целого остатка при делении целого данного на целое.

Операция MOD часто используется для определения, делится ли целое число X без остатка на два, т.е. является ли X четным числом. С помощью операции $X _ MOD _ 2$ вычисляется остаток. Если он равен нулю, то число X четное, а если имеется остаток, то нечетное. Точно так же можно определить, кратно ли какое-то число трем, четырем и т.д.

К данным целого типа можно применять встроенные функции.

Пусть, например, A, B, C — переменные целого типа, принимающие значения A=25, B=5, C=3. Тогда:

Действие	Результат	Действие	Результат
A+5	30	B MOD A	5
B-A	-20	A MOD C	1
C*15	-45	A DIV C	-8
A DIV 7	3	-25 DIV -3	8
A MOD 7	4	-25 MOD -3	-1

Ввод и вывод. При вводе данных целого типа с помощью оператора READ, числа разделяются пробелом, пробелами или нажатием клавиши BK. Пробелы и BK игнорируются перед числом.

Пусть, например, переменные A, B, C будут целого типа. Тогда во всех трех случаях ввода получим один и тот же результат (A=8, B=17, C=4).

I		I		I		I
I	READ(A,B,C);	I	READ(A,B,C);	I	READ(A,B,C);	I
I	I	I	I
I		I		I		I
I	8 17 -4	I	8 17	I	8	I
I		I	-4	I	17	I
I		I		I	-4	I
I		I		I		I

Вывод данных целого типа допускается с форматом и без него. При бесформатном выводе на изображение целого числа отводится 13 позиций. Если число занимает меньше позиций, то перед числом выводится соответствующее количество пробелов. Например, вывод значений тех же переменных A, B, C с помощью оператора WRITE(A,B,C), имеет следующий вид:

----- 8 ----- 17 ----- -4 -----
 13 13 13

Для наглядности результатов предусмотрены форматы. Формат указывается в операторе WRITE вслед за выводимым данным через двоеточие. Для данных целого типа формат имеет вид

x:m,

где x — данное целого типа (константа, переменная, выражение), m — поле выводимого числа, включая знак числа. В качестве m могут быть константа, переменная или выражение целого типа. Если формат отведен больше, чем цифр в числе, то перед числом выводится соответствующее количество пробелов. Для вывода значений рассмотренных переменных A, B, C можно указать, например, следующие форматы:

I	-----	I		I
I	WRITE(A:4, B:5, C:5)	I	VAR K,N : INTEGER;	I
I	-----	I	I
I	CONST N = 4;	I		I
I	M = 5;	I	N:= 4;	I
I	-----	I	K:= 5;	I
I		I	WRITE(A:N, B:K, C:5)	I
I	WRITE(A:N, B:M, C:M)	I		I
I	-----	I		I

Во всех трех случаях на экране дисплея появится строка

___ 8 ___ 17 ___ -4

Фока. Диапазон целых чисел ограничен. А что будет, если ввести число, выходящее за диапазон, например 985467?

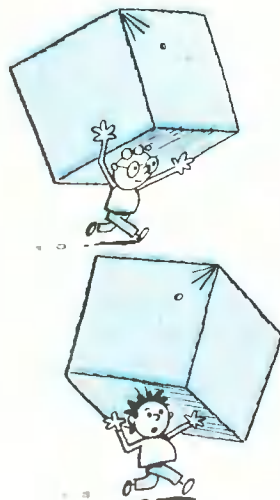
Дока. Компьютер ответит BAD INTEGER. Это означает, что вводимое число имеет плохой тип, не соответствующий целому.

Задача 8. Операцию MOD почему-то многие путают. А что здесь непонятного? Делим целое число на целое и получаем целый остаток. Вот и все!

```
(*-----*)
! ДЕЛЕНИЕ ЦЕЛЫХ ЧИСЕЛ !
(*-----*)
PROGRAM DEL EN (INPUT, OUTPUT);
VAR
  A: INTEGER; (* ДЕЛИМОЕ *)
  B: INTEGER; (* ДЕЛИТЕЛЬ *)
  Y: INTEGER; (* ЦЕЛОЕ ЧАСТНОЕ *)
  Z: INTEGER; (* ЦЕЛЫЙ ОСТАТОК *)
BEGIN
  WRITE('ВВЕДИТЕ ЗНАЧЕНИЯ A,B: ');
  READLN(A,B);
  Y:=A DIV B;
  Z:=A MOD B;
  WRITELN('ЧАСТНОЕ=', Y:3);
  WRITE('ОСТАТОК=', Z:3)
END.
=====
=
= ВВЕДИТЕ ЗНАЧЕНИЯ A,B: 59 17 =
= ЧАСТНОЕ= 3 =
= ОСТАТОК= 8 =
=====
```


Задача 9. Вычислить объемы двух кубов, одного со стороной A1, другого — A2.

```
(*-----*)
!           ОБЪЕМ КУБА           !
(*-----*)
PROGRAM KUB2(INPUT,OUTPUT);
  VAR
    A1,V1: INTEGER; (* СТОРОНА И ОБЪЕМ 1-ГО КУБА *)
    A2,V2: INTEGER; (* СТОРОНА И ОБЪЕМ 2-ГО КУБА *)
BEGIN
  WRITE('ВВЕДИТЕ ЗНАЧЕНИЕ A1: ');
  READLN(A1);
  V1:=A1*A1*A1;
  WRITELN('ОБЪЕМ 1-ГО КУБА =', V1:5);
(*-----*)
  WRITE('ВВЕДИТЕ ЗНАЧЕНИЕ A2: ');
  READLN(A2);
  V2:=A2*A2*A2;
  WRITELN('ОБЪЕМ 2-ГО КУБА =', V2:5)
END.
=====
=      ВВЕДИТЕ ЗНАЧЕНИЕ A1: 5      =
=      ОБЪЕМ 1-ГО КУБА = 125      =
=      ВВЕДИТЕ ЗНАЧЕНИЕ A2: 13    =
=      ОБЪЕМ 2-ГО КУБА = 2197     =
=====
```



Конечно, если кубов несколько, то программу можно составить проще, используя циклы. Но об этом позже.

Картина 3. ДАННЫЕ ДЕЙСТВИТЕЛЬНОГО ТИПА

Дока. Данные действительного (вещественного) типа используются значительно чаще, чем целого типа. Они необходимы в тех случаях, когда числовые значения могут содержать дробные части.

Константа действительного типа может быть представлена в языке Паскаль в двух видах: числом с фиксированной и плавающей точкой.

Число с фиксированной точкой изображается десятичным числом с дробной частью (дробная часть может быть нулевой). Дробная часть отделяется от целой с помощью точки, например

127.3, 25.0, -16.003, 200.59, 1.00, 0.54.

В математике для изображения очень больших и малых чисел используется запись числа с десятичным порядком. Например, число 250000000 можно записать в виде $25 \cdot 10^7$, где 7 — порядок числа, или в виде $2,5 \cdot 10^8$, где 8 — порядок числа.

Другой пример: число 0,00000005 можно записать как $5 \cdot 10^{-8}$, где порядком является число -8.

В языке Паскаль также можно изображать числа с порядком. Они имеют вид $mEр$. Здесь m называется мантиссой, а $р$ — порядком числа. Символ E является признаком записи числа с десятичным порядком. В качестве m могут быть целые числа и действительные числа с фиксированной точкой. В качестве $р$ — только целые числа. Как мантисса, так и порядок могут содержать знаки "+" или "-".

Число, представленное с порядком, называется с плавающей точкой. Примеры чисел с плавающей точкой:

Математическая запись	Запись в языке Паскаль
$4 \cdot 10^{-5}$	4E-5
$0,62 \cdot 10^4$	0.62E+4
$-10,8 \cdot 10^{12}$	-10.8E12
$-20 \cdot 10^{-3}$	-20E-3

Следует обратить внимание на то, что в языке Паскаль знак умножения не ставится.

Рассмотрим, например, числа с плавающей точкой:

0.52E+3, 5.2E+2, 52E+1, 520E0, 5200E-1, 52000E-2.

Все эти записи представляют одно и то же число 520. Перемещая положение десятичной точки в мантиссе (точка "плывет") и одновременно изменяя величину порядка, можно выбрать наиболее подходящее представление числа в определенной задаче.

Значение констант действительного типа как с фиксированной, так и с плавающей точкой можно задать в разделе описания констант, например

```
CONST
  E = -1.602E-19;
  NA = 6.022E23;
  PI = 3.14;
  SUMMA = -5.6;
```

Переменная действительного типа принимает значение числа с фиксированной или плавающей точкой. Описание переменной имеет тип REAL (действительный).

Пусть имеется описание действительных переменных

```
VAR
  U, I, R : REAL;
  Z1 : REAL;
  СКОРОСТЬ : REAL;
```

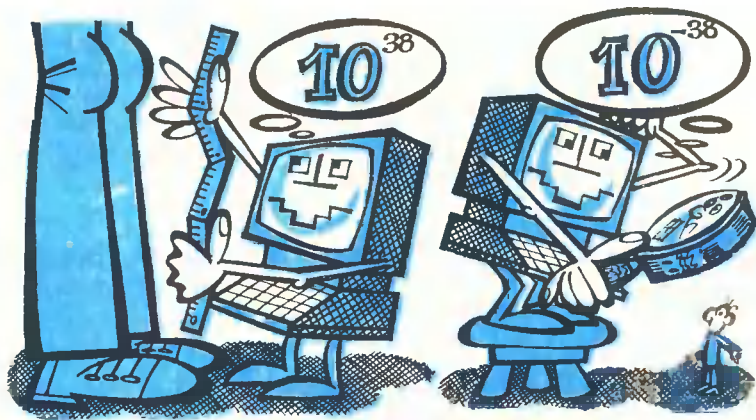
тогда в разделе операторов можно, например, указать

```
Z1 := -0.03E14;
СКОРОСТЬ := 12.6;
I := 5;
```

В последнем случае целое число 5 автоматически преобразуется к действительному типу 5.0 для присвоения переменной I.

Самое малое по модулю действительное число (не считая нуля) равно 10^{-38} , а самое большое равно 10^{+38} . Несмотря на большой диапазон по сравнению с целыми числами действительные числа обладают недостатком: они представлены в памяти ЭВМ с некоторым приближением. Как правило, верными могут быть первые шесть или семь цифр в числе. Это объясняется неточным переводом многих действительных чисел из десятичной системы в двоичную. При вычислениях погрешность может накапливаться и за этим нужно следить самому программисту.





Операции. Над данными действительного типа можно выполнять следующие операции: + сложение, — вычитание, * умножение, / деление.

Если оба операнда действительного типа, то и результат операции тоже действительного типа.

Допускается, что из двух данных, участвующих в операции, одно может быть целого типа. Результат при этом все равно будет действительного типа. К действительным данным можно применять встроенные функции.

Пусть, например, переменные X, Y, Z — действительного типа, тогда возможны выражения:

$2 \cdot X + Y;$
 $\text{SIN}(X + Z/4);$
 $\text{SQR}(X + Y + Z);$

$X/Z + 5;$
 $\text{SQRT}(Y + 5.7);$
 $Y \cdot Y \cdot Y.$

Поскольку в языке Паскаль нет операции возведения в степень, то при необходимости ее использования применяют стандартные функции. Например, a^x заменяют выражением

$\text{EXP}(X \cdot \text{LN}(A)),$

где EXP и LN — встроенные функции (экспонента и натуральный логарифм). Значение A должно быть больше нуля.

Ввод и вывод. При вводе данных действительного типа с помощью оператора READ, числа разделяются пробелом, пробелами или возвратом каретки BK. Пробелы и BK игнорируются перед числом. Допускается ввод чисел с фиксированной и плавающей точкой, а также ввод целых чисел, которые автоматически преобразуются в действительные.

Примеры:

I	READ(R, S, T);	I	READ(R, S, T);	I
I	I	I
I	-0.18 315.24E-4 6	I	-0.18	I
I		I	315.24E-4	I
I		I	6	I
I		I		I

В обоих случаях переменные получают одни и те же значения $R = -0.18$, $S = 315.24 \cdot 10^{-4}$, $T = 6$.

Вывод действительных данных допускается с форматом и без него. Если отсутствует формат, то число выводится с плавающей точкой — мантисса и порядок. На изображение числа отводится 13 позиций, при этом в целой части

мантиссы присутствует только одна значащая цифра, в дробной части — шесть цифр, а порядок числа принимает соответствующее значение и на него отводится с учетом знака три позиции. Так, при выводе значений рассмотренных переменных с помощью оператора `WRITE(R,S,T)`, получим

$\underbrace{-1.800000E-01}_{13} \quad \underbrace{3.152400E-02}_{13} \quad \underbrace{6.000000E+00}_{13}$

Для наглядности результатов предусмотрены форматы. Формат указывается в операторе `WRITE` вслед за выводимым данным через двоеточие. Для данных действительного типа формат имеет вид

`x:m:n`,

где `x` — выводимое данное действительного типа (константа, переменная, выражение); `m` — общее поле выводимого числа (включая знак числа, целую часть, точку и дробную часть); `n` — поле дробной части. В качестве `m` и `n` могут быть целые константы, переменные, выражения. При использовании форматов число выводится в форме с фиксированной точкой.

Если формат указан больше, чем необходимо для изображения числа, то перед целой частью располагаются избыточные пробелы, а после дробной части — нули.

Так, для вывода значения `R` равного `-0.18` достаточно указать формат `WRITE(R:5:2)`. Использование формата `WRITE(R:10:4)` приводит к выводу значения

`-0.1800`

Для вывода значений рассмотренных переменных `R,S,T` можно выбрать, например, следующие форматы:

```

-----
I      WRITE( R:5:2, S:8:6, T:3:1 )      I
I      .....                          I
I      .....                          I
I      -0.180.0315246.0                  I
-----

```

```

-----
I      WRITE( R:5:2, S:10:6, T:5:1 )      I
I      .....                          I
I      .....                          I
I      -0.18  0.031524  6.0              I
-----

```

Если действительное число представлено в машине приближенно, то при выводе оно округляется. Рассмотрим, например, значения `X` и вывод их с помощью оператора `WRITE(X:6:4)`:

Значение X	Результат
0.99999	1.0000
0.99919	0.9992
0.99914	0.9991
0.99915	0.9992

Фока. Мне непонятно вот что: с одной стороны, мы говорим, что диапазон действительных чисел очень большой, а с другой стороны, что верными могут быть представлены только шесть или семь цифр в числе. Как это понимать?

Допустим я хочу выполнить действия с числом 576984379. В каком виде мне его записать?

Дока. Это число можно представить в виде 5.769844E+8. Седьмая цифра округлена, а остальные не учитываются.

Фока. И еще вопрос. Что значит "приближенное представление числа в памяти ЭВМ"?

Дока. Пусть дано $X = -0.1$. Если выполнить многократное сложение $X + X + X + X + X + X$, то вместо числа 0.6 получится число 0.599999. Понятно?

Фока. Вот теперь понятно. Давай посмотрим программирование задач с действительными данными. Я готов.



Картина 4. ДВА ПЛЮС ТРИ

Дока. Мы рассмотрели понятие "выражение". Если операндами выражения являются данные целого или действительного типов, то выражение называется арифметическим.

В арифметическом выражении принят следующий приоритет операций:

- 1) вычисление функций;
- 2) *, /, DIV, MOD;
- 3) +, -.

Если аргумент функции представляет собой выражение, то сначала вычисляется значение этого выражения. Например, в выражении

$$\text{COS} (X+PI/4)$$

сначала вычисляется аргумент $X+PI/4$, а затем функция COS.

Выражение

$$2*A*B+C/D-F$$

будет выполняться в следующем порядке:

- 1) $2*A$,
- 2) $2*A*B$,
- 3) C/D
- 4) $2*A*B+C/D$,
- 5) $2*A*B+C/D-F$.

В арифметическом выражении можно использовать круглые скобки. Если выражений в скобках несколько и они вложены друг в друга, то сначала выполняются действия в самых внутренних скобках. Например, выражение

$$A*(B+C*(D+E*(F+G)))$$

выполняется в следующей последовательности:

- | | |
|------------------|----------------------------|
| 1) (F+G) | 4) $C*(D+E*(F+G))$, |
| 2) $E*(F+G)$, | 5) $(B+C*(D+E*(F+G)))$, |
| 3) $D+E*(F+G)$, | 6) $A*(B+C*(D+E*(F+G)))$. |

Если операции, следующие непосредственно одна за другой, обладают одинаковым старшинством, то они выполняются в том же порядке, в каком и записаны. Например, в выражении $A*B/C*D/E$ последовательность операций будет такой:

- | | |
|--------------|------------------|
| 1) $A*B$, | 3) $A*B/C*D$, |
| 2) $A*B/C$, | 4) $A*B/C*D/E$. |

Фока. Можно ли в одном арифметическом выражении использовать данные и целого и действительного типов?

Дока. Конечно, нежелательно, чтобы в одном выражении использовались данные различного типа. Однако в некоторых версиях языка Паскаль допускается наличие в одной операции операндов как целого, так и действительного типов. Результат такой операции будет действительного типа. Следует заметить, что операцию деления "/" нельзя применять к данным целого типа. В операторе присваивания нужно следить за тем, чтобы тип выражения соответствовал типу переменной, стоящей слева от операции присваивания.

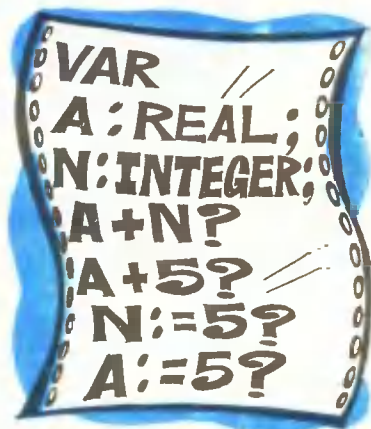
ПРОВЕРЬ СЕБЯ:

25 MOD 3*4 =4
 25 DIV 3*4 =32
 5*5 DIV 3*4 =32
 5+5 MOD 3-4 =3

Задача 10. Вычислить сопротивление цепи, состоящей из резисторов, соединенных:

последовательно $R_{\text{пос}} = R_1 + R_2$;

параллельно $R_{\text{пар}} = \frac{R_1 R_2}{R_1 + R_2}$.



(*-----*)
 ! СОПРОТИВЛЕНИЕ ЦЕПИ !
 (*-----*)

```
PROGRAM RESIST(INPUT,OUTPUT);
VAR
  R1,R2:REAL; (* СОПРОТИВЛЕНИЯ РЕЗИСТОРОВ *)
  RPOS:REAL; (* СОПР-Е ПОСЛЕДОВ-УХ РЕЗИСТ-ОВ *)
  RPAR:REAL; (* СОПР-Е ПАРАЛЛЕЛЬНЫХ РЕЗИСТ-ОВ *)
BEGIN
  WRITE('ВВЕДИТЕ ЗНАЧЕНИЯ R1 И R2 : ');
  READLN(R1,R2);
  WRITELN;
  WRITELN('          ПОСЛЕДОВАТЕЛЬНАЯ ЦЕПЬ          ');
  WRITELN('          -----I-----I-----I----- ');
  WRITELN('          RPOS:=R1+R2; ');
  WRITELN('          ':13, 'R=', RPOS:7:2);
  WRITELN;
```



```

WRITELN('          ПАРАЛЛЕЛЬНАЯ ЦЕПЬ          ');
WRITELN('          -----I-----I-----I');
WRITELN('          -----I-----I-----I');
WRITELN('          -----I-----I-----I');
RPAR:=R1*R2/(R1+R2);
WRITE('  ':14, 'R=', RPAR:7:2)
END.
=====
=
=      ВВЕДИТЕ ЗНАЧЕНИЯ  R1 И R2 : 20  30      =
=
=      ПОСЛЕДОВАТЕЛЬНАЯ ЦЕПЬ                  =
=      -----I-----I-----I-----I----- =
=      R=  50.00                                =
=
=      ПАРАЛЛЕЛЬНАЯ ЦЕПЬ                      =
=      -----I-----I-----I-----I----- =
=      R=  12.00                                =
=====

```

Задача 11. Вычислить значения Y и R , если $Y=\sqrt{X}$, $R=\sin(X + \pi/4)$.
 где X — действительное данное. Кроме того, найти остаток от деления
 целого K на целое N .

```

'/*-----*
I      АРИТЕТИЧЕСКИЕ ВЫРАЖЕНИЯ      I
*-----*/

PROGRAM EX(INPUT,OUTPUT);
CONST PI=3.14;
VAR  Y,R,X : REAL;
      K, N : INTEGER;
      ОСТАТОК : INTEGER;

BEGIN
  WRITELN('          ДОБРЫЙ ДЕНЬ ! ');
  WRITELN(' ВВЕДИТЕ ЗНАЧЕНИЯ  X,K,N : ');
  READ( X,K,N );
  Y:= SQRT( X );
  R:= SIN( X + PI/4 );
  ОСТАТОК:= K MOD N ;

  WRITELN;
  WRITELN('РЕЗУЛЬТАТЫ ВЫЧИСЛЕНИЙ :');
  WRITELN('Y=', Y:6:2 );
  WRITELN('R=', R:6:2 );
  WRITELN('ОСТАТОК =', ОСТАТОК :3 );
  WRITE('          ДО СВИДАНИЯ ! ')
END.

=====
=
=      ДОБРЫЙ ДЕНЬ !                          =
=      ВВЕДИТЕ ЗНАЧЕНИЯ  X,K,N : ');          =
=      6.25  17  3                             =
=
=      РЕЗУЛЬТАТЫ ВЫЧИСЛЕНИЙ :');            =
=      Y=  2.50                                =
=      R=  0.68                                =
=      ОСТАТОК =  3                             =
=      ДО СВИДАНИЯ !                          =
=====

```




Задача 12. Подсчитать число молекул в комнате по формуле

$$N = \frac{\rho_0 V N_a}{M},$$

где N_a — постоянная Авогадро ($6,022 \cdot 10^{23}$ моль $^{-1}$); M — средняя молярная масса воздуха ($29 \cdot 10^{-3}$ кг/моль); ρ_0 — плотность воздуха ($1,29$ кг/м 3); V — объем (м 3).

```

(*-----
  I      ЧИСЛО МОЛЕКУЛ В КОМНАТЕ      I
  -----*)
PROGRAM МОЛЕК(INPUT,OUTPUT);
CONST
  M=29E-3;      (* МОЛЕКУЛЯРНАЯ МАССА *)
  RO=1.29;      (* ПЛОТНОСТЬ ВОЗДУХА *)
  NA=6.022E23;  (* ЧИСЛО АВОГАДРО *)
VAR
  A,B,C:REAL;   (* РАЗМЕРЫ КОМНАТЫ *)
  V:REAL;       (* ОБЪЕМ КОМНАТЫ *)
  N:REAL;       (* ЧИСЛО МОЛЕКУЛ *)
BEGIN
  (* N=MB/MO, ГДЕ
     MB - МАССА ВОЗДУХА,
     MO - МАССА ОДНОЙ МОЛЕКУЛЫ,
     MB=RO*V, MO=M/NA
     / В А)
     C !
     ! A
     -----
     / В А)

  WRITE('КАКОВА ДЛИНА ВАШЕЙ КОМНАТЫ? ');
  READLN(A);
  WRITE('КАКОВА ШИРИНА ВАШЕЙ КОМНАТЫ? ');
  READLN(B);
  WRITE('КАКОВА ВЬСОТА ВАШЕЙ КОМНАТЫ? ');
  READLN(C);
  V:=A*B*C;
  N:=RO*V*NA/M;
  WRITELN;
  WRITELN('ОБЪЕМ ВАШЕЙ КОМНАТЫ = ', V:5:1);
  WRITE('ЧИСЛО МОЛЕКУЛ В ВАШЕЙ КОМНАТЕ = ', N)
END.
```



```

=====
=
=   КАКОВА ДЛИНА ВАШЕЙ КОМНАТЫ? 5
=   КАКОВА ШИРИНА ВАШЕЙ КОМНАТЫ? 4
=   КАКОВА ВЫСОТА ВАШЕЙ КОМНАТЫ? 3
=
=   ОБЪЕМ ВАШЕЙ КОМНАТЫ = 60.0
=   ЧИСЛО МОЛЕКУЛ В ВАШЕЙ КОМНАТЕ = 1.607251E+27
=
=====

```

Картина 5. ДАННЫЕ ЛОГИЧЕСКОГО ТИПА

Дока. Логический тип данных часто называют булевым по имени английского математика Д.Буля, создателя особой области математики — математической логики. Основу математической логики составляют две константы: 1 (истина) и 0 (ложь). Логические данные широко используются при сравнении величин. Результат сравнения может оказаться истинным или ложным.

П р и м е р ы:

- | | |
|----------------------|--|
| а) $a^2 = b^2 + c^2$ | истинно при $a = 5, b = 4, c = 3$,
ложно при $a = b = c = 2$; |
| б) $x^2 < 0$ | всегда ложно; |
| в) $a > b + 3$ | истинно при $a = 10$ и $b = 2$,
ложно при $a = 2$ и $b = 10$. |

В языке Паскаль тоже имеются две логические константы: TRUE (истина) и FALSE (ложь). Их можно использовать в явном виде или обозначать именем в разделе CONST, например:

```

CONST
  T = TRUE;
  F = FALSE;
  LC = TRUE;

```

Логическая переменная может принимать значение TRUE или FALSE. Описание переменной имеет тип BOOLEAN (логический), например:

```

VAR
  LOGIC : BOOLEAN;
  A : BOOLEAN;
  L1, L2, L3 : BOOLEAN;

```

В разделе операторов такой переменной можно присвоить значение логической константы, например,

```
L1 := TRUE; L3 := L1; L2 := FALSE.
```

Если операндами отношения будут арифметические данные, то получим логическое значение: истинно выражение или ложно.

Операции. В языке Паскаль для сравнения данных предусмотрены следующие операции отношений:

- < меньше,
- <= меньше или равно,
- = равно,

BOOLEAN
— ЛОГИЧЕСКИЙ ТИП

TRUE — ИСТИНА

FALSE
— ЛОЖЬ

<> не равно,
> больше,
>= больше или равно.

П р и м е р ы :

а) отношение $6 < 2$ (читается "шесть меньше двух?") дает ложный результат FALSE;

б) отношение $A < > B$ (читается "A не равно B?") даст истинный результат TRUE при $A=10$ и $B=4$;

в) отношение $C = 4 * T - R$ дает ложный результат FALSE при $C = 6$, $T = 2$, $R = 3$.

Отношения можно использовать в правой части оператора присваивания. Например, при описании

```
VAR L1, L2: BOOLEAN;
```

можно указать

```
L1:= A<>B;
```

```
L2:= C = 4*T-R;
```

Переменная L1 получит значение TRUE, а переменная L2 - значение FALSE.

В языке Паскаль, как и в математической логике, имеются специальные логические операции:

OR — логическое сложение (ИЛИ),

AND — логическое умножение (И),

NOT — логическое отрицание (НЕ).

Логическое сложение дает ложный результат только в том случае, если оба операнда ложные. Если хотя бы один операнд истинный, то результат будет истинным.

B	OR	C
ложно, если		
B — FALSE		
C — FALSE		

B	AND	C
истинно, если		
B — TRUE		
C — TRUE		

Например, результат операции

```
(A>5) OR (C=3)
```

ложный при $A=2$, $C=1$; истинный: при $A=10$, $C=1$; $A=10$, $C=3$; $A=2$, $C=3$.

Логическое умножение дает истинный результат только в том случае, если оба операнда истинные. Если же хотя бы один операнд ложный, то результат также будет ложным.

Например, результат операции

```
(A>5) AND (C=3)
```

истинный: при $A=10$, $C=3$; ложный: при $A=2$, $C=3$; $A=10$, $C=1$; $A=2$, $C=1$.

NOT B

ИСТИННО, если

B — FALSE

Операция отрицания дает ложный результат, если операнд истинный, и наоборот дает истинный результат, если операнд ложный.

Например, результат операции

NOT (A=10)

истинный при A=2; ложный при A=10.

Логический тип определяется таким образом, что FALSE < TRUE.

Ввод и вывод. В языке Паскаль нет возможности ввода логических данных с помощью оператора READ. Однако предусмотрен вывод при использовании оператора WRITE. В этом случае для логических значений TRUE и FALSE автоматически отводится по шесть позиций: две — перед словом TRUE и одна — перед FALSE (в некоторых версиях языка Паскаль для логических значений отводится соответственно 4 или 5 позиций).

Картина 6. ИСТИНА ИЛИ ЛОЖЬ?

Фока. Мне нравятся логические операции и вообще мне нравится логически рассуждать. Можно ли объединять отдельные отношения в какие-то логические выражения?

Дока. Да. Логическое выражение строится из логических данных, логических операций и операций отношений. В операциях отношения могут участвовать арифметические и логические выражения, а также символьные данные. Результатом логического выражения является значение TRUE или FALSE.

В логическом выражении принят следующий приоритет операций:

- 1) NOT (выполняется в первую очередь);
- 2) * , / , DIV, MOD; AND;
- 3) + , - , OR;
- 4) < , <= , = , <> , >= , >.

Операции, указанные в одной строке, имеют одинаковый приоритет.

В логическом выражении допускается использование только круглых скобок. При наличии скобок сначала выполняются действия в них (в первую очередь, в самых внутренних), а затем вне скобок.

В круглые скобки обязательно заключаются части выражения, стоящие слева и справа от логических операций AND и OR.

Пример. Определить результат логического выражения

(A>3) AND (B=A+6) OR NOT (C=4)

ли A=2, B=8, C=5.

Порядок выполнения операций следующий:

а) выполняется операция сравнения A>3 в первых скобках, результат ее FALSE (так как 2<3);

б) выполняются действия во вторых скобках в соответствии с приоритетом — сначала вычисляется значение A+6, а затем сравнивается значение B с значением A+6, и в результате получаем TRUE (так как 8=8);

- в) выполняется операция сравнения $C=4$ в третьих скобках, результат FALSE (так как $5 \neq 4$);
- г) выполняется операция NOT ($C=4$), равная NOT FALSE и результат TRUE;
- д) выполняется операция AND над первыми и вторыми скобками (FALSE AND TRUE), результат ее FALSE;
- е) выполняется операция OR над выражениями слева и справа от н (FALSE OR TRUE) и результат ее TRUE.

Таким образом, получаем окончательный результат логического выражения — TRUE.

Задача 13. Определить значения логических переменных L1, L2, L3. Исходные данные представлены в программе.

```

(*-----*)
I          ЛОГИКА          I
(*-----*)

PROGRAM LOG2(INPUT,OUTPUT);
CONST T=TRUE;              (* ЛОГИЧЕСКАЯ КОНСТАНТА *)
VAR
  L1,L2,L3 : BOOLEAN;      (* ЛОГИЧЕСКИЕ ПЕРЕМЕННЫЕ *)
BEGIN
  L1:= FALSE;
  L2:= T AND L1 ;
  L1:= (NOT L2) OR (NOT L1);
  L3:= L1 AND L2 AND T;
  WRITE('L1=',L1, ' ':4,'L2=',L2, ' ':4,'L3=',L3 )
END.

=====
=
= L1= TRUE    L2= FALSE    L3= FALSE =
=
=====

```

Задача 14. Определить значение логического выражения $(A>B) \text{ AND } (B=A+2) \text{ OR NOT } (C<>B)$ при следующих исходных данных: $A=7, B=9, C=5$.

```

(*-----*)
I          ЛОГИКА          I
(*-----*)

PROGRAM LOG1(INPUT,OUTPUT);
VAR
  L : BOOLEAN;              (* ЛОГИЧЕСКАЯ ПЕРЕМЕННАЯ *)
  A,B,C : REAL;             (* ДЕЙСТВИТЕЛЬНЫЕ ПЕРЕМЕННЫЕ *)
BEGIN
  WRITE('ВВЕДИТЕ ЗНАЧЕНИЯ A,B,C: ');
  READ(A,B,C);
  L:=(A>5) AND (B=A+2) OR NOT(C<>B);
  WRITE('ЗНАЧЕНИЕ L =', L)
END.

=====
=
= ВВЕДИТЕ ЗНАЧЕНИЯ A,B,C: 7 9 5 =
= ЗНАЧЕНИЕ L = TRUE =
=====

```


Картина 7. ДАННЫЕ СИМВОЛЬНОГО ТИПА

Дока. Язык Паскаль позволяет обрабатывать не только числа, но и символы. Это дает возможность представлять в программах тексты и выполнять над ними некоторые операции, обрабатывать различные ведомости, документы, справочники и т.д.

Набор символов в ЭВМ достаточно большой. Все символы упорядочены, т.е. каждый символ имеет свой порядковый номер (см. Приложение 2). Символы с номерами 0—31 используются для управления обменом данными между ЭВМ и внешними устройствами. Символы с номерами 32—126 представлены на клавиатуре и могут быть изображены на экране. Последний символ с номером 127 — забой, он стирает предыдущий введенный символ. Оставшиеся номера отведены для кодов строчных букв и графических символов.

Константа. Символьная константа (литера) — это символ, заключенный в апострофы, например:

'A', 'R', '+', '7', '.,'.

Чтобы представить апостроф, его повторяют дважды и заключают в апострофы: '''''. Внешние апострофы (по одному слева и справа) не входят в константу, они являются признаком символьной константы. Символьная константа занимает один байт памяти. Ее можно обозначать именем и задавать в разделе констант, например:

```
CONST
  SIM = 'A' ;
  A = 'A' ;
  S = '+' ;
  D1 = '.,' ;
```

Переменная. Символьная переменная принимает значение одного символа. Она описывается в разделе переменных описателем CHAR (символьный тип), например:

```
VAR
  BUKWA : CHAR;
  B1, B2 : CHAR;
  R : CHAR;
```

Допускается использование символьных данных в операторе присваивания, например:

BUKWA:= '*'; R:= BUKWA; B1:= SIM;

После выполнения этих операторов символьные переменные примут следующие значения:

Переменная	B1	B2	R	BUKWA
Значение	A	+	*	A

Операции. Так как символы языка упорядочены, то к символьным данным применимы операции отношений (сравнений):

<, <=, =, >, >=, <>.

Поэтому 'A' < 'B'; '+' < '.'; '*' < '4'. Результатом операции сравнения является логическая константа TRUE (истина) или FALSE (ложь). Например,



результатом операции 'D' < 'R' является значение TRUE, а операции '4' = '3' значение FALSE.

К символьным данным можно применять встроенные функции:

ORD(X) — определяет порядковый номер символа X, например ORD('R') = 82;

CHR(X) — определяет символ, стоящий по порядковому номеру X, например CHR(68) = 'D';

PRED(X) — определяет предыдущий символ по отношению к X, например PRED('N') = 'M';

SUCC(X) — определяет последующий символ по отношению к X, например SUCC('R') = 'S'.

Необходимо помнить, что при использовании функции PRED(X) и SUCC(X) должны быть как последующий, так и предыдущий символ по отношению к X, иначе значение этих функций не определено. Аргументами этих функций могут быть и другие типы данных (см. Приложение 5), но только не действительные, так как для них не существует понятия предыдущего и последующего элемента.

Ввод и вывод. Ввод символьных данных имеет особенности. Поскольку пробел, как и любой символ языка Паскаль, относится к символьным данным, то символьные данные вводятся сплошной строкой в соответствии с оператором ввода. Напомним, что одной переменной можно присвоить значение только одного символа.

Пусть имеется фрагмент программы

```
VAR
  S1, S2, S3 : CHAR;
  .....
  READ( S1, S2, S3 )
```

Если после набора на экране дисплея всей программы и запуска ее на выполнение ввести данные в виде

ABC BK

где BK — признак конца строки, то переменные S1, S2, S3 получают следующие значения:

S1 = 'A', S2 = 'B', S3 = 'C'.

При вводе данных в виде

A _ B _ C BK

переменные получают следующие значения:

S1 = 'A', S2 = ' ', S3 = 'B'.

Если необходимо вводить одновременно и числовые и символьные данные, то рекомендуется ввод символьных данных выполнять с новой строки, иначе могут возникнуть ошибки.

Кроме того, в компьютере типа ДБК необходимо наличие одного пустого оператора READLN, если первый оператор вводит символьные данные.

Пусть, например, имеется следующее описание переменных:

```
VAR
  P1, P2 : INTEGER; (* ЦЕЛЫЙ ТИП *)
  S1, S2 : CHAR;    (* СИМВОЛЬНЫЙ ТИП *)
```


тогда допустимы следующие операторы ~~вывода~~:

I		I		I
I	READLN(P1,P2);	I	READLN; (* для АВК *)	I
I	READ(S1,S2);	I	READLN(S1,S2);	I
I	I	READ(P1,P2);	I
I	513 8-	I	I
I	+A	I	+A	I
I		I	513 8	I
I		I		I
I		I		I
I	READ(P1,P2);	I	READLN; (* для АВК *)	I
I	READLN;	I	READLN(S1);	I
I	READ(S1);	I	READLN(S2);	I
I	READ(S2);	I	READ(P1,P2);	I
I	I	I
I	513 8	I	+	I
I	+	I	A	I
I	A	I	513 8	I
I		I		I

Во всех этих случаях переменные получают следующие значения:

P1 = 513; P2 = 8; S1 = '+'; S2 = 'A'.

Вывод значений символьных данных можно организовать с помощью формата и без него. При бесформатном выводе отводится одна позиция для одного значения, например вывод значений символьных переменных B1 = 'W', B2 = '+', B3 = 'R' с помощью оператора WRITE (B1,B2,B3), имеет вид:

W+R

Поскольку пробел есть символьное данные, то его можно использовать в качестве константы. Например, оператор

WRITE (B1, '_', B2, '_', B3)

выведет на экран дисплея строку

W _ _ R

Вывод с форматами позволяет оформлять результаты в более наглядном виде. Формат для символьных данных имеет вид:

x:m

где x — символьное данные (константа, переменная), m — поле выводимого значения. В качестве m может быть константа, переменная или выражение целого типа. Если значение m больше единицы, то перед символом выводится соответствующее число пробелов, например при выводе рассмотренных значений B1, B2, B3 с помощью оператора

WRITE(B1:3, B2:4, B3:4)

получим

_ _ W _ _ _ _ * _ _ _ _ R

Для вывода нескольких пробелов символ пробела также можно указать с форматом, например '_' :7 для вывода семи пробелов. Пробел с форматом широко используется для пропуска пустых позиций перед выводимыми данными

или между ними. Пусть, например, A и B — переменные целого типа, тогда оператор

```
WRITE(' ':5, A:4, ' ':3, B:3)
```

выведет строку

```
-----A-----B
```

Символьные строки. В языке Паскаль имеется особый вид данных — символьная строка или просто строка. Она представляется в виде последовательности символов, заключенной в апострофы, например

```
'НОМЕР СТРАНИЦЫ', 'СУММА', '-----', 'A+B'
```

Для представления внутреннего апострофа ставится двойной внутренний апостроф, например

```
'КИНОТЕАТР "ОРБИТА"' или 'Д"АРТАНЬЯН'
```

В некоторых версиях языка Паскаль длина строки может быть ограничена и чаще всего числом 256.

Строки можно обозначать именем в разделе констант или использовать явно в разделе операторов, например:

```
CONST  
  C1 = '-----';  
  C2 = 'РЕЗУЛЬТАТ';  
  ...
```

При выполнении этого фрагмента на экране появится информация в виде

```
WRITELN(C1);  
WRITELN(C2);  
WRITELN(C1);  
  
-----  
РЕЗУЛЬТАТ  
-----
```

Точно такую же информацию можно получить, если строки указать явно в операторе вывода без использования раздела констант:

```
WRITELN('-----');  
WRITELN('РЕЗУЛЬТАТ');  
WRITELN('-----');
```

К строкам применимы операции отношения, при этом результатом сравнения являются логические константы TRUE (истина) или FALSE (ложь). Сравнение происходит посимвольно слева направо. Например, результатом операции 'ABC' >= 'ACB' будет FALSE, так как при сравнении вторых символов имеем 'B' < 'C'.

Следует заметить, что переменной нельзя присвоить значение строки. Не допустим, например, при описании VAR S : CHAR следующий оператор:

```
S:='РЕЗУЛЬТАТ'
```

так как переменной можно присвоить значение только одного символа. Для работы со строковыми переменными используется понятие массива (подробнее об этом будет рассказано в действии 7).

Задача 15. Сравнить две строковые константы "СТОЛ" и "СТУЛ". Какая из них меньше? Или они равны?

```
(*-----*)
I      СИМВОЛЬНЫЕ ДАННЫЕ      I
-----*)
PROGRAM L14(INPUT,OUTPUT);
  CONST
    C1 = 'СТОЛ' ;
    C2 = 'СТУЛ' ;
  VAR
    K1: INTEGER;  (* ПОРЯДКОВЫЙ НОМЕР "О" *)
    K2: INTEGER;  (* ПОРЯДКОВЫЙ НОМЕР "У" *)
    L : BOOLEAN;  (* TRUE ИЛИ FALSE      *)
  BEGIN
    K1:= ORD('О');
    K2:= ORD('У');
    L:= C1<C2;
    WRITELN('ПОРЯДКОВЫЙ НОМЕР "О" =', K1:4 );
    WRITELN('ПОРЯДКОВЫЙ НОМЕР "У" =', K2:4 );
    WRITE('СТОЛ МЕНЬШЕ СТУЛА? ', L )
  END.
```

```
=====
=
=      ПОРЯДКОВЫЙ НОМЕР "О" = 111      =
=      ПОРЯДКОВЫЙ НОМЕР "У" = 117      =
=      СТОЛ МЕНЬШЕ СТУЛА?   TRUE       =
=
=====
```

Задача 16. Использование встроенных функций ORD, PRED, SUCC для символьных данных.

Нужно ввести любой символ языка Паскаль и определить его порядковый номер, а также предыдущий и последующий символы

```
(*-----*)
I      СИМВОЛЬНЫЕ ДАННЫЕ      I
-----*)
PROGRAM SIMB1(INPUT,OUTPUT);
  VAR
    S1,S2,S3: CHAR;  (* СИМВОЛЬНЫЕ ПЕРЕМЕННЫЕ *)
    N: INTEGER;  (* ПЕРЕМЕННАЯ ЦЕЛОГО ТИПА *)
  BEGIN
    READLN; (* ++++++ *)
    WRITE('ВВЕДИТЕ ЗНАЧЕНИЕ СИМВОЛА S1: ');
    READ(S1);
    N:=ORD(S1);
    S2:=PRED(S1);
    S3:=SUCC(S1);
    WRITELN('ПОРЯДКОВЫЙ НОМЕР = ', N:3);
    WRITELN('ПРЕДЫДУЩИЙ СИМВОЛ = ', S2);
    WRITELN('ПОСЛЕДУЮЩИЙ СИМВОЛ = ', S3)
  END.
```

```
=====
=
=      ВВЕДИТЕ ЗНАЧЕНИЕ СИМВОЛА S1: *      =
=      ПОРЯДКОВЫЙ НОМЕР      - 42          =
=      ПРЕДЫДУЩИЙ СИМВОЛ    - )          =
=      ПОСЛЕДУЮЩИЙ СИМВОЛ    - +          =
=
=====
```



```

-----
ВВЕДИТЕ ЗНАЧЕНИЕ СИМВОЛА S1: D
ПОРЯДКОВЫЙ НОМЕР      = 68
ПРЕДЫДУЩИЙ СИМВОЛ    = C
ПОСЛЕДУЮЩИЙ СИМВОЛ   = E
-----

```

Результаты выполнения программы представлены дважды. В первом случае введен символ 'C', а во втором — символ 'D'.
Первый оператор программы READLN нужен только для машин типа ДБК.



Задача 17. Пусть задан символ '+', обозначенный именем A, и символ 'R'. Необходимо ввести два любых символа языка Паскаль (переменные SIM1 и SIM2) и определить результат выполнения операций: A>SIM1; SIM2 = 'R'.

```

(*-----*)
I      СИМВОЛЬНЫЕ ДАННЫЕ      I
(*-----*)
PROGRAM SIMB2(INPUT,OUTPUT);
CONST A='+'; (* СИМВОЛЬНАЯ КОНСТАНТА *)
VAR
  SIM1,SIM2: CHAR; (* СИМВОЛЬНЫЕ ПЕРЕМЕННЫЕ *)
  L1,L2: BOOLEAN;
BEGIN
  READLN; (* ++++++ *)
  WRITE('ВВЕДИТЕ ЗНАЧЕНИЯ СИМВОЛОВ SIM1 и SIM2 : ');
  READ(SIM1,SIM2);
  L1:= A > SIM1;
  L2:= SIM2 = 'R';
  WRITELN('L1 =', L1);
  WRITELN('L2 =', L2)
END.
=====
=
= ВВЕДИТЕ ЗНАЧЕНИЯ СИМВОЛОВ SIM1 и SIM2 : BR
= L1 = FALSE
= L2 = TRUE
=
=====

```

Поскольку в условии задачи сказано, что задано значение символьной константы A = '+', то в программе оно определяется в разделе CONST. При выводе данных переменная SIM1 получает значение 'B', а переменная SIM2 — значение 'R'.

Действие четвертое

ПОЙДЕТ НАПРАВО — ПЕСНЬ ЗАВОДИТ, НАЛЕВО — СКАЗКУ ГОВОРИТ

Картина 1. ЛЮБИШЬ КАТАТЬСЯ, ЛЮБИ И САНОЧКИ ВОЗИТЬ

Дока. Мы уже много позанимались, пора и отдохнуть. Пойдем, Фока, завтра на санках кататься?

Фока. Если будет хорошая погода, то пойдем кататься на санках, а иначе — пойдем в кино. Согласен?

Дока. Ого! Сама жизнь мне подсказывает, как лучше объяснить тебе новый оператор, который мы сейчас будем изучать.

Нам часто приходится что-либо делать лишь при выполнении определенных условий. Вот, например, мы завтра пойдем кататься на санках лишь при условии, что будет хорошая погода. Аналогичное положение мы встречаем и в математике, например,

если $x < 0$, то вычислить $y = x + 1$,

если $x \geq 0$, то вычислить $y = 2x$.

Это условие можно записать короче:

если $x < 0$, то вычислить $y = x + 1$,

иначе вычислить $y = 2x$.

Здесь в зависимости от условия выполняется одно из двух действий: либо вычисляется $y = x + 1$, либо $y = 2x$.

Для программирования таких ситуаций в языке Паскаль предусмотрен условный оператор, который имеет две формы — полную и краткую.

Полная форма условного оператора имеет вид

IF *логическое выражение* THEN *оператор 1*
ELSE *оператор 2*.

Здесь оператор 1 и оператор 2 — простые или составные операторы, а IF (если), THEN (тогда), ELSE (иначе) — служебные слова.

Если логическое выражение истинно, тогда выполняется оператор 1, иначе (если логическое выражение ложно) выполняется оператор 2. В качестве операторов 1 и 2 могут быть также условные операторы.

Для рассмотренного примера условный оператор выглядит следующим образом:

```
IF X < 0 THEN Y := X + 1  
ELSE Y := 2 * X
```



Здесь значение X должно быть определено до выполнения оператора IF.

Условие, управляющее разветвлением вычислений, не обязательно должно иметь форму операции отношения. Оно может принимать вид любого логического выражения и, в частности, логической переменной.

Если в нашем примере описать логическую переменную B типа BOOLEAN и определить ее значение отдельно

$B := X < 0$

то условный оператор примет вид

IF B THEN Y := X + 1

ELSE Y := 2 * X

Рассмотрим другой пример. Вычислить $A = B + C$, если K больше 20, но меньше 50. При всех других значениях K вычислить $W = R - T$. Условный оператор имеет вид:

IF (K > 20) AND (K < 50) THEN A := B + C
ELSE W := R - T



Правила составления программы позволяют записывать ее в свободной форме. Однако для удобства восприятия программы, особенно большой и сильно разветвленной, рекомендуется слово ELSE писать под тем словом IF, к которому оно относится, например:

```
IF A=B THEN
    IF C<D THEN X:=1
    ELSE X:=2
ELSE X:=3
```


Этот пример соответствует следующей алгебраической схеме:

$$x = \begin{cases} 1, & \text{если } A=B \text{ и } C>D \\ 2, & \text{если } A=B \text{ и } C\geq D \\ 3, & \text{если } A\neq B. \end{cases}$$

З а м е ч а н и е. Некоторые программисты используют другую запись условного оператора:

```
IF логическое выражение
  THEN оператор 1
  ELSE оператор 2
```

На первый взгляд она более целесообразна, но, как показал опыт, предыдущая форма записи более наглядна в сложной логической структуре, а вторая — в простой. Мы будем пользоваться и той, и другой формой записи условного оператора в зависимости от решаемой задачи.

Действие условного оператора может быть расширено использованием составного оператора. В этом случае после слов THEN и ELSE могут быть составные операторы:

```
IF логическое выражение THEN
  BEGIN
    оператор 1;
    оператор 2;
    ...
    оператор n-1;
    оператор n
  END
ELSE
  BEGIN
    оператор 1;
    оператор 2;
    ...
    оператор n-1;
    оператор n
  END
```

Следует обратить внимание на то, что перед служебным словом ELSE не стоит знак точки с запятой.

Внутри составных операторов могут быть также условные операторы, содержащие простые или составные операторы. Глубина вложенности может быть ограничена в различных версиях языка.

П р и м е р. Если $A>B$, то нужно вычислить три оператора: $Y1 = 7$, $Y2 = A$, $Y3 = A+B$. Если $A\leq B$, то нужно вычислить $T1 = 2\cdot A$ и $T2 = A-B$. Условный оператор имеет вид

```
IF A>B THEN
  BEGIN
    Y1:=7;
    Y2:=A;
    Y3:=A+B
  END
ELSE
  BEGIN
    T1:=2*A;
    T2:=A-B
  END
```


В данном примере использовались два составных оператора. Возможны случаи, когда используются один составной оператор, а другой простой. Начинаящие программисты часто допускают такую ошибку: после слов THEN или ELSE имеют в виду составной оператор, а операторные скобки BEGIN-END ставить забывают. В этом случае выдается сообщение о неправильной конструкции условного оператора.

Мы рассмотрели полную форму условного оператора. В языке допускается и краткая форма, которая имеет вид

IF логическое выражение THEN оператор 1

Если логическое выражение истинно, то выполняется оператор 1, иначе (если логическое выражение ложно) выполняется оператор, расположенный в программе после этого условного оператора IF. П р и м е р

```
IF A > 15 THEN Y:=7;
Z:=SUM+1;
```

Если условие $A > 15$ истинно, то вычисляется значение Y и происходит передача управления на вычисление значения Z. Если условие $A > 15$ ложно (т.е. $A \leq 15$), то сразу выполняется оператор присваивания $Z := SUM + 1$.

Сравним два примера:

```
1) IF X=5 THEN X:=-7;
    IF X=7 THEN X:=-5;
2) IF X=5 THEN X:=-7
    ELSE IF X=7 THEN X:=-5
```

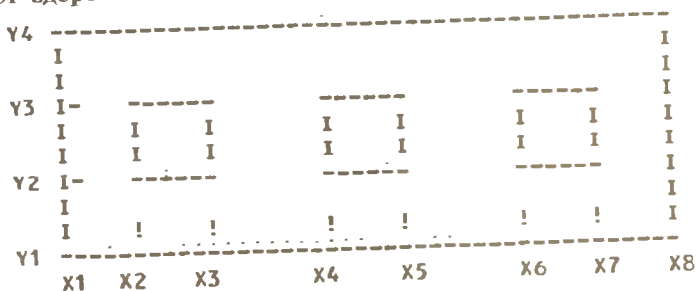
В первом примере представлены два условных оператора, записанные в краткой форме. Во втором — один условный оператор, записанный в полной форме. Чему будет равно значение X в результате выполнения операторов, если начальное значение X равно сначала 5, затем 7?

Дока. Пора нам с тобой, Фока, покрасить свой дом, а то краска начала облезать. Давай сначала покрасим лицевую сторону, а уж потом другие стороны.

Фока. Это неплохая идея!

Дока. Действовать будем так: составим алгоритм покраски, я буду называть координаты точки стены, а ты в соответствии с алгоритмом будешь красить эту точку в зеленый, розовый или голубой цвет.

Фока. Вот здорово! Начинаем...




```

WRITELN('ПОКРАСЬ, ФОКА, ТОЧКУ СТЕНЫ С КООРДИНАТАМИ:');
READ(X,Y);
IF ((Y>Y1) AND (Y<Y2) AND (X>X1) AND(X<X8)) OR
  ((Y>Y2)AND(Y<Y3) AND
  ( ((X>X1)AND(X<X2)) OR ((X>X7)AND(X<X8)) ) THEN
  WRITELN('ТОЧКА ПОКРАШЕНА В КРАСНЫЙ ЦВЕТ')
ELSE
  IF (Y>Y3)AND(Y<Y4) AND
    ( ((X>X1)AND(X<X3)) OR ((X>X6)AND(X<X8)) ) THEN
    WRITELN('ТОЧКА ПОКРАШЕНА В ГОЛУБОЙ ЦВЕТ')
  ELSE WRITELN('ТОЧКА ПОКРАШЕНА В ЗЕЛЕНый ЦВЕТ');

```

Фока. Ой, что такое получилось? Все окна оказались покрашенными в зеленый цвет. Похоже, в алгоритме содержится ошибка!

Дока. Странно... Но давай разберёмся. Красный и голубой цвета мы нанесли верно. А вот когда решили оставшуюся незакрашенную часть стены покрасить зеленым цветом, то надо было тоже дать ограничения. Вот где наша ошибка!

Какие ограничения нужно задать в алгоритме и как будет выглядеть покрашенная стена, пусть решают прохожие, а мы пока очистим окна от краски.

Задача 18. Однажды встретились три толстяка и начали спорить, кто же из них самый толстый. После долгого спора решили обратиться к мудрецу. Мудрец думал, думал, да и сказал: "Тот из вас самый толстый, у кого больше масса". Так кто же из них?

```

(*-----
I          ТРИ ТОЛСТЯКА          I
-----*)
PROGRAM TOLS (INPUT,OUTPUT);
VAR
  A : REAL;  (* ПЕРВЫЙ ТОЛСТЯК *)
  B : REAL;  (* ВТОРОЙ ТОЛСТЯК *)
  C : REAL;  (* ТРЕТИЙ ТОЛСТЯК *)
BEGIN
  WRITELN(' УВАЖАЕМЫЕ ТОЛСТЯКИ ! ');
  WRITELN(' ПРИГЛАШАЮ НА ВЕСЫ ! ');
  WRITELN(' ЧТО ОНИ ПОКАЗЫВАЮТ ? ');
  READ(A,B,C);
  IF ( A>B ) AND ( A>C )
    THEN WRITELN('САМЫЙ ТОЛСТЫЙ ТОЛСТЯК - "А" ');
    ELSE IF ( B>C )
      THEN WRITELN('САМЫЙ ТОЛСТЫЙ ТОЛСТЯК - "В" ');
      ELSE WRITELN('САМЫЙ ТОЛСТЫЙ ТОЛСТЯК - "С" ');
  WRITELN('*****');
  WRITELN('ЕСЛИ ВЫ НЕ САМЫЙ ТОЛСТЫЙ ТОЛСТЯК');
  WRITELN('      НЕ ОТЧАИВАЙТЕСЬ,      ');
  WRITELN('АППЕТИТ ПРИХОДИТ ВО ВРЕМЯ ЕДЫ! ');
END.

```

```

=====
=
=  УВАЖАЕМЫЕ ТОЛСТЯКИ !
=  ПРИГЛАШАЮ НА ВЕСЫ !
=  ЧТО ОНИ ПОКАЗЫВАЮТ ?
=  187   205   201
=  САМЫЙ ТОЛСТЫЙ ТОЛСТЯК - "В"
=  *****
=  ЕСЛИ ВЫ НЕ САМЫЙ ТОЛСТЫЙ ТОЛСТЯК
=      НЕ ОТЧАИВАЙТЕСЬ,
=  АППЕТИТ ПРИХОДИТ ВО ВРЕМЯ ЕДЫ!
=
=====

```




Задача 19. Вычислить значение функции $y=1/x$ при заданном значении аргумента x . Поскольку машина не может выполнить операцию деления на ноль, то в программе необходимо предусмотреть проверку аргумента x . Если он равен нулю, то вычисления производить не нужно.

```

(*)-----
I  ВЫЧИСЛЕНИЕ ФУНКЦИИ Y=1/X  I
-----(*)-----
PROGRAM PT2(INPUT,OUTPUT);
  VAR X,Y: REAL;
BEGIN
  WRITELN('ВВЕДИТЕ ЗНАЧЕНИЕ X: ');
  READ(X);
  IF X=0 THEN WRITE('НЕ МОГУ ВЫЧИСЛИТЬ')
  ELSE BEGIN
    Y:=1/X;
    WRITE('Y=', Y)
  END
END.

```

```

-----
=      ВВЕДИТЕ ЗНАЧЕНИЕ X:      =
=      6                        =
=      Y= 1.666667E-01          =
-----
=      ВВЕДИТЕ ЗНАЧЕНИЕ X:      =
=      0                        =
=      НЕ МОГУ ВЫЧИСЛИТЬ!      =
-----

```

По окончании программы представлено два результата ее выполнения: при $x=6$ и $x=0$. Так как в операторе вывода не указан формат, значение функции выводится с плавающей точкой, т.е. в виде мантиссы и порядка. Последняя цифра в мантиссе округляется.

Задача 20. Составить программу, в результате вычисления которой логической переменной A будет присвоено значение TRUE, если точка с координатами X,Y принадлежит прямоугольнику, и значение FALSE, если не принадлежит.

```

(*-----*)
!   ПРИНАДЛЕЖНОСТЬ ТОЧКИ ПРЯМОУГОЛЬНИКУ !
(*-----*)
PROGRAM A40A(INPUT,OUTPUT);
VAR
  X1,X2,Y1,Y2 : REAL ; (* КООРДИНАТЫ ПРЯМОУГОЛЬНИКА *)
  X, Y : REAL ; (* КООРДИНАТЫ ТОЧКИ *)
  A : BOOLEAN; (* ЛОГИЧЕСКАЯ ПЕРЕМЕННАЯ *)
BEGIN
  WRITELN('      ПРЯМОУГОЛЬНИК С КООРДИНАТАМИ X,Y      ');
  WRITELN('      ');
  WRITELN('      Y2 ! ----- ');
  WRITELN('      ! !             ');
  WRITELN('      ! !             ');
  WRITELN('      Y1 ! ----- ');
  WRITELN('      ! !             ');
  WRITELN('      ! !             ');
  WRITELN('      ! ----- ');
  WRITELN('      X1             X2      ');
  WRITELN('      ');
  WRITELN(' ВВЕДИТЕ КООРДИНАТЫ ПРЯМОУГОЛЬНИКА X1,X2,Y1,Y2 : ');
  READLN( X1,X2,Y1,Y2 );
  WRITELN(' ВВЕДИТЕ КООРДИНАТЫ ТОЧКИ : ');
  READLN( X,Y );
  IF ( X>=X1 )AND( X<=X2 )AND( Y>=Y1 )AND( Y<=Y2 )
    THEN A:= TRUE
    ELSE A:= FALSE;
  WRITE('ЗНАЧЕНИЕ A =', A )
END.

```

```

-----
=
=      ПРЯМОУГОЛЬНИК С КООРДИНАТАМИ X,Y
=
=      !
=      Y2 ! -----
=      ! !             !
=      ! !             !
=      Y1 ! -----
=      ! !             !
=      ! !             !
=      ! -----
=      X1             X2
=
=
=      ВВЕДИТЕ КООРДИНАТЫ ПРЯМОУГОЛЬНИКА X1,X2,Y1,Y2 :
=      3 19 5 9
=      ВВЕДИТЕ КООРДИНАТЫ ТОЧКИ :
=      5.5 7.1
=      ЗНАЧЕНИЕ A = TRUE
=
=
-----

```

Картина 2. ПРАВО ВЫБОРА

Фока. Мне кажется, что в условном операторе очень строгое ограничение: либо условие истинно, либо прямо противоположно - ложно. Но ведь нам часто приходится иметь дело с более многообразными ситуациями. Например, из многих возможных вариантов выбрать что-то одно или выполнить ряд действий в зависимости от многих условий. Как быть в этом случае?

Дока. Ну, во-первых, во **всех** подобных случаях можно использовать условным оператором. Правда, он будет выглядеть довольно сложно: одни условия вложены в другие. Во-вторых, в языке Паскаль имеется специальный оператор - оператор выбора.

Оператор выбора (варианта) используется в тех случаях, когда в зависимости от значения какого-либо выражения необходимо выполнить один из нескольких последовательных операторов. Оператор выбора имеет следующую форму записи:

```
CASE _ выражение _ OF:
    константа 1: оператор 1;
    константа 2: оператор 2;
    ...
    константа n-1: оператор n-1;
    константа n : оператор n
END
```

Здесь CASE (в случае), OF (из), END (конец) — служебные слова. В качестве оператора может быть и составной оператор.

Оператор выбора действует следующим образом. В случае, если значение выражения равно одной из констант, то выполняется соответствующий ей оператор. Затем управление передается за пределы оператора выбора. Если значение выражения не совпадает ни с одной константой, то управление передается на оператор END и выполнение программы продолжается.

Значение выражения должно принадлежать простому типу (кроме действительного). В соответствии с этим и константа не может быть действительного типа. Тип константы должен совпадать с типом выражения.

Пример записи оператора выбора

```
CASE K+1 OF
    5 : Y:= SQR(X) ;
    11 : Y:= SQRT(X) ;
    4 : Z:= 4*(A-B) ;
    7 : WRITE(A,B)
END
```

Если значение K+1 будет равно 5, то выполнится оператор присваивания $Y:=SQR(X)$ и управление будет передано на оператор, расположенный после слова END. Аналогично этому, если значение K+1 будет равно 11, 7 или 7, то выполнится один соответствующий оператор и управление будет передано за пределы оператора выбора.

CASE — ОПЕРАТОР ВЫБОРА



CASE — ДЕНЬ НЕДЕЛИ

OF ПН. СР. СБ. } СПОРТ

ВТ. ПТ. } ПРОГРАММИРОВАНИЕ

ЖИВОПИСЬ { ЧТ. ХР-Р-ПЕС-С-С

ВС. } НИЧЕГО НЕ ДЕЛАНИЕ

END

Переменная К должна быть объявлена как переменная целого типа. Кроме того, переменные К,Х,А,В должны получить значения до выполнения оператора CASE.

В операторе выбора в качестве константы допускается использование списка констант, например:

```

CASE S OF
  '+', '-', '*', '/' : P:=1 ;
  'A', 'B' : P:=2 ;
  '.' : P:=3 ;
END

```

Переменная S должна быть объявлена в разделе описаний как символьная. Если значением S будет один из знаков '+', '-', '*', '/', то переменная P получит значение 1. Если значением S будет символ 'A' или 'B', то P получит значение 2. Если значением S будет знак '.', то переменной P будет присвоено значение 3.

Задача 21. Ввести номер дня недели и вывести соответствующий ему день недели на русском и английском языках. Ниже представлена программа и ответ для введенного номера недели 5.

```

(*-----*
!                ДЕНЬ НЕДЕЛИ                !
*-----*)
PROGRAM E32(INPUT,OUTPUT);
VAR N : INTEGER;      (* НОМЕР ДНЯ НЕДЕЛИ *)
BEGIN
  WRITELN('ВВЕДИТЕ НОМЕР ДНЯ НЕДЕЛИ : ');
  READ( N );
  CASE N OF
    1 : WRITELN ('ПОНЕДЕЛЬНИК - MONDAY' );
    2 : WRITELN ('ВТОРНИК - TUESDAY' );
    3 : WRITELN ('СРЕДА - WEDNESDAY' );
    4 : WRITELN ('ЧЕТВЕРГ - THURSDAY' );
    5 : WRITELN ('ПЯТНИЦА - FRIDAY' );
    6 : WRITELN ('СУББОТА - SATURDAY' );
    7 : WRITELN ('ВОСКРЕСЕНЬЕ - SUNDAY' );
  END;
END.

=====
=
=      ВВЕДИТЕ НОМЕР ДНЯ НЕДЕЛИ :      =
=      5                                =
=      ПЯТНИЦА - FRIDAY                  =
=====

```

Картина 3. КУДА ПОЙТИ?

Дока. Мы не всегда выполняем действия последовательно, не так ли, Фока? Например, мне нужно купить продукты, книги и зайти в аптеку, а аптека оказалась закрытой на перерыв. Мне приходится сначала обойти аптеку и вернуться к ней позже.

В практике программирования задач также возникает необходимость обхода при выполнении программы. Для этой цели предназначен оператор перехода, который имеет следующую форму записи:

GOTO _ метка



Метка представляет собой любое целое число без знака в диапазоне от 1 до 9999. Это число записывают перед оператором, к которому нужно перейти, и отделяют от него двоеточием:

метка: оператор

Например,

```
GOTO 32 ;
10 : A := 2 ;
      ...
32 : Y := X / Z
```

В этом случае после оператора GOTO _ 32 выполняется оператор с меткой 32. Следует отметить, что оператор, следующий за оператором перехода, также должен быть помечен. Иначе все операторы в программе между оператором GOTO и операторы с меткой 32 будут лишними, так как они никогда не будут выполняться.

Метка должна быть объявлена в разделе описания меток. Объявление метки имеет вид

LABEL _ метка;

Допускается объявлять список меток:

LABEL _ метка 1, метка 2, . . . , метка n;

Для рассмотренного примера объявление меток выглядит следующим образом:

LABEL _ 10, 32;

В простых программах оператор перехода не вызывает затруднений, но они возникают, когда программа сложная и записана на многих листках. Тогда скачки при выполнении операторов программы проследить бывает трудно. При этом ясность и понятность программы могут сильно страдать. Поэтому программист стремится избежать применения операторов перехода. В ряде случаев это удастся сделать простыми способами. Например, отрезок программы, типичный для многих языков программирования




```

        IF A > B THEN GOTO 1 ;
        A:= A - B ;
        GOTO 2 ;
1:      A:= A + B ;
2:      Y:= A

```

вполне можно заменить следующей конструкцией языка Паскаль:

```

IF A > B THEN A:= A + B
ELSE A:= A - B ;
Y:= A

```

Тем не менее встречаются случаи, когда оператор безусловного перехода оказывается весьма полезным. Например, пусть необходимо прекращать выполнение программы, если встречаются так называемые ситуации прерывания. Например, если встречаются выражения, содержащие функции логарифма LN или корня квадратного SQRT от отрицательных аргументов, или необходимо выполнить деление на выражение, которое может обращаться в нуль:

```

IF Y<=0 THEN GOTO 100
ELSE A:=X*LN(Y);

IF Z<0 THEN GOTO 100
ELSE B:=X+SQRT(Z);

IF S=0 THEN GOTO 100
ELSE R:=T/S;

```

```
100: WRITE('ВЫПОЛНЕНИЕ ПРОГРАММЫ ЗАКАНЧИВАЕТСЯ')
```

В этом примере всякий раз, когда встречается недопустимая ситуация, происходит переход к оператору с меткой 100. На экран дисплея выводится сообщение о прерывании программы и ее выполнение заканчивается.

```

-----
I                                     I
I      (*      СЦЕНА ИЗ СЕМЕЙНОЙ ЖИЗНИ      *)      I
I      (*      -----      *)      I
I                                     I
I      WRITELN('СЫНОК, СХОДИ, ПОЖАЛУЙСТА, ЗА МОЛОКОМ ', I
I      'В МАГАЗИН N 47 ');      I
I      GOTO 47;      I
I      2: WRITELN('СЫНОК, СХОДИ, ПОЖАЛУЙСТА, ЗА ЖУРНАЛАМИ ', I
I      'НА ПОЧТУ N 58 ');      I
I      GOTO 58;      I
I      3: WRITELN('СЫН, ГДЕ ТЫ ТАК ДОЛГО ГУЛЯЕШЬ? ', I
I      'ПОЧЕМУ НЕ УЧИШЬ УРОКИ? ', I
I      'НЕМЕДЛЕННО ОТПРАВЛЯЙСЯ В ШКОЛУ! ');      I
I      GOTO 100;      I
I      ...      I
I      47: WRITELN('КУПИЛ МОЛОКО ');      I
I      GOTO 2;      I
I      58: WRITELN('СХОДИЛ НА ПОЧТУ ');      I
I      GOTO 3;      I
I      ...      I
I      100: WRITELN('ИДУ В ШКОЛУ ...');      I
I                                     I
-----

```


БЕЛКА В КОЛЕСЕ

Картина 1. ЦИКЛ, ЦИКЛ, ЦИКЛ...



Дока. Решение многих задач содержит повторяющиеся действия. Например, нужно вычислить значение функции при десяти различных значениях аргумента. Чтобы не записывать десять раз функцию, в языке Паскаль предусмотрены операторы повторения, которые называются операторами цикла. Применение циклов в программе позволяет эффективно использовать машину, приводит к сокращению длины программы и времени на ее составление и отладку.

Фока. Как хорошо, что имеются операторы цикла, а то замучаешься записывать одни и те же операторы по сто раз. Верно?

Дока. Конечно. Более того, для удобства программирования в языке Паскаль предусмотрены три вида операторов цикла:

WHILE

— ОПЕРАТОР
ЦИКЛА С
ПРЕДВАРИТЕЛЬНЫМ
УСЛОВИЕМ

REPEAT

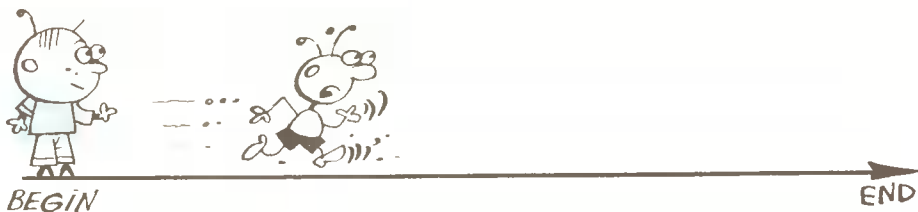
— ОПЕРАТОР
ЦИКЛА С
ПОСЛЕДУЮЩИМ
УСЛОВИЕМ

Для всех трех видов операторов цикла характерна следующая особенность. Повторяющиеся действия записываются всего лишь один раз. Вход в цикл возможен только через его начало. Необходимо предусмотреть выход из цикла: или по естественному его окончанию, или по оператору перехода. Если этого не предусмотреть, то циклические действия будут повторяться бесконечно. В этом случае говорят, что "программа зациклилась".

Картина 2. ПРОВЕРЬ — ПОТОМ ДЕЛАЙ

Дока. Представь себе, Фока, что ты спринтер и во время тренировки многократно пробегаешь стометровку. Тренер стоит на старте и





перед каждым забегом спрашивает тебя: "Есть еще силы бежать?" Если ты говоришь "Да", то снова бежишь, ну, а если "Нет", то бег заканчивается. Это типичная ситуация, которую отражает оператор цикла с предварительным условием. Он имеет следующую форму записи:

WHILE _ логическое выражение _ **DO**
BEGIN
 операторы циклической
 части программы
END



Слова **WHILE** (пока) и **DO** (выполнить) являются служебными словами. Оператор цикла действует следующим образом. Каждый раз предварительно проверяется значение логического выражения. Пока оно истинно, выполняются операторы циклической части. Как только значение логического выражения становится ложным, происходит выход за пределы цикла. Если с самого начала значение логического выражения ложно, то операторы циклической части не выполняются ни разу. Операторы циклической части, заключенные в операторные скобки **BEGIN** – **END**, представляют собой составной оператор.

Возможен случай, когда в циклической части стоит оператор перехода, передающий управление за пределы цикла. В такой ситуации цикл может завершиться до его естественного окончания (при истинном значении логического выражения).

Если в циклической части стоит всего один оператор, то операторные скобки **BEGIN**-**END** можно не указывать и оператор цикла принимает вид

WHILE _ логическое выражение _ **DO** _ оператор

П р и м е р. Вычислить $y=x^2$ при $x = 2, 4, 6, 8, 10$. Фрагмент циклической части программы имеет вид:

```
x:=2;
while x<=10 do
begin
  y:=x*x;
  writeln(x:3,y:5);
  x:=x+2
end;
```

Во время выполнения программы переменные принимают следующие значения:

X	Логическое выражение	Значение	Y
2	$2 \leq 10$	Истина	4
4	$4 \leq 10$	Истина	16
6	$6 \leq 10$	Истина	36
8	$8 \leq 10$	Истина	64
10	$10 \leq 10$	Истина	100
12	$12 \leq 10$	Ложь	Выход из цикла

Рассмотрим следующий фрагмент программы с использованием оператора цикла с предусловием

```

A:=1;
N:=1;
WHILE 2*A <= 3*N+1 DO
  BEGIN
    A:=A+2;
    N:=N+1
  END

```

Пока условие $2A \leq 3N+1$ является истинным (т.е. $2 \cdot A$ оказывается меньше или равно $3N+1$), выполняются операторы циклической части: значение A увеличивается на 2 и результат вновь присваивается переменной A ; значение N увеличивается на 1 и результат обозначается переменной N . Начальные значения переменных заданы до начала оператора цикла.

Переменные A и N , а также логическое выражение принимают следующие значения в процессе выполнения этой части программы:

A	N	$2A \leq 3N+1$	Истинно или ложно условие
1	1	$2 \leq 4$	Истинно
3	2	$6 \leq 7$	Истинно
5	3	$10 \leq 10$	Истинно
7	4	$14 \leq 13$	Ложно

Следует обратить внимание на то, что циклическая часть программы выполнялась 3 раза.

В операторе цикла можно применять стандартную функцию EOLN, которая связана с вводом данных: она принимает значение TRUE, если достигнут конец строки, в противном случае — FALSE.

Например:

```

VAR A,I:INTEGER;
...
READLN;
I:=0;
WHILE NOT EOLN DO
  BEGIN
    READ(A);
    I:=I+1;
    ...
  END

```

В данном фрагменте в циклической части вводится значение переменной A и подсчитывается количество введенных чисел. Циклическая часть выполняется до тех пор, пока в данных не встретится символ конца строки "BK", например

5 18 6 7 13 47 9 4 BK

В этом случае оператор цикла WHILE читается так: пока нет конца строки, выполнять циклическую часть. Для машин типа ДБК перед использованием оператора WHILE NOT EOLN DO, необходимо указывать оператор READLN без параметров. Это отражено в приведенном фрагменте программы.

Бесконечный цикл:

```
PROGRAM L75(INPUT,OUTPUT);
  VAR X: REAL;
BEGIN
  X:=0;
  WHILE X<>2 DO
  BEGIN
    WRITELN(X:8:5);
    X:=X+0.1
  END
END.
```

**БЕСКОНЕЧНЫЙ
ЦИКЛ:**



Правильно:

```
WHILE X<=2 DO
```

При использовании оператора цикла WHILE могут быть случаи, когда цикл будет выполняться бесконечно. Например, в силу приближенного представления действительного числа X может никогда не выполниться точное равенство в операторе

```
WHILE X=2 DO
```

Задача 22. Вычислить объем каждого из нескольких шаров, а затем найти суммарный объем всех шаров. Известно, что радиус первого шара R, а радиус каждого последующего шара больше предыдущего на величину DR. Радиус последнего шара равен RK.

В программе объем каждого шара обозначен через V, а суммарный объем — через VM.

Программа имеет вид:

```
(*-----*
!           ВЫЧИСЛЕНИЕ ОБЪЕМА СИСТЕМЫ ШАРОВ           !
*-----*)
PROGRAM E10(INPUT,OUTPUT);
CONST PI=3.14;
VAR
  V,VM: REAL;      (* ОБЪЕМ *)
  R,RK: REAL;      (* РАДИУС *)
  DR: REAL;        (* ИЗМЕНЕНИЕ РАДИУСА *)
BEGIN
  WRITELN('ВВЕДИТЕ ЗНАЧЕНИЯ R,RK,DR:');
  READ(R,RK,DR);
  VM:= 0;
  WHILE R <= RK DO
  BEGIN
    V:=4*PI*R*R*R/3;
    VM:= VM + V;
    WRITELN(' R=',R:6:2, ' ':5, ' V=',V:7:3);
    R:=R+DR
  END;
  WRITELN('*****');
  WRITELN('ОБЩИЙ ОБЪЕМ СИСТЕМЫ: VM=',VM:8:3)
END.
```



```

=====
=
=      ВВЕДИТЕ ЗНАЧЕНИЯ РАДИУСА R, RK, DR:      =
=      0.2  1.8  0.4      V=  0.033      =
=      R=  0.20      V=  0.904      =
=      R=  0.60      V=  4.187      =
=      R=  1.00      V=  11.488      =
=      R=  1.40      V=  24.417      =
=      R=  1.80      V=  41.029      =
=      *****      =
=      ОБЩИЙ ОБЪЕМ СИСТЕМЫ:  VM=  41.029      =
=
=====

```

Задача 23. Дан произвольный текст. Признаком конца текста считать нажатие клавиши ВК. Подсчитать общее количество введенных символов текста и число букв "N" в тексте. Поскольку заранее неизвестно, сколько раз будет выполняться цикл, для его организации используем оператор цикла WHILE. Условием окончания цикла является проверка конца строки. Пока не обнаружен конец строки (NOT EOLN), цикл продолжает выполняться.

Программа имеет вид:

```

(*-----*)
!      АНГЛИЙСКАЯ ПОГОВОРКА      !
(*-----*)
PROGRAM ЕЗА(INPUT,OUTPUT);
VAR
  BUK:CHAR;      (* БУКВА ТЕКСТА *)
  N:INTEGER;      (* СЧЕТЧИК БУКВЫ "N" *)
  K:INTEGER;      (* СЧЕТЧИК ВСЕХ БУКВ *)
BEGIN
  K:=0;  N:=0;
  READLN;
  WRITELN('ВВЕДИТЕ ТЕКСТ:');
  WHILE NOT EOLN DO
    BEGIN
      READ(BUK);
      K:= K + 1;
      IF BUK = 'N' THEN  N:= N+1
    END;
  WRITELN;
  WRITELN('КОЛИЧЕСТВО СИМВОЛОВ В ТЕКСТЕ =',K:3);
  WRITELN('ЧИСЛО БУКВЫ "N" В ТЕКСТЕ =',N:3)
END.
=====
=
=      ВВЕДИТЕ ТЕКСТ:      =
=      NO SOONER SAID, THAN DONE      =
=
=      КОЛИЧЕСТВО СИМВОЛОВ В ТЕКСТЕ = 25      =
=      ЧИСЛО БУКВЫ "N" В ТЕКСТЕ = 4      =
=
=====

```



Задача 24. Одному крестьянину понравилось ходить в сауну соседнего города. Но жена его была недовольна и разрешила ему посещать сауну только по четным субботам.

Требуется определить, сколько раз крестьянин был в сауне в течение двух месяцев, если дни суббот были следующие:

1, 8, 15, 22, 29, 5, 12, 19, 26.


```

(*-----*)
!   САУНА - ТОЛЬКО ПО ЧЕТНЫМ СУББОТАМ !
(*-----*)
PROGRAM E34C(INPUT,OUTPUT);
  VAR
    KC:INTEGER; (* СЧЕТЧИК ЧЕТНЫХ СУББОТ *)
    KN:INTEGER; (* СЧЕТЧИК НЕЧЕТНЫХ СУББОТ*)
    X:INTEGER; (* ВВОДИМОЕ ЧИСЛО *)
  BEGIN
    KC:=0; KN:=0; (* НАЧАЛЬНЫЕ ЗНАЧЕНИЯ *)
    READLN;
    WRITELN('УКАЖИТЕ ЧИСЛА СУББОТ:');
    WHILE NOT EOLN DO
      BEGIN
        READ(X);
        IF ODD(X) THEN KN:=KN+1
        ELSE KC:=KC+1
      END;
    WRITELN;
    WRITELN('РЕЗУЛЬТАТ :');
    WRITELN('КОЛИЧЕСТВО ЧЕТНЫХ СУББОТ =' ,KC:3);
    WRITELN('КОЛИЧЕСТВО НЕЧЕТНЫХ СУББОТ =' ,KN:3)
  END.

```



```

=====
.                                     =
=      УКАЖИТЕ ЧИСЛА СУББОТ:      =
=      1 8 15 22 29 5 12 19 26    =
=                                     =
=      РЕЗУЛЬТАТ :                =
=      КОЛИЧЕСТВО ЧЕТНЫХ СУББОТ =  4  =
=      КОЛИЧЕСТВО НЕЧЕТНЫХ СУББОТ =  5  =
=                                     =
=====

```

Для определения четности числа используется встроенная функция ODD. Если число не делится на 2, то результат этой функции истинный, а если делится, то ложный.

Цикл организован с помощью оператора WHILE, в котором введена проверка на конец строки (встроенная функция EOLN). Цикл выполняется до тех пор, пока в вводимой строке чисел не встретится символ конца строки "BK".

Задача 25. Гадание на ромашке: любит — не любит.

```

(*-----*)
I      ГАДАНИЕ НА РОМАШКЕ      I
(*-----*)
PROGRAM ROM(INPUT,OUTPUT);
  VAR
    ЛЕПЕСТОК:CHAR; (* ЛЕПЕСТОК РОМАШКИ *)
    I:INTEGER;      (* ЧИСЛО ЛЕПЕСТКОВ *)
  BEGIN
    ЛЕПЕСТОК:='Y'; I:=1; (* НАЛИЧИЕ ХОТЯ БЫ ОДНОГО
                          ЛЕПЕСТКА У РОМАШКИ *)
    READLN; (* ++++++ *)
    WHILE ЛЕПЕСТОК='Y' DO
      BEGIN
        IF ODD(I) THEN WRITELN('***** ЛЮБИТ *****')
        ELSE WRITELN('***** НЕ ЛЮБИТ *****');
        WRITE('ЕСТЬ ЛЕПЕСТОК У РОМАШКИ ? ');
        READLN(ЛЕПЕСТОК);
        I:=I+1
      END
    END
  END
END

```



```

=====
***** ЛЮБИТ *****
ЕСТЬ ЛЕПЕСТОК У РОМАШКИ ? Y
***** НЕ ЛЮБИТ *****
ЕСТЬ ЛЕПЕСТОК У РОМАШКИ ? Y
***** ЛЮБИТ *****
ЕСТЬ ЛЕПЕСТОК У РОМАШКИ ? Y
***** НЕ ЛЮБИТ *****
ЕСТЬ ЛЕПЕСТОК У РОМАШКИ ? Y
***** ЛЮБИТ *****
ЕСТЬ ЛЕПЕСТОК У РОМАШКИ ? N
=====

```

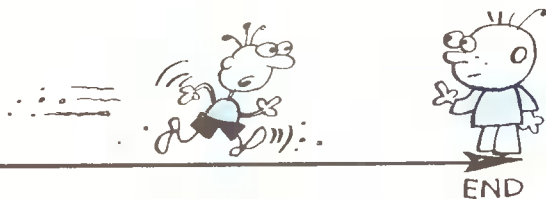


В операторе цикла сначала проверяется условие: есть ли лепесток у ромашки? Если есть, то выполняется циклическая часть. Стандартная функция ODD дает истинный результат, если значение I нечетно и ложный, если четно.

С помощью оператора READLN (ЛЕПЕСТОК) происходит ввод данных: если имеется очередной лепесток у ромашки, то нужно ввести символ "Y" от слова YES (да). Как только будет введен символ "N" от слова NO (нет), цикл завершает работу.

Картина 3. СНАЧАЛА СДЕЛАЙ — ПОТОМ ПРОВЕРЬ

Дока. Снова ты на старте, Фока! Но теперь тренер стоит на финише. После каждого забега он спрашивает: "Устал?". Если ты говоришь: "Нет", то снова бежишь от старта к финишу, а иначе, бег заканчивается.



BEGIN

END

Оператор цикла с последующим условием имеет вид

REPEAT

*операторы циклической
части программы*

UNTIL *логическое выражение*

Слова REPEAT (повторить) и UNTIL (до тех пор) являются служебными словами. Оператор цикла с последующим условием действует следующим образом. Операторы циклической части выполняются повторно (по крайней мере, один раз) до тех пор, пока значение логического выражения ложно. Условием прекращения циклических вычислений является истинное значение логического выражения. Итак, сначала выполняется цикличе-



ская часть, а затем проверяется условие. Необходимо обратить внимание на то, что эти действия прямо противоположны действиям оператора цикла с предварительным условием, где сначала проверяется условие, а затем выполняются операторы циклической части.

Следует подчеркнуть, что нижняя граница операторов циклической части четко обозначена словом UNTIL, поэтому нет необходимости заключать операторы циклической части в скобки вида BEGIN-END. В то же время и дополнительное наличие этих операторных скобок не является ошибкой.

Если в циклической части встречается оператор перехода, указывающий на метку за пределы цикла, то цикл может завершиться до его естественного окончания.

Пример использования оператора цикла с последующим условием: вычислить значение функции

$$V=X^2$$

при $X=2,4,6,8,10$

Фрагмент программы имеет вид:

```
X:=2;
REPEAT
  Y:=X*X;
  WRITELN(X:3,Y:5);
  X:=X+2
UNTIL X>10;
```

Здесь сначала задается первое значение аргумента $X=2$. Внутри циклической части выполняются следующие действия:

вычисляется значение Y при текущем значении X ;

значения X и Y выводятся на экран дисплея;

вычисляется новое значение аргумента X прибавлением числа 2 к предыдущему значению X .

Циклическая часть программы повторяется до тех пор, пока выражение $X>10$ не станет истинным. В процессе выполнения этой части программы переменные принимают следующие значения:

x	y	x:=x+2	логическое выражение	Значение
2	4	4	$4 > 10$	Ложь
4	16	6	$6 > 10$	Ложь
6	36	8	$8 > 10$	Ложь
8	64	10	$10 > 10$	Ложь
10	100	12	$12 > 10$	Истина

Задача 26. Составить программу, которая вычисляет последовательно все действия арифметического выражения до тех пор, пока в выражении не встретится знак равенства. В программе использован оператор цикла REPEAT, внутри которого находится оператор выбора: какой введен знак арифметического действия, такое будет вычисляться арифметическое выражение.

Программа имеет вид:

```
(*-----*)
!          ПРОГРАММА-КАЛЬКУЛЯТОР          !
!      БЕЗ СОХРАНЕНИЯ ПРИОРИТЕТА ЗА ДЕЙСТВИЯМИ      !
!-----*)
PROGRAM E9(INPUT,OUTPUT);
VAR
    ОПЕРАЦИЯ:CHAR;  (* АРИФМЕТИЧЕСКАЯ ОПЕРАЦИЯ *)
    ОТВЕТ:REAL;     (* РЕЗУЛЬТАТ ДЕЙСТВИЙ *)
    X:REAL;         (* ВВОДИМОЕ ЧИСЛО *)

BEGIN
    ОТВЕТ:=0;        (* ИСХОДНОЕ СОСТОЯНИЕ *)
    ОПЕРАЦИЯ:='+';   (* ИСХОДНОЕ СОСТОЯНИЕ *)
    WRITE('ВВЕДИТЕ ВЫРАЖЕНИЕ:');
    REPEAT
        READ(X);
        CASE ОПЕРАЦИЯ OF
            '+' : ОТВЕТ:=ОТВЕТ+X;
            '-' : ОТВЕТ:=ОТВЕТ-X;
            '*' : ОТВЕТ:=ОТВЕТ*X;
            '/' : ОТВЕТ:=ОТВЕТ/X;
        END;
        READ(ОПЕРАЦИЯ);
    UNTIL ОПЕРАЦИЯ '=' ;
    WRITELN(ОТВЕТ)
END.

=====
ВВЕДИТЕ ВЫРАЖЕНИЕ:
6*5-6+2=

РЕЗУЛЬТАТ ВЫЧИСЛЕНИЯ ВЫРАЖЕНИЯ:
2.600000E+01
=====
```

Картина 4. ДЕЛАЙ РОВНО СТОЛЬКО — СКОЛЬКО ЗАДАНО

Дока. Скажи, Фока, сколько раз ты пробегал стометровку на прошлых тренировках?

Фока. А я не считал. Я бегал до тех пор, пока были силы.

Дока. Вот именно! Но ведь может быть и другая ситуация во время тренировок: тренер не спрашивает, устал ты или нет, а дает задание: "Пробеги стометровку 5 раз".

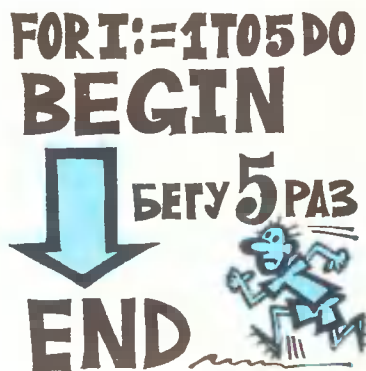
В тех случаях, когда заранее известно, сколько раз должен повториться цикл, используют оператор цикла с параметром.

Оператор цикла с параметром имеет следующую форму записи:

```
FOR _ i:=m1 _ TO _ m2 _ DO
BEGIN
    операторы циклической
    части программы
END
```

Здесь

FOR (для), TO (до), DO (выполнить) — служебные слова; i — параметр цикла; m1, m2 — начальное и конечное значение параметра цикла.



Циклическая часть программы выполняется повторно для каждого значения параметра цикла i от его начального m_1 до конечного значения m_2 включительно.

В качестве параметра цикла может быть только переменная, в качестве m_1 и m_2 могут быть выражения любого простого типа за исключением действительного.

Чаще всего параметр цикла i является переменной целого типа, а шаг его изменения равен $+1$ или -1 . Если значение параметра цикла возрастает, то шаг его изменения равен $+1$. Если значение параметра цикла уменьшается, то шаг его изменения равен -1 и в операторе цикла FOR вместо слова TO записывается слово DOWNT0.

Рассмотрим использование оператора цикла с параметром.

Пр и м е р 1. Пусть имеется фрагмент программы с переменными целого типа N, A, B:

```
FOR N:=1 TO 4 DO
  BEGIN
    A:=A*N;
    B:=2*N+1;
    WRITELN(A:3, B:3)
  END
```

Циклическая часть программы выполняется повторно 4 раза, при этом параметр цикла N изменяет свое значение от 1 до 4. В результате выполнения программы переменные получают следующие значения:

N	A	B
1	2	3
2	4	5
3	6	7
4	8	9

Этот же пример, но с убыванием значений параметра цикла, выглядит так:

```
FOR N:=4 TO DOWNT0 1 DO
  BEGIN
    A:=A*N;
    B:=2*N+1;
    WRITELN(A:3, B:3)
  END
```

Переменные получают следующие значения:

N	A	B
4	8	9
3	6	7
2	4	5
1	2	3

Пр и м е р 2. Вычислить значения $Y = X^2$ при $X = 2, 4, 6, 8, 10$.

Программа имеет вид


```

X:=2;
FOR K:=5 DOWNTO 1 DO
BEGIN
  Y:=X*X;
  WRITELN(X:3,Y:5);
  X:=X*2
END

```

Здесь K — счетчик, переменная целого типа. В результате выполнения этого фрагмента программы переменные получают следующие значения:

K	X	Y
1	2	4
2	4	16
3	6	32
4	8	64
5	10	100

Решение этой задачи приведено для всех видов операторов цикла: WHILE, REPEAT, FOR. Следует обратить внимание на программы и результаты.

Особенности оператора цикла с параметром. Если циклическая часть программы содержит только один оператор, то операторные скобки BEGIN—END можно не указывать. В этом случае цикл с параметром записывают в следующем виде:

```
FOR _ i:=m1 _ TO _ m2 _ DO _ оператор
```

или

```
FOR _ i:=m1 _ TO _ m2 _ DO
  оператор
```

Параметр цикла *i* не должен переопределяться внутри циклической части. Если шаг изменения параметра цикла равен +1 и $m_1 > m_2$, то циклическая часть не выполнится ни разу.

После естественного завершения цикла значение параметра цикла не определено. Это означает, что при последнем выполнении циклической части значение $i=m_2$, а после ухода за пределы цикла значение *i* теряется.

Задача 27. Вывести на экран дисплея все буквы латинского алфавита. Поскольку буквы латинского алфавита упорядочены (см. Приложение 2), то можно составить следующую программу:

```

(*-----*)
!   ПЕЧАТЬ БУКВ ЛАТИНСКОГО АЛФАВИТА   !
(*-----*)
PROGRAM E33(INPUT,OUTPUT);
  VAR SIM : CHAR;
BEGIN
  WRITELN('ЛАТИНСКИЙ АЛФАВИТ :');
  FOR SIM:='A' TO 'Z' DO
    WRITE( SIM )
  END.

=====
=
=   ЛАТИНСКИЙ АЛФАВИТ :   =
=   ABCDEFGHIJKLMNOPQRSTUVWXYZ   =
=
=====

```


Задача 28. Каждый будний день недели рыбак ходил ловить рыбу. Сколько он поймал рыб в среднем за день, если ежедневный улов был таким: 8, 7, 5, 9, 7, 9.

В программе сначала вводится количество дней, обозначенное именем N, а затем организуется цикл по числу дней. Внутри цикла вводится очередное значение R — числа пойманных рыб за один день. Это значение прибавляется к накапливаемой сумме S.

По окончании выполнения цикла вычисляется среднее значение, обозначенное также именем S.

```
(*-----*
!      ЛОВИСЬ РЫБКА БОЛЬШАЯ И МАЛЕНЬКАЯ      !
*-----*)
PROGRAM EX8(INPUT,OUTPUT);
VAR
  I:INTEGER;  (* СЧЕТЧИК ДНЕЙ          *)
  N:INTEGER;  (* КОЛИЧЕСТВО ДНЕЙ        *)
  R:REAL;     (* ЧИСЛО РЫБ ЗА ДЕНЬ       *)
  S:REAL;     (* СРЕДНЕЕ ЧИСЛО РЫБ      *)
BEGIN
  R:=0;
  WRITELN('СКОЛЬКО ДНЕЙ РЫБАК ЛОВИЛ РЫБУ?');
  READ( N );
  FOR I:=1 TO N DO
    BEGIN
      WRITE('СКОЛЬКО РЫБ ОН ПОЙМАЛ ЗА ДЕНЬ? ');
      READ(R);
      S:=S+R
    END;
  S:=S/N;
  WRITELN('-----');
  WRITELN('СРЕДНЕЕ ЧИСЛО РЫБ ЗА ДЕНЬ =',S:5:1)
END.
=====
=
=   СКОЛЬКО ДНЕЙ РЫБАК ЛОВИЛ РЫБУ?   =
=   6                                 =
=   СКОЛЬКО РЫБ ОН ПОЙМАЛ ЗА ДЕНЬ?   8   =
=   СКОЛЬКО РЫБ ОН ПОЙМАЛ ЗА ДЕНЬ?   7   =
=   СКОЛЬКО РЫБ ОН ПОЙМАЛ ЗА ДЕНЬ?   5   =
=   СКОЛЬКО РЫБ ОН ПОЙМАЛ ЗА ДЕНЬ?   9   =
=   СКОЛЬКО РЫБ ОН ПОЙМАЛ ЗА ДЕНЬ?   7   =
=   СКОЛЬКО РЫБ ОН ПОЙМАЛ ЗА ДЕНЬ?   9   =
=   -----                           =
=   СРЕДНЕЕ ЧИСЛО РЫБ ЗА ДЕНЬ =   7.5   =
=                                     =
=====
```

Задача 30. Составить программу, результатом выполнения которой является таблица значений градусов температуры по Цельсию и Фаренгейту.

Программа имеет вид:

```
(*-----*)
! ТАБУЛИРОВАНИЕ    ТЕМПЕРАТУРЫ    ОТ 0    ДО    !
! 20  ГРАДУСОВ    ПО  ЦЕЛЬСИЮ    СОВМЕСТНО С    !
! ЭКВИВАЛЕНТАМИ ДО 0.1 ГРАДУСА ПО  ФАРЕНГЕЙТУ    !
*-----*)
PROGRAM E6(OUTPUT);
CONST
  CONST1 =1.8;      (* МНОЖИТЕЛЬ ПЕРЕВОДА      *)
  CONST2 =32.0;     (* СЛАГАЕМОЕ ПЕРЕВОДА      *)
VAR
  CENTEMP:INTEGER;  (* ТЕМПЕРАТУРА ПО ЦЕЛЬСИЮ *)
  FARTEMP:REAL;     (* ТЕМПЕРАТУРА ПО ФАРЕНГЕЙТУ *)
BEGIN
  WRITELN('ПО ЦЕЛЬСИЮ * ПО ФАРЕНГЕЙТУ');
  FOR CENTEMP:=0 TO 20 DO
    BEGIN
      FARTEMP:=CENTEMP*CONST1+CONST2;
      WRITELN(CENTEMP:3,' ':8,'*', ' ',FARTEMP:6:1);
    END;
  END.
=====
=      ПО ЦЕЛЬСИЮ * ПО ФАРЕНГЕЙТУ      =
=      0          *      32.0            =
=      1          *      33.8            =
=      2          *      35.6            =
=      3          *      37.4            =
=      4          *      39.2            =
=      5          *      41.0            =
=      6          *      42.8            =
=      7          *      44.6            =
=      8          *      46.4            =
=      9          *      48.2            =
=     10          *      50.0            =
=     11          *      51.8            =
=     12          *      53.6            =
=     13          *      55.4            =
=     14          *      57.2            =
=     15          *      59.0            =
=     16          *      60.8            =
=     17          *      62.6            =
=     18          *      64.4            =
=     19          *      66.2            =
=     20          *      68.0            =
=====
```



Рис. 2. Окружность, удаленная от центра координат

Задача 31. Составить программу игры "Угадай число". За дисплеем сидят два человека. При выполнении программы после появления на экране текста

ВВЕДИТЕ УГАДЫВАЕМОЕ ЧИСЛО

один из игроков отворачивается в сторону, а другой вводит любое целое число (допустимое для данной машины). Число исчезает с экрана за счет того, что на экране дисплея помещается 24 строки, а организован переход на новую строку 25 раз. Теперь можно смотреть на экран и играть.


```

(*)-----!
!                УГАДАЙ ЧИСЛО                !
-----*)
PROGRAM E24(INPUT,OUTPUT);
  LABEL 5;
  VAR
    I:INTEGER;  (* ПЕРЕМЕННАЯ ЦИКЛА *)
    N:INTEGER;  (* ПЕРЕМЕННАЯ УГАДЫВАЮЩЕГО ЧИСЛА *)
    L:INTEGER;  (* УГАДЫВАЕМОЕ ЧИСЛО *)
    K:INTEGER;  (* СЧЕТЧИК КОЛ-ВА УГАДЫВАНИЙ *)

  BEGIN
    K:=0;
    WRITELN('ВВЕДИТЕ УГАДЫВАЕМОЕ ЧИСЛО:');
    READ(L);
    FOR I:=1 TO 25 DO
      BEGIN
        WRITELN;
      END;
    5: WRITELN('УГАДАЙТЕ ЧИСЛО:');
    READ(N);
    K:=K+1;
    IF N=L THEN WRITELN('ПРАВИЛЬНО')
    ELSE
      IF N>L THEN
        BEGIN
          WRITELN('ВАШЕ ЧИСЛО БОЛЬШЕ');
          GOTO 5;
        END
      ELSE
        BEGIN
          WRITELN('ВАШЕ ЧИСЛО МЕНЬШЕ');
          GOTO 5;
        END;
    IF K 1 THEN WRITELN('НО, ВАМ ПРОСТО ПОВЕЗЛО. ');
    IF K<=5 THEN WRITELN('МОЛОДЦЫ! ВЫ БЫСТРО УГАДАЛИ. ')
    ELSE WRITELN('НО ВЫ ДОЛГО УГАДЫВАЛИ. ');
  END.

```



```

=====
- ВВЕДИТЕ УГАДЫВАЕМОЕ ЧИСЛО:
- 67
- ( ЧИСЛО ИСЧЕЗАЕТ С ЭКРАНА ЗА СЧЕТ ОПЕРАТОРА
-   FOR I:=1 TO 25 DO WRITELN; )
-
- УГАДАЙТЕ ЧИСЛО:
- 34
- ВАШЕ ЧИСЛО МЕНЬШЕ
- УГАДАЙТЕ ЧИСЛО:
- 78
- ВАШЕ ЧИСЛО БОЛЬШЕ
- УГАДАЙТЕ ЧИСЛО:
- 67
- ПРАВИЛЬНО
- МОЛОДЦЫ! ВЫ БЫСТРО УГАДАЛИ.
=====

```

Картина 5. СКОЛЬКО МАТРЕШЕК В МАТРЕШКЕ?

Дока. Циклы могут быть вложены один в другой. При использовании вложенных циклов необходимо составлять программу таким образом, чтобы внутренний цикл полностью укладывался в циклическую часть внешнего



цикла. Внутренний цикл может также в свою очередь содержать другой внутренний цикл (циклы).

Фока. Получается, что вложенные циклы напоминают матрешек?

Дока. Это удачное сравнение. Но надо заметить, что внутри одной матрешки находится только одна матрешка, внутри которой опять одна матрешка и т.д. Внутри же одного цикла могут быть расположены сразу несколько, каждый из которых может содержать также несколько циклов.

```
FOR I:=1 TO 5 DO
  FOR J:=1 TO 7 DO
    FOR K:=1 TO 4 DO
      BEGIN
```

```
        I
```

```
      END
```

```
    FOR I:=K TO N DO
```

```
      BEGIN
```

```
        FOR J:=N TO L DO
```

```
          BEGIN
```

```
            I
```

```
          END
```

```
        REPEAT
```

```
          I
```

```
      UNTIL A>B
```

```
    END
```

Задача 32. Будем считать билет счастливым, если его номер является шестизначным числом, у которого сумма первых трех цифр равна сумме трех последних и равна 13.

Подсчитаем число счастливых билетов для всех шестизначных чисел. В программе организуем тройной цикл для определения количества первых трехзначных чисел, сумма цифр которых равна 13. Это количество чисел обозначим именем N. Тогда общее количество счастливых билетов равно N^2 .

```
(*-----*)
I      СЧАСТЛИВЫЕ БИЛЕТЫ      I
(*-----*)
```

```
PROGRAM L80(INPUT,OUTPUT);
```

```
VAR
```

```
  I,J,K:INTEGER; (* СЧЕТЧИКИ *)
  N:INTEGER;      (* ЧИСЛО СЧАСТЛИВЫХ БИЛЕТОВ *)
```



```

BEGIN
  N:=0;
  FOR I:=0 TO 9 DO
    FOR J:=0 TO 9 DO
      FOR K:=0 TO 9 DO
        IF I+J+K=13 THEN N:=N+1;
      N:=SQR(N);
    WRITE('ЧИСЛО СЧАСТЛИВЫХ БИЛЕТОВ =', N:6)
  END.
=====
=
=   ЧИСЛО СЧАСТЛИВЫХ БИЛЕТОВ = 5625   =
=
=====

```

Задача 33. Вычислить $y=2x+t$ при всех значениях $x=1,5; 2; 2,5; 3$ и $t=1; 3; 5$.
 Можно составить программу следующим образом: X — параметр внешнего цикла, T — параметр внутреннего цикла. Тогда при одном значении X переменная T принимает все свои значения 1,3,5. При другом значении X снова переменная T принимает все свои значения и т. д. Всего получится 12 значений Y .

```

      I      ВЛОЖЕННЫЕ ЦИКЛЫ      I
-----*-----
PROGRAM SIMB1 (INPUT, OUTPUT);
CONST A='-----';
VAR
  Y: REAL;      (* ФУНКЦИЯ *)
  X, T: REAL;    (* АРГУМЕНТЫ *).
BEGIN
  WRITELN(A);
  WRITELN(' X      T      Y');
  WRITELN(A);
  X:=1.5;
  WHILE X<=3 DO
    BEGIN
      T:=1;
      WHILE T<=5 DO
        BEGIN
          Y:=2*X+T;
          WRITELN(X:4:1, ' ':3, T:4:1, ' ':3, Y:6:2)
          T:=T+2
        END;
        X:=X+0.5
      END
    END
  END.

```

```

=====
=
=   -----
=   X      T      Y
=   -----
=   1.5    1.0    4.00
=   1.5    3.0    6.00
=   1.5    5.0    8.00
=   2.0    1.0    5.00
=   2.0    3.0    7.00
=   2.0    5.0    9.00
=   2.5    1.0    6.00
=   2.5    3.0    8.00
=   2.5    5.0   10.00
=   3.0    1.0    7.00
=   3.0    3.0    9.00
=   3.0    5.0   11.00
=
=====

```

**МОЖНО ЛИ
ОПЕРАТОР ЦИКЛА
WHILE
ПО ПЕРЕМЕННОЙ T
ЗАМЕНИТЬ НА ЦИКЛ
FOR ?**

НЕСТАНДАРТНЫЕ ТИПЫ

Картина 1. ЕЩЕ РАЗ О ТИПАХ

Дока. Мы установили, что тип данных задает набор значений, которые могут принимать данные. Например, данные целого типа могут принимать значения целых чисел от -32768 до +32767, данные логического типа — значения — TRUE и FALSE. Задание типа данных устанавливает также ограниченный набор операций, которые могут выполняться над этими данными. Например, над действительными данными можно выполнять операции $+$, $-$, $*$, $/$, но нельзя производить операции DIV и MOD, которые предназначены для данных целого типа. Типизация данных значительно уменьшает возможность появления ошибок. Если все же ошибки возникают, то компьютер сам их обнаруживает и сообщает нам об этом. Это особенно важно, если в программе встречаются абсурдные ошибки.

Фока. Да разве могут быть абсурдные ошибки?

Дока. Сколько угодно. Пусть, например, переменная В обозначает число яблок, а переменная G — ускорение свободного падения. Если эти переменные описать в одном разделе

```
VAR B,G: REAL;
```

то при ошибочной операции $B+G$ компьютер не откликнется на ошибку и выдаст результат, хотя он и абсурдный.

Учитывая "ценность" типизации данных, в языке Паскаль помимо стандартных типов (целого, действительного, логического, символьного) введены и другие типы — нестандартные. Они делятся на простые и сложные.

К простым нестандартным типам относятся перечисляемый и ограниченный, которые мы сейчас и рассмотрим.



Картина 2. ПЕРЕЧИСЛЯЕМЫЙ ТИП

Дока. Этот тип данных получил название перечисляемого (иногда говорят перечислимого), потому что он задается в виде перечисления некоторых значений. Перечисляемый тип может быть задан самим программистом в зависимости от того, какую задачу он решает.

Перечисляемый тип состоит из списка констант. Переменные этого типа могут принимать значение любой из этих констант.

Описание перечисляемого типа имеет вид:

TYPE — имя типа = (список констант);

VAR — имя переменной: имя типа;



Здесь под константой понимается особый вид констант — констант, задаваемых пользователем. Под списком понимается перечень констант, разделенных запятыми. Сам список заключается в круглые скобки.

Пример описания перечисляемого типа:

```
TYPE ГОД = (ЗИМА, ВЕСНА, ЛЕТО, ОСЕНЬ);
```

```
VAR A: ГОД;
```

Здесь ГОД — имя перечисляемого типа; ЗИМА, ВЕСНА, ЛЕТО, ОСЕНЬ — константы; А — переменная, которая может принимать значение любой из констант.

В языке допускается указывать константы перечисляемого типа непосредственно в разделе переменных без использования раздела TYPE, например

```
VAR A: (ЗИМА, ВЕСНА, ЛЕТО, ОСЕНЬ);
```

Однако хорошим стилем программирования является описание данных с использованием раздела TYPE, так как это помогает сократить появление ошибок и облегчает отладку программы.

Каждая из констант имеет порядковый номер, счет начинается с нуля. Так, ЗИМА имеет порядковый номер 0, ВЕСНА — 1, ЛЕТО — 2, ОСЕНЬ — 3.

Упорядоченность констант позволяет применять к ним операции отношения $<$, $<=$, $=$, $>$, $>=$. Результат операции будет логического типа (TRUE, FALSE).

Для приведенного примера имеем

```
ЗИМА < ВЕСНА < ЛЕТО < ОСЕНЬ
```

К переменным и константам перечисляемого типа можно применять стандартные функции ORD, PRED, SUCC (см. Приложение 5).

В языке Паскаль нельзя вводить и выводить данные перечисляемого типа с помощью операторов READ и WRITE.

Задача 34. Имеется перечень зверей: лиса, волк, заяц, зубр, тигр, лев, медведь, косуля, олень, барс. Составить программу, позволяющую определить порядковый номер тигра N1 и порядковый номер зверя после косули N2.

```
(*----- ПЕРЕЧИСЛЯЕМЫЙ ТИП -----*)
1                                     1
```

```
PROGRAM PODS(INPUT,OUTPUT);
  TYPE ЗВЕРЬ=(ЛИСА,ВОЛК,ЗАЯЦ,ЗУБР,
              ТИГР,ЛЕВ,МЕДВЕДЬ,
              КОСУЛЯ,ОЛЕНЬ,БАРС);
```

```
VAR
  P1,P2 : ЗВЕРЬ;
  N1,N2 : INTEGER;
```




```

BEGIN
  P1:=ТИГР;
  P2:=SUCC(КОСУЛЯ);
  N1:=ORD(P1)+1;
  N2:=ORD(P2)+1;
  WRITELN('ПОРЯДКОВЫЙ НОМЕР ТИГРА = ', N1:2);
  WRITE('ПОРЯДКОВЫЙ НОМЕР ЗВЕРЯ ',
        'ПОСЛЕ КОСУЛИ = ', N2:2)
END.

```

```

=====
=
=   ПОРЯДКОВЫЙ НОМЕР ТИГРА = 5   =
=   ПОРЯДКОВЫЙ НОМЕР ЗВЕРЯ ПОСЛЕ КОСУЛИ = 9   =
=
=====

```

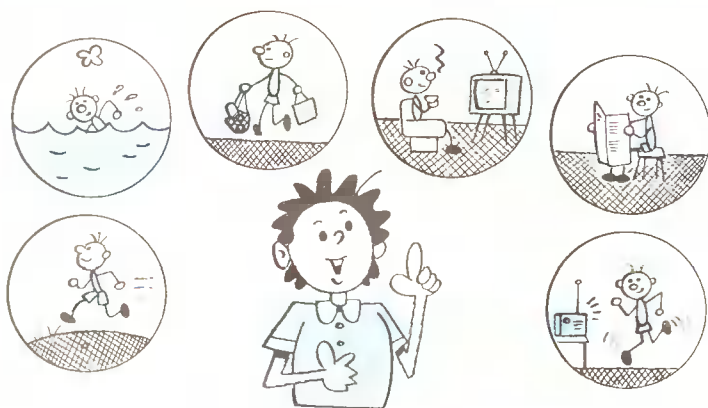
Так как счет перечисляемых данных начинается от нуля, а не от единицы, то для вычисления порядковых номеров N1 и N2 необходимо прибавить единицу.

Для машин типа ДВК нужно осторожно пользоваться русскими словами, так как некоторые буквы запрещены при составлении программ, чаще всего это буквы Ч, Ю, Щ, Ъ.

Фока. Вот это, да! А как же мне перечислить дни недели, если среди них встречается "ЧЕТВЕРГ" с буквой "Ч", или месяцы года "ИЮНЬ", "ИЮЛЬ" с буквой "Ю"?

Дока. Это очень неприятный момент и приходится идти на различные ухищрения: сокращать слова, использовать латинские буквы, английские слова и т.д. Но здесь нужно учесть, что оператор вывода WRITE имеет меньше ограничений на русские буквы (как правило, запрещены "Щ" и "Ъ"), поэтому результаты выполнения программы могут иметь нормальный вид, вместо твердого знака используют апостроф, а слово с буквой "Щ" заменяют на синоним.

Задача 35. Рассмотрим, например, следующую задачу: мой друг в свободное время вечером занимается музыкой по средам и пятницам, спортом — в остальные будние дни. Как составить программу, чтобы результатом ее выполнения были последовательно расположенные виды занятий на всю неделю, начиная с понедельника?




```

(*-----*)
I      СВОБОДНОЕ ВРЕМЯ      I
(*-----*)
PROGRAM DEN(INPUT,OUTPUT);
  TYPE WEEK=( MONDAY,TUESDAY,WEDNESDAY,
               THURSDAY, FRIDAY,SATURDAY,
               SUNDAY );
  VAR DAY:WEEK;
BEGIN
  FOR DAY:=MONDAY TO SUNDAY DO
    IF (DAY=WEDNESDAY)OR(DAY=FRIDAY)
      THEN WRITELN('МУЗЫКА')
    ELSE IF( DAY=SUNDAY )
      THEN WRITELN('УРА! ОТДЫХ! ')
      ELSE WRITELN('СПОРТ')
END.
=====
=    СПОРТ    =
=    СПОРТ    =
=    МУЗЫКА    =
=    СПОРТ    =
=    МУЗЫКА    =
=    СПОРТ    =
=    УРА! ОТДЫХ!  =
=              =
=====

```

Картина 3. ОГРАНИЧЕННЫЙ ТИП

Дока. Если какая-то переменная принимает не все значения своего типа, а только значения, содержащиеся в некотором ограниченном диапазоне, то ее можно рассматривать как переменную ограниченного типа (иногда говорят — интервального типа).

Пусть, например, в разделе описания указан перечисляемый тип ГОД:

```

TYPE ГОД = (ЯНВАРЬ, ФЕВРАЛЬ, МАРТ, АПРЕЛЬ, МАЙ,
            ИЮНЬ, ИЮЛЬ, АВГУСТ, СЕНТЯБРЬ, ОКТЯБРЬ, НОЯБРЬ,
            ДЕКАБРЬ);

```

А нам предстоит решать задачи, связанные только с весенним и летним временами года. Тогда можно ввести ограниченный тип:

```

ВЕСНАЛЕТО = МАРТ . . АВГУСТ

```

который является частью перечисляемого типа.

Как видно из примера, при определении ограниченного типа указываются начальное и конечное значения диапазона, разделенные двумя точками.

Общий вид описания ограниченного типа имеет вид

```

TYPE _ имя типа = константа 1 . . константа N;
VAR _ имя переменной: имя типа;

```

Например:

```

TYPE T = 1..100 ;
VAR A,B: T ; .

```


Переменные А и В могут принимать любое значение из диапазона 1 . . 100. При использовании ограниченного типа должны выполняться следующие правила:

- а) обе граничные константы должны быть одинакового типа;
- б) начальное значение не должно быть больше конечного;
- в) тип констант может быть любой простой, кроме действительного, например

```
K = 15..70 ;  
LET = 'A'..'Z' ;  
ЛЕТО = июнь..август ;
```

Если константы имеют стандартный тип, то описание ограниченности типа можно делать непосредственно в разделе переменных, например:

```
VAR S1, S2: 1..100 ;  
SIM : 'A'..'R' ;
```

Тип констант называется базовым. Над переменными ограниченного типа разрешается выполнять все операции, которые допустимы для данных его базового типа.

Использование ограниченных типов позволяет программисту более четко и определенно изложить свою задачу. Указывая в явном виде диапазон изменения значений переменных, ограниченный тип делает программы более понятными и наглядными.

Например, если в программе переменная К принимает только целые значения 4, 5, 6, 7, 8, 9, 10, то лучше дать описание

```
VAR K: 4 . . 10;
```

чем

```
VAR K: INTEGER;
```

В случае выхода значения К за диапазон 4 . . . 10 в первом варианте будет выдано диагностическое сообщение, которое поможет программисту обнаружить ошибку. Во втором варианте может быть выдан неверный результат, что затруднит поиск ошибки. Вторым вариантом описания переменных рекомендуется использовать только тогда, когда диапазон значений переменной либо заранее неизвестен, либо занимает весь допустимый в языке диапазон для рассматриваемого типа.

Действие седьмое

СЛОЖНЫЕ ТИПЫ

Картина 1. ЗРИТЕЛИ И ПАССАЖИРЫ

Дока. Когда мы имеем дело с большим количеством данных одного типа, то обозначать их различными именами и обрабатывать становится затруднительно. Например, пусть имеется совокупность действительных чисел

1.6, 14.9, -5.0, 8.5, 0.46.

Конечно, можно каждое число обозначить своим именем, например
 $A_1, SB, M, NIC, NUM.$

Но можно для всей совокупности чисел ввести одно обобщающее имя, а индексом отметить конкретное число, например

A_1, A_2, A_3, A_4, A_5

Да мы и в обыденной жизни встречаемся с таким положением. Ну, вот сидят незнакомые нам люди в театре и смотрят спектакль. Мы не знаем их конкретных имен, но можем дать им одно обобщающее имя — зритель. Это позволяет нам обратиться к конкретному зрителю: "Уважаемый зритель, сидящий на 13 месте, подойдите, пожалуйста, к сцене. Вам вручается ценный подарок".

Фока. Или в автобусе. Подходят ко мне и говорят: "Уважаемый пассажир, сидящий на 7-м месте, Вы оказались миллионным пассажиром! Получите, пожалуйста, бесплатный годовой проездной билет!"

Дока. Ну, Фока, ты и размечтался ... Придется вернуться с небес на землю, к программированию.

Обобщенные имена используются и в математике, и в программировании в том случае, когда рассматриваются массивы данных.

Под массивом понимается конечная совокупность данных одного типа, упорядоченных по значениям индекса. Каждый элемент массива обозначается именем массива с индексом.

В математике, как правило, индекс либо заключается в круглые скобки, либо указывается несколько ниже имени массива, например:

$A(1), A(2), A(3), A(4), A(5)$

или

$A_1, A_2, A_3, A_4, A_5,$



или в общем виде

$\{A_i\}$, где $i = 1, 2, \dots, n$.

В языке Паскаль индекс заключается в квадратные скобки. Для рассмотренного примера элементами массива A являются

$A[1]=1.6$, $A[2]=14.9$, $A[3]=-5.0$, $A[4]=8.5$, $A[5]=0.46$.

Если в программе используется массив, то он должен быть описан либо в разделе переменных `VAR`, либо в разделе типов `TYPE`. Рассмотрим сначала описание массива в разделе переменных `VAR`. Форма описания имеет вид

`VAR _ имя массива: ARRAY [t1] OF _ t2;`

Здесь: `ARRAY` (массив) и `OF` (из) — служебные слова; t_1 — тип индекса, в качестве которого может быть любой простой тип, кроме стандартных типов `REAL` и `INTEGER`; t_2 — тип элементов массива, который называется базовым типом. В качестве базового типа может быть простой или сложный тип, допустимый в языке Паскаль.

Для предыдущего примера описание массива имеет вид

`VAR A: ARRAY [1 .. 5] OF REAL;`

Здесь A — имя массива, элементы которого имеют базовый тип `REAL`. Тип индекса — ограниченный от 1 до 5.

Во многих задачах с массивами нумерация индекса начинается от 1 и заканчивается каким-либо положительным числом. Однако это совсем необязательное условие. Если, например, в программе обрабатывается численность населения земного шара по григорианскому календарю до 1992 г., то массив можно описать следующим образом:

`VAR НАСЕЛЕНИЕ: ARRAY[1582 .. 1992] OF INTEGER`

Если же в программе обрабатывается численность населения города Рима до нашей эры, то описание массива имеет вид:

`VAR НАСЕЛЕНИЕ: ARRAY[-754 .. -1] OF INTEGER`

Поскольку значениями индексов могут быть не только числа, то рассмотрим и другие примеры описания массивов.

1. Частота появления латинских букв в тексте:

`VAR ЧАСТОТА: ARRAY ['A' .. 'Z'] OF INTEGER;`

2. Среднегодовая температура воздуха на островах:

`TYPE ОСТРОВ = (ГАИТИ, СУМАТРА, ФЕМАРН, ТАЙМЫР,
ТАИТИ, ЯВА, ТАСМАНИЯ, СИЦИЛИЯ);`

`VAR T: ARRAY [ОСТРОВ] OF REAL;`

3. Имя самого рослого мужчины среди указанных имен:

`TYPE ИМЯ = (ВАНЯ, НИНА, КАТЯ, ФЕДЯ, КОЛЯ, МАША, МАРИНА);`

`VAR ПОЛ: ARRAY [ИМЯ] OF (МУЖ, ЖЕН);`

`РОСТ: ARRAY [ИМЯ] OF 140 .. 240;`

Следует обратить внимание на то, что поскольку тип индекса не может быть стандартным целым или действительным типом, то нельзя делать следующее описание массива (часто встречающаяся ошибка):

```
VAR A: ARRAY [5] OF REAL;
```

или

```
VAR A: ARRAY [INTEGER] OF REAL;
```

Примеры правильного описания массивов:

```
VAR  
  MASSIV : ARRAY [1..N] OF REAL ;  
  ГОД    : ARRAY [ЯНВАРЬ..ДЕКАБРЬ] OF INTEGER;  
  L      : ARRAY [СТРОКА] OF BOOLEAN ;  
  M1     : ARRAY [CHAR] OF КРУГ ;
```

Если несколько массивов имеют одинаковый тип индексов и одинаковый базовый тип, то допускается в описании объединять массивы в список, например

```
VAR A,B,C: ARRAY [1..50] OF REAL
```

Здесь объявлено списком три массива A,B,C действительных чисел. каждый из которых содержит по 50 элементов (от 1 до 50):

```
A[1], A[2],..., A[50],  
B[1], B[2],..., B[50],  
C[1], C[2],..., C[50]
```



В качестве индекса может быть выражение, частным случаем которого является константа или переменная. Элемент массива иначе называется переменной с индексом. В отличие от нее переменная без индекса называется простой переменной.

Элементы массива могут стоять как в левой части оператора присваивания, так и в выражениях. Над элементами массива можно производить те же операции, которые допустимы для данных его базового типа. Если базовый тип есть INTEGER, то допустимы все операции над данными целого типа, включая и стандартные функции. Примеры использования элементов массива в разделе операторов:

```
B[5] := B[3] + 1 ;  
SUM  := SUM - C[K] ;  
P1   := A[2*1+1] ;
```

Для ввода и вывода численных значений массива используются циклы. Например, цикл

```
FOR I:=1 TO 9 DO  
  READ(A[I]) ;
```

организует ввод девяти значений элементов массива A. Цикл

```
FOR I:=1 TO 9 DO  
  WRITE(A[I]) ;
```

организует вывод девяти значений элементов того же массива.

Пример использования массивов в программе. Вычислить сумму пятнадцати целых чисел. Для обозначения чисел введем имя массива X, элементы которого пусть имеют индекс I. Сумму обозначим именем SUM. Тогда

$$SUM = X[1] + X[2] + \dots + X[15].$$

Фрагмент программы имеет вид:

```
VAR X : ARRAY [1..15] OF INTEGER ;
...
SUM:=0 ;
FOR I:=1 TO 15 DO
  BEGIN
    READ( x[I] ) ;
    SUM:= SUM + x[I]
  END;
```

Внутри цикла вводится значение одного элемента массива и оно прибавляется к значению текущей суммы SUM. Цикл повторяется 15 раз. Элементы массива описаны как данные целого типа (INTEGER), а индекс имеет ограниченный тип 1..15. Следует обратить внимание на то, что параметр цикла I используется в качестве индекса.

Итак, переменная I, с одной стороны, используется как параметр цикла, а с другой стороны, в качестве индекса. Как в этом случае правильно описать переменную I? Описание можно сделать двояко:

а) в разделе переменных VAR указать переменную I как ограниченного типа. Например:

```
VAR
  X : ARRAY [1..15] OF INTEGER
  I : 1..15 ;
```

б) учитывая, что элементами ограниченного типа являются целые числа, можно ввести описание INTEGER, например:

```
VAR
  X : ARRAY [1..15] OF INTEGER ;
  I : INTEGER ;
```

Поскольку в языке Паскаль требуется описание всех переменных, то необходимо описать еще и переменную SUM. Естественно, что она имеет тип INTEGER, так как все суммируемые числа X[I] имеют целый тип.

В языке Паскаль помимо явного описания массивов в разделе переменных имеется другая форма описания, состоящая из двух этапов.

Сначала в разделе описания типов TYPE указывается тип массива. Затем в разделе описания переменных VAR перечисляются массивы, относящиеся к указанному типу.

Введение типа массива увеличивает раздел описаний, но в то же время упрощает отладку программы и удерживает от абсурдных ошибок.

Указание типов в разделе описаний помогает достичь логической ясности программы и является хорошим стилем программирования.

TYPE _ имя типа = ARRAY [t₁] OF _ t₂;

VAR _ имя массива : имя типа;

Здесь t₁ — тип индекса; t₂ — базовый тип элементов массива.

Пусть, например, в программе используется массив R, состоящий из десяти элементов действительного ТИПА. Обозначим тип массива именем MAS. Тогда описание массива можно выполнить следующим образом:


```
TYPE MAS = ARRAY [1..10] OF REAL ;
VAR R: MAS ;
```

Если в программе несколько массивов, например R, A, B, C имеют тип MAS, то изменится лишь раздел описания переменных

```
VAR R, A, B, C: MAS;
```

В разделе операторов программы используются массивы R, A, B, C. Тип массива MAS введен формально только в разделе описаний и нигде в программе не указывается и не обрабатывается.

Рассмотрим, как составлять программу вычисления суммы чисел. Программу составим в двух вариантах. В первом варианте каждое вводимое число сразу же обрабатывается, поэтому массив не используется (задача "Умная хозяйка"). Во втором варианте используется массив, все значения которого сначала вводятся, а потом обрабатываются (задача "Глупая хозяйка").

Задача 36. Четыре дня подряд хозяйка ходила на базар и что-нибудь покупала. Каждый день она подсчитывала расходы с учетом предыдущих дней. Поэтому она всегда знала израсходованную сумму денег.

Требуется составить программу, как действовала хозяйка.

```
(*-----*
!           " УМНАЯ ХОЗЯЙКА "           !
!-----*)

PROGRAM A30A(INPUT,OUTPUT);
CONST N=4;          (* ЧИСЛО ДНЕЙ          *)
VAR
  A : REAL ;        (* РАСХОДЫ ЗА ДЕНЬ      *)
  SUM : REAL ;      (* СУММА РАСХОДОВ  *)
  I : INTEGER ;     (* СЧЕТЧИК ДНЕЙ    *)
BEGIN
  SUM:=0;
  FOR I:= 1 TO N DO
    BEGIN
      WRITE('КАКОВЫ РАСХОДЫ ЗА ДЕНЬ? ');
      READLN(A);
      SUM:= SUM + A ;
      WRITELN('СУММА РАСХОДОВ:', SUM:6:2 );
    END;
  WRITELN('-----');
  WRITELN('РАСХОДЫ ЗА ВСЕ ДНИ =', SUM:6:2 )
END.

=====
=
= КАКОВЫ РАСХОДЫ ЗА ДЕНЬ? 4.60
= СУММА РАСХОДОВ: 4.60
= КАКОВЫ РАСХОДЫ ЗА ДЕНЬ? 6.10
= СУММА РАСХОДОВ: 10.70
= КАКОВЫ РАСХОДЫ ЗА ДЕНЬ? 2.95
= СУММА РАСХОДОВ: 13.65
= КАКОВЫ РАСХОДЫ ЗА ДЕНЬ? 4.00
= СУММА РАСХОДОВ 17.65
= -----
= РАСХОДЫ ЗА ВСЕ ДНИ = 17.65
=
=====
```





Поскольку в задаче заранее оговорено количество дней, то в программе используется оператор цикла FOR. Внутри циклической части вводится значение расходов за день A и сразу же подсчитывается сумма расходов SUM. Таким образом нет необходимости в использовании массива.

Задача 37. Четыре дня подряд другая хозяйка ходила на базар и каждый день что-нибудь покупала, но расходы не считала. Через четыре дня хозяйка решила подсчитать сумму израсходованных денег. Стала вспоминать, сколько денег она потратила в первый день, во второй и т.д.

Требуется составить программу, как действовала хозяйка. В программе используются два независимых цикла и массив расходов. С помощью первого цикла вводятся ежедневные расходы. Второй цикл предназначен для подсчета суммы расходов.

```

(*-----*
!      " ГЛУПАЯ ХОЗЯЙКА "      !
!-----*)
PROGRAM A32A(INPUT,OUTPUT);
CONST N=4;
TYPE T = ARRAY[1..N] OF REAL; (* КОЛИЧЕСТВО ДНЕЙ *)
VAR (* ТИП МАССИВА *)
    A : T; (* МАССИВ РАСХОДОВ *)
    SUM : REAL; (* СУММА РАСХОДОВ *)
    I : INTEGER; (* СЧЕТЧИК ДНЕЙ *)
BEGIN
    (* ЕЖЕДНЕВНЫЕ РАСХОДЫ *)
    -----*)
    FOR I:= 1 TO N DO
        BEGIN
            WRITE('РАСХОДЫ ЗА ДЕНЬ? ');
            READ( A[I] );
            END;
            (* БАТЮШКИ МОИ, КУДА Я ПОТРАТИЛА ДЕНЬГИ? *)
            (* ВРОДЕ БЫ, ГУСЯ ПОКУПАЛА, СЫР...? *)
            (* ОЙ! КТО ЖЕ МНЕ ПОДСЧИТАЕТ РАСХОДЫ ? *)
            WRITELN;
            WRITELN('ФОКА, ПОМОГИ ХОЗЯЙКЕ, ПОЖАЛУЙСТА!');
            SUM:=0;
            FOR I:= 1 TO N DO
                SUM:= SUM + A[I] ;
            WRITELN('-----');
            WRITELN('РАСХОДЫ ЗА ВСЕ ДНИ =', SUM:6:2)
END.

```

```

=====
=
=   РАСХОДЫ ЗА ДЕНЬ?   4.60
=   РАСХОДЫ ЗА ДЕНЬ?   6.10
=   РАСХОДЫ ЗА ДЕНЬ?   2.95
=   РАСХОДЫ ЗА ДЕНЬ?   4.00
=
=   ФОКА, ПОМОГИ ХОЗЯЙКЕ, ПОЖАЛУЙСТА!
=   -----
=   РАСХОДЫ ЗА ВСЕ ДНИ = 17.65
=
=====

```


Задача 38. В товарищеском матче встретились две команды баскетболистов. Определить рост самого высокого игрока.

```
(*-----*)
  I   САМЫЙ ВЫСОКИЙ СПОРТСМЕН   I
(*-----*)
PROGRAM MAX(INPUT,OUTPUT);
CONST N=10;
TYPE MASSIV=ARRAY[1..N]OF INTEGER; (* ТИП *)
VAR
  SPORTSMEN:MASSIV; (* МАССИВ *)
  MAX:INTEGER;      (* МАКСИМАЛЬНЫЙ РОСТ *)
  I:INTEGER;        (* ПАРАМЕТР ЦИКЛА *)
BEGIN
  (* ВВОД ЗНАЧЕНИЙ РОСТА СПОРТСМЕНОВ *)
  (*-----*)
  WRITELN('УКАЖИТЕ РОСТ СПОРТСМЕНОВ: ');
  FOR I:=1 TO N DO
    READ(SPORTSMEN[I]);

    (* ОПРЕДЕЛЕНИЕ САМОГО ВЫСОКОГО СПОРТСМЕНА *)
    (*-----*)
    MAX:=SPORTSMEN[1];
    FOR I:=2 TO N DO
      IF SPORTSMEN[I] > MAX THEN MAX:=SPORTSMEN[I];
    WRITELN;
    WRITE('РОСТ САМОГО ВЫСОКОГО СПОРТСМЕНА =', MAX:4)
  END.
  =====
  =
  =   УКАЖИТЕ РОСТ СПОРТСМЕНОВ:   =
  =   190 185 202 216 195 182 204 175 200 180   =
  =   РОСТ САМОГО ВЫСОКОГО СПОРТСМЕНА = 216   =
  =
  =====
```

В программе используются два независимых цикла: первый предназначен для ввода значений массива, второй — для определения максимального значения.

Сначала первый элемент массива обозначается именем MAX. Затем каждый последующий элемент сравнивается со значением MAX, и если он окажется больше, то получает имя MAX.

Упакованные массивы. Как правило, одно целое число или один символ занимают в памяти ЭВМ два байта. В то же время для изображения символа достаточно одного байта. С целью экономии памяти машины при использовании символьных данных в языке Паскаль введено понятие упакованного массива. Элементы упакованного массива хранятся по два в двух байтах памяти ЭВМ.

Упакованный массив символьных данных можно описывать в разделе переменных или с использованием раздела типов TYPE. Рассмотрим более универсальный подход — описание в разделе типов:

```
TYPE _ имя типа = PACKED _ ARRAY[t1] OF _ CHAR;
VAR _ имя переменной: имя типа;
```

Здесь служебное слово PACKED указывает на то, что массив данных является упакованным; t1 — тип индекса.

Пусть, например, имеется строка символов:

"У ЕГОРКИ ВСЕГДА ОТГОВОРКИ"

Эту строку можно считать массивом, состоящим из 25 символов, включая пробелы (кавычки не входят в строку). Если этот массив символов обозначить именем СТРОКА, то описание примет вид:

```
TYPE T = PACKED ARRAY[1..25] OF CHAR ;
VAR СТРОКА: T ;
```

Здесь сначала введен тип массива с именем Т, а затем в разделе переменных указывается, что СТРОКА имеет тип Т.

Один элемент массива принимает значение одного символа, например

```
СТРОКА[1]='У', СТРОКА[2]='_', . . . ,
СТРОКА[25]='И'.
```

Элементы упакованного массива используются в программе точно так же, как и элементы неупакованного массива. Только память машины при этом автоматически выделяется меньше.

Считается, что символьные константы (см. действие 3) имеют тип упакованных массивов

```
PACKED ARRAY [1 . . N] OF CHAR;
```

где N — длина строки.

Символьные массивы и константы могут участвовать в операциях присваивания и сравнения. Пусть, например, описание символьного массива имеет вид

```
TYPE MASSIV = PACKED ARRAY[1..46] OF CHAR ;
VAR A: MASSIV ;
```

тогда допустим оператор

A:=НАДО КОЛПАК ПЕРЕКОЛПАКОВАТЬ. ПЕРЕВЫКОЛПАКОВАТЬ'

Элементы массива могут принимать свои значения с помощью оператора ввода READ, который располагается внутри цикла.

Примеры ввода:

```
1)      I:= 1
        READLN
        WHILE NOT EOLN DO
          BEGIN
            READ(A[I]) ;
            I:= I+1
          END
```

```
2)      FOR I:= 1 TO 46 DO
          READ(A[I]) ;
```

В первом случае элементы массива принимают значения до тех пор, пока не встретится символ конца строки "BK". Во втором случае цикл выполняется 46 раз.

В обоих случаях элементы массива принимают следующие значения:

```
A[1]='Н', A[2]='А', . . . , A[46]='Б'.
```


Для машин типа ДВК перед вводом символьных данных необходимо указывать оператор без параметров READLN.

Вывод массива символьных данных также организуется с использованием массива. Так, для вывода значений массива А рассмотренного примера, можно составить следующую программу:

З а м е ч а н и е. Во многих версиях языка Паскаль при описании символьных данных можно не указывать слово PACKED, так как оно подразумевается по умолчанию. Кроме того, допускается ввод и вывод одномерных массивов без организации циклов. В этом случае в операторах ввода и вывода представлены имена массивов без индекса, например READ(A) или WRITE(A).

Задача 39. Дана строка символов, обозначенная именем S1:

"КАЖДЫЙ ОХОТНИК ЖЕЛАЕТ ЗНАТЬ, ГДЕ СИДИТ ФАЗАН"

Требуется сформировать новый массив с именем S2, который содержит представленную строку символов с добавлением в конце вопросительного и восклицательного знаков.

```
(*-----*)
!           ОХОТНИК И ФАЗАН           !
-----*)
PROGRAM A33A(INPUT,OUTPUT);
TYPE STROKA = PACKED ARRAY[1..50] OF CHAR;
VAR
  S1,S2:STROKA ;    (* СТАРАЯ И НОВАЯ СТРОКИ *)
  I : INTEGER;      (* СЧЕТЧИК ВСЕХ СИМВОЛОВ *)
  K : INTEGER;      (* ПАРАМЕТР ЦИКЛА      *)
BEGIN
  (* ВВОД СИМВОЛОВ СТРОКИ
  -----*)
  WRITELN('ВВЕДИТЕ СТРОКУ S1:');
  I:= 0 ;
  READLN;
  WHILE NOT EOLN DO
    BEGIN
      I:= I + 1 ;
      READ( S1[I] )
    END;
  S2:=S1;
  S2[I+1]:='?';
  S2[I+2]:='!';
  (* ВЫВОД НОВОЙ СТРОКИ
  -----*)
  WRITELN('НОВАЯ СТРОКА S2:');
  FOR K:= 1 TO I+2 DO
    WRITE(S2[K])
  END.
=====
=
= ВВЕДИТЕ СТРОКУ S1:
= КАЖДЫЙ ОХОТНИК ЖЕЛАЕТ ЗНАТЬ, ГДЕ СИДИТ ФАЗАН
= НОВАЯ СТРОКА S2:
= КАЖДЫЙ ОХОТНИК ЖЕЛАЕТ ЗНАТЬ, ГДЕ СИДИТ ФАЗАН?!
=
=====
```

Задача 40. Два дружка WIK и RIK захотели поиграть в "казаков-разбойников", но никак не могли решить, кому быть казаком, а кому — разбойником.

Проходящий мимо Дока помог им: сначала WIK будет разбойником, а RIK казаком, а потом они поменяются ролями:

```
(*-----*)
I      KTO  EST'  KTO ?      I
(*-----*)
PROGRAM PT3(INPUT,OUTPUT);
CONST N=10;
TYPE MAS=ARRAY[1..N]OF CHAR;
VAR WIK,RIK,X: MAS;
BEGIN
  WRITE('WIK - ЭТО КТО? ');
  READ(WIK);
  WRITE('RIK - ЭТО КТО? ');
  READ(RIK);
  WRITELN('ПРОИСХОДИТ ЗАМЕНА:');
  X:=WIK;
  WIK:=RIK;
  RIK:=X;
  WRITELN('WIK - ', WIK);
  WRITELN('RIK - ', RIK)
END.

=====
=      WIK - ЭТО КТО?  РАЗБОЙНИК      =
=      RIK - ЭТО КТО?  КАЗАК          =
=      ПРОИСХОДИТ ЗАМЕНА:              =
=      WIK - ЭТО КАЗАК                  =
=      RIK - ЭТО РАЗБОЙНИК              =
=====
```

В программе используются два символьных массива WIK и RIK. Для того чтобы можно было поменяться ролями, выделен третий массив X. Поскольку значения массивов имеют разную длину ("казак" — 5 символов, "разбойник" — 9), то тип массива MAS должен иметь описание не меньше десяти символов.

Можно ли выполнить замену ролей, не используя промежуточного массива X? Что получится, если замену произвести с помощью двух операторов:

WIK:=RIK; RIK:=WIK;

Следует обратить внимание на то, что для ввода и вывода символьных массивов не применяется цикл.

Понятие многомерных массивов. До сих пор мы рассматривали массивы, каждый элемент которых содержал только один индекс. Такие массивы обычно называются одномерными. В математике часто используются многомерные массивы, т.е. массивы массивов. Особенно широкое распространение получили двумерные массивы, иначе называемые матрицами. Так, например, изображение целых чисел последовательно в нескольких строках является матрицей:

```
15, 4, 3, 6
2, 8, 1, 7
4, 3, 19, 5
```

Данная матрица имеет размер 3 на 4, т.е. матрица состоит из трех строк и четырех столбцов. Если всю матрицу обозначить одним именем, например A, то каждый элемент матрицы обозначается с двумя индексами, например A[I,J]. Здесь первый индекс I обозначает номер строки (I = 1, 2, 3), второй индекс J — номер столбца (J = 1, 2, 3, 4). Такую матрицу можно описать следующим образом (с использованием имени типа T):

- 1) TYPE T = ARRAY [1..3, 1..4] OF INTEGER;
 VAR A: T;
- 2) TYPE T = ARRAY [1..3] OF ARRAY [1..4] OF INTEGER;
 VAR A: T;

В первом случае описывается каждый тип индекса, затем указывается простой базовый тип элементов массива INTEGER. Во втором случае сначала описывается один тип индекса 1 .. 3, затем указывается сложный базовый тип

ARRAY [1 .. 4] OF INTEGER;

который в свою очередь содержит описание второго типа индекса и простого базового типа INTEGER.

Если в программе необходимо выделять отдельные строки матрицы, то удобно ввести такое описание

```

TYPE
    T1 = ARRAY [1..4] OF INTEGER;
    T = ARRAY [1..3] OF T1;
VAR
    A: T ;
    B: T1;
```

Здесь сначала описывается тип одной строки T1, а затем, через тип строки T1, описывается тип всей матрицы T. В разделе переменных указывается, что A является двумерным массивом (т.е. матрицей), а B — одномерным массивом.

При описании многомерных массивов типы индексов могут быть различными. Это позволяет формулировать данные в более естественном виде, не прибегая к сложной символике.

Сделаем, например, описание шахматной доски. Поскольку шахматная доска обозначается как латинскими буквами, так и цифрами, введем следующее описание:

```

TYPE  БУКВА = (A, B, C, D, E, F, G, H);
TYPE  ШАХМДОСКА = ARRAY [ БУКВА, 1..8 ] OF ФИГУРА;
```



Тип ФИГУРА можно задать перечислением всех белых и черных фигур, при этом пустую клетку обозначить каким-либо именем, например ПУСТО:

```

TYPE  ФИГУРА = (ПБ, ЛБ, КБ, СБ, ФБ, КРБ,
                ПЧ, ЛЧ, КЧ, СЧ, ФЧ, КРЧ, ПУСТО);
```

Такое описание шахматной доски является более естественным, чем описание

TYPE ШАХМДОСКА = [1..8, 1..8] OF ФИГУРА;

где для латинских букв необходимо вводить дополнительные признаки, которые тоже должны обрабатываться в программе.

Задача 41. Дана матрица действительных чисел {B}

1.2	1.5	1.3	1.4	1.2
1.6	1.3	1.4	1.7	1.5
1.5	1.9	1.6	1.8	1.4

Получить матрицу {R}, значения которой равны удвоенным значениям матрицы {B}.

```
(*-----*)
I          МАТРИЦЫ          I
(*-----*)
PROGRAM MATRY(INPUT,OUTPUT);
  CONST N=3;                (* КОЛИЧЕСТВО СТРОК      *)
        M=5;                (* КОЛИЧЕСТВО СТОЛБЦОВ *)
  TYPE MAS=ARRAY[1..N,1..M]OF REAL;
  VAR
    B,R : MAS;              (* МАССИВЫ ТИПА MAS    *)
    I : INTEGER;            (* ИНДЕКС СТРОКИ       *)
    J : INTEGER;            (* ИНДЕКС СТОЛБЦА      *)
  BEGIN
    (* ВВОД ЗНАЧЕНИЙ МАТРИЦЫ "B"
       -----*)
    WRITELN('ВВЕДИТЕ ЗНАЧЕНИЯ МАТРИЦЫ "B" : ');
    FOR I:=1 TO N DO
      FOR J:=1 TO M DO
        READ(B[I,J]);

    (* ВЫЧИСЛЕНИЕ ЗНАЧЕНИЙ МАТРИЦЫ "R"
       -----*)
    FOR I:=1 TO N DO
      FOR J:=1 TO M DO
        R[I,J]:=B[I,J]*2;

    (* ВЫВОД ЗНАЧЕНИЙ МАТРИЦЫ "R"
       -----*)
    WRITELN;
    WRITELN('ЗНАЧЕНИЯ МАТРИЦЫ "R" : ');
    FOR I:=1 TO N DO
      BEGIN
        FOR J:=1 TO M DO
          WRITE(R[I,J]:3:1, ' ':2);
        WRITELN;
      END;
  END.
```

```
=====
=
=      ВВЕДИТЕ ЗНАЧЕНИЯ МАТРИЦЫ "B" :      =
=      1.2   1.5   1.3   1.4   1.2          =
=      1.6   1.3   1.4   1.7   1.5          =
=      1.5   1.9   1.6   1.8   1.4          =
=
=      ЗНАЧЕНИЯ МАТРИЦЫ "R" :              =
=      2.4   3.0   2.6   2.8   2.4          =
=      3.2   2.6   2.8   3.4   3.0          =
=      3.0   3.8   3.2   3.6   2.8          =
=====
```

В программе представлены все необходимые пояснения: данные, их описание и использование.

Картина 2. МНОЖЕСТВО — ЭТО ВСЕГДА МНОГО?

Дока. В математике под множеством понимается некоторый набор элементов. Например, множество фигур на плоскости

{круг, ромб, квадрат, треугольник, прямоугольник}

Все элементы одного множества различны и не упорядочены. Элементы множества не могут повторяться, например неверная запись множества {круг, ромб, круг}.

Фока. Можно ли в таком случае считать, что два множества

{круг, ромб, квадрат} и {ромб, круг, квадрат}

одинаковые и равны между собой?

Дока. Совершенно верно! Особенность множеств заключается в том, что к ним применимы специальные операции. Рассмотрим их на примере. Пусть А есть множество цветов {пион, астра, роза}, В — множество {ромашка, пион}. Тогда допустимы следующие операции.

Объединение множеств $A \cup B$. Каждый элемент нового множества является элементом или множества А или В:

{пион, астра, роза, ромашка}

Пересечение множеств $A \cap B$. Каждый элемент нового множества является одновременно элементом множества А и множества В: {пион}.

Разность множеств $A \setminus B$. Каждый элемент нового множества является элементом множества А, но не является элементом В: {роза, астра}.

Множество в языке Паскаль — это ограниченный, неупорядоченный набор различных элементов одинакового типа.

В отличие от математики элементы множества задаются не в фигурных скобках, а в квадратных, например:

[АСТРА, РОЗА] или [25,6,14,18].

Множество может не содержать ни одного элемента. В этом случае оно называется пустым, например [].

Во многих версиях языка Паскаль элементы множества нельзя вводить с помощью оператора ввода READ, но значения множеств можно задавать в операторе присваивания, например:

C1:=[АСТРА, РОЗА]; C2:=[25,6,14,18].

Если множества используются в программе, то они должны быть описаны либо с помощью раздела TYPE, либо непосредственно в разделе переменных.

Описание множества в разделе типов имеет вид

```
TYPE _ имя типа = SET _ OF _ t;  
VAR _ имя множества : имя типа;
```

Здесь t — базовый тип элементов множества. В качестве базового может быть любой простой тип, кроме действительного.

Пример описания:

```
TYPE M = SET OF (ПИОН, АСТРА, РОЗА, РОМАШКА);  
VAR A,B:M;
```



Здесь задан тип множества M. В разделе переменных указано, что переменные A и B имеют тип M, т.е. могут принимать значения любых из перечисленных цветов, например:

A:= [ПИОН, АСТРА, РОЗА];

B:= [РОМАШКА, ПИОН];

Другой пример описания:

```
TYPE MNOG = SET OF 1980..2000;
VAR M1, M2: MNOG;
```

В программе элементами множеств M1, M2 и M3 могут быть любые целые числа от 1980 до 2000, например:

M1:= [1988, 1995, 1981];

M2:= [1980, 1981, 1982, 1983];

M3:= [2000];

Приведем примеры описания множеств непосредственно в разделе переменных:

```
VAR M1, M2: SET OF 1980..2000;
```

```
VAR MS: SET OF CHAR;
```

В последнем случае элементами множества MS являются символьные константы, например:

MS:= ['A', 'N', 'R']

Количество элементов, входящих в множество, может быть ограничено. Оно зависит от типа компьютера и версии языка Паскаль.

В языке Паскаль имеются следующие операции над множествами:

+ объединение множеств;

* пересечение множеств;

— вычитание множеств;

=, <> проверка множеств на равенство, неравенство.

Множество A равно множеству B, если каждый элемент множества A является элементом множества B, и наоборот, каждый элемент множества B является элементом множества A. Иначе множества A и B не равны друг другу. Результат операции будет логического типа, TRUE (истина) или FALSE (ложь);

<= проверка множества на включение. Множество A включено в множество B, если все элементы множества A являются также элементами множества B. Результат операции $A \leq B$ логический, TRUE (истина) или FALSE (ложь);

IN проверка на принадлежность какого-либо значения множеству. Результат операции — логический, TRUE или FALSE. Операция $S \text{ IN } A$ служит для проверки, принадлежит ли элемент базового типа S множеству A.

Задача 42. МЕШАНИНА: если взять то общее, что есть у боба с ложкой, добавить кота и поместить в теплое место, то получится муравей.

Так ли это?

Состоит ли муравей из кота?



Пусть Y1, Y2, Y3, Y4 — заданные множества символьного типа. Они получают значения с помощью операторов присваивания, например Y1:= ['B', 'E', 'A', 'N']. Применяя соответствующие операции над множествами, формируем новое множество

$$X := (Y1 * Y2) + Y3 - Y4.$$

Чтобы посмотреть содержимое этого множества, выведем его значения на экран дисплея.

```
(*-----*)
!      БОБ * ЛОЖКА + КОТ - ХОЛОД = МУРАВЕЙ ?      !
!-----*)
PROGRAM A28A(INPUT,OUTPUT );
VAR
  Y1,Y2,Y3,Y4,X : SET OF CHAR; (* МНОЖЕСТВА *)
  S : CHAR;        (* СИМВОЛ      *)
BEGIN
```

```
  Y1:=['B', 'E', 'A', 'N' ] ;
  Y2:=['S', 'P', 'O', 'O', 'N' ] ;
  Y3:=['C', 'A', 'T' ] ;
  Y4:=['C', 'O', 'L', 'D' ];
```

```
(* ФОРМИРОВАНИЕ И ПЕЧАТЬ МНОЖЕСТВА'X'
-----*)
```

```
X:= (Y1 * Y2) + Y3 - Y4 ;
WRITE( ' МНОЖЕСТВО X = ' );
FOR S:= 'A' TO 'Z' DO
  IF S IN X THEN WRITE(S) ;
WRITELN;
```

```
(* ПРОВЕРКА ВКЛЮЧЕНИЯ МНОЖЕСТВА'Y3' В 'X'
-----*)
```

```
IF Y3 <= X
  THEN WRITE(' МУРАВЕЙ СОСТОИТ ИЗ КОТА ')
  ELSE WRITE(' МУРАВЕЙ НЕ СОСТОИТ ИЗ КОТА' )
```

```
END.
```

```
=====
=
=      МНОЖЕСТВО X = ANT
=      МУРАВЕЙ НЕ СОСТОИТ ИЗ КОТА
=
=====
```

Для вывода значений нового множества X воспользуемся оператором цикла FOR. Параметром цикла является символьная переменная S, которая принимает значение каждого символа латинского алфавита от 'A' до 'Z'. Если значение переменной S принадлежит множеству X, то оно выводится на экран дисплея.

Для того чтобы решить вторую часть задачи "Состоит ли муравей из кота?", используется операция включения <= в условном операторе.

Задача 43. Из множества целых чисел [2..20] выделить следующие множества:

делящихся на 6 без остатка;

делящихся без остатка или на 2 или на 3.

Введем обозначения:

N2 — множество чисел, делящихся на 2;

Y1 BEAN — БОБ

Y2 SPOON — ЛОЖКА



Y3 CAT — КОТ

Y4 COLD — ХОЛОД

(ПОМЕСТИТЬ В ТЕПЛО — ЗНАЧИТ УБРАТЬ ХОЛОД)

X ANT —

МУРАВЕЙ

N3 — множество чисел, делящихся на 3;
 N6 — множество чисел, делящихся на 6;
 N23 — множество чисел, делящихся на 2 или на 3.

Программа имеет вид.

```
(*-----*)
!                МНОЖЕСТВО КРАТНЫХ ЧИСЕЛ                !
(*-----*)
PROGRAM A23(INPUT,OUTPUT);
  CONST  N=20;      (* РАЗМЕРНОСТЬ МНОЖЕСТВА *)
  VAR
    N2,N3,N6,N23 : SET OF INTEGER;
    K : INTEGER;
BEGIN
  N2:=[];           (* НАЧАЛЬНОЕ ЗНАЧЕНИЕ N2 *)
  N3:=[];           (* НАЧАЛЬНОЕ ЗНАЧЕНИЕ N3 *)
  FOR K:=1 TO N DO
    BEGIN
      IF K MOD 2 = 0 THEN N2:=N2+[K];
      IF K MOD 3 = 0 THEN N3:=N3+[K]
    END;
    N6 := N2*N3 ;
    N23:= N2+N3 ;
    WRITELN(' НА 6 ДЕЛЯТСЯ ЧИСЛА : ');
    FOR K:=1 TO N DO
      IF K IN N6 THEN WRITE(K:3);
    WRITELN;
    WRITELN(' НА 2 ИЛИ НА 3 ДЕЛЯТСЯ ЧИСЛА : ');
    FOR K:=1 TO N DO
      IF K IN N23 THEN WRITE(K:3)
    END;
  END;

=====
=
=  НА 6 ДЕЛЯТСЯ ЧИСЛА :
=   6 12 18
=
=  НА 2 ИЛИ НА 3 ДЕЛЯТСЯ ЧИСЛА :
=   2 3 4 6 8 9 10 12 14 15 16 18 20
=
=====
```

Картина 3. ЕСЛИ БЫ Я СТАЛ ДИРЕКТОРОМ

Дока. Что ты сделал бы, Фока, в первую очередь, если бы стал директором?

Фока. Во-первых, я бы сразу приобрел портфель. Во-вторых, стал бы ходить медленно и говорить солидно. В-третьих, весь премиальный фонд я бы раздал лучшим сотрудникам. Например, большую премию в зависимости от стажа работы получили бы следующие мои коллеги:

№ п/п	Фамилия И.О.	Руб., коп.
1	Демина Л.Г.	1111.11
2	Тверитин Ю.А.	1313.13
3	Ковалева Л.А.	1000.01
4	Терещенко С.М.	999.99

Дока. Вот мы и подошли к новому понятию в программировании. Во многих экономических и информационных задачах обрабатываются ведомости, документы, каталоги, списки. При этом появляется необходимость объединять данные различного типа в одну группу. Так, каждая строка рассмотренной ведомости представлена различными данными:

порядковый номер — целое положительное число;
 фамилия, имя, отчество — массив символов;
 премия — действительное число.

Для работы с группой данных различного типа введено понятие записи.

В языке Паскаль запись представляет собой совокупность ограниченного числа данных различного типа.

П р и м е р ы записей.

Данные о студенте:

Фамилия — массив символов,
 имя — массив символов,
 год рождения — целое число,
 название института — массив символов.

Формуляр книги в библиотеке:

Автор — массив символов,
 название — массив символов,
 год издания — целое число,
 издательство — массив символов,
 цена — действительное число.

Пусть имеется два списка учащихся с их оценками по программированию:

с п и с о к 1

с п и с о к 2

№ п/п	Фамилия, имя	Оценка
1	Фортранов Форт	5 4 5 4
2	Бейсикова Беся	4 5 4 5
3	Фокалов Фок	4 4 5 4

№ п/п	Фамилия, имя	Оценка
1	Паскалев Пас	5 5 5 5
2	Адова Ада	5 4 5 5
3	Прологов Прол	5 5 5 5
4	Модулова Мода	5 4 4 5

Здесь представлены данные следующих типов:

порядковый номер — целое десятичное число,
 фамилия, имя — массив символов,
 оценки — массив целых чисел.

Все эти данные можно объединить в одну группу и считать записью. Запись в целом и отдельные ее элементы обозначаются именами. Введем, например, следующие обозначения: C1 — имя записи первого списка, C2 — имя записи второго списка, N — порядковый номер, ФАМ — фамилия, имя, ОЦЕН — оценки. К каждому элементу записи можно обратиться с помощью уточненного имени. Оно содержит имя записи, а через точку — имя элемента, например:

C1.N	C2.N
C1.ФАМ	C2.ФАМ
C1.ОЦЕН	C2.ОЦЕН

Записи, как и другие данные, объявляются в разделе описаний и используются в разделе операторов.



Описание записи имеет следующий вид:

```
TYPE _ имя типа = RECORD
    имя элемента 1 : тип;
    имя элемента 2 : тип;
    ...
    имя элемента n-1: тип;
    имя элемента n : тип
END;
VAR _ имя записи: имя типа;
```

Здесь служебное слово RECORD (запись) выполняет роль открывающей операторной скобки, END — закрывающей операторной скобки. Внутри операторных скобок описываются элементы записи. Допускается вместо имени записи указывать список имен, т.е. имена записей, разделенные запятыми.

Элементы записи вместе с их описанием называются полями записи.

Списки 1 и 2 можно описать следующим образом:

```
TYPE T = RECORD
    N: INTEGER;
    ФАМ: ARRAY[1..20]OF CHAR;
    ОЦЕН: ARRAY[1..4]OF INTEGER
END;
VAR C1,C2: T;
```

Здесь сначала введен тип записи T, а затем в разделе переменных указано, что переменные C1 и C2 имеют тип T.

Каждый элемент записи имеет свое описание: N — переменная целого типа; ФАМ — массив из 20 символов (если фамилия и имя содержат меньше 20 символов, то оставшиеся позиции заполняются пробелами); ОЦЕН — массив из четырех целых чисел.

Допускается описание записи непосредственно в разделе переменных:

```
VAR имя записи : RECORD
    имя элемента 1 : тип;
    имя элемента 2 : тип;
    ...
    имя элемента n-1: тип;
    имя элемента n : тип
END;
```

Элемент записи используется в программе в том же самом смысле, как и обычная переменная. Таким образом, элемент записи можно указывать как в левой части оператора присваивания, так и в выражениях. Над элементом записи можно выполнять все те действия, которые допустимы для данных его типа. Например, если тип элемента записи — целый, то можно выполнять все операции, допустимые для целых данных. Так, для рассмотренной ведомости над элементами записи можно, например, выполнить следующие операции:

а) ввести значения порядковых номеров

```
READ (C1.N); READ (C2.N)
```

б) вычислить сумму первых двух оценок

```
СУММА1 := C1.ОЦЕН[1] + C1.ОЦЕН[2];
СУММА2 := C2.ОЦЕН[1] + C2.ОЦЕН[2];
```


Обращение к записи в целом, а не только к ее элементам, допускается лишь в операторе присваивания. Слева и справа от знака присваивания при этом должны использоваться имена записей одинакового типа.

Оператор присоединения. Мы отметили, что обращение к элементам записи происходит с помощью уточненного имени. Оператор присоединения позволяет упростить обращение к элементу записи. Имя записи выносится в заголовок оператора присоединения, а в блоке BEGIN—END используются только имена элементов записи.

Оператор присоединения имеет вид:

```
WITH — имя записи — DO
  BEGIN
    операторы
  END
```

Например, для первого списка оператор присоединения можно записать следующим образом:

```
WITH C1 DO
  BEGIN
    READ(N);
    СУММА:= ОЧЕН[1] + ОЧЕН[2]
  END
```

Задача 44. Имеется ведомость учащихся с оценками

Колбаскин Коля	4
Булах Митя	5
Булах Рома	5
Жуков Сережа	5
Бубликова Буля	4
Мазеин Дима	5

Составить программу для вычисления среднего балла. Поскольку в каждой строке ведомости представлены данные различного типа, введем запись с именем СПИСОК, состоящую из двух полей: ФАМ - фамилия и имя — упорядоченный массив из 15 символов, БАЛЛ - оценка — целое число.

Для вычисления среднего балла СРБАЛЛ находится сумма всех оценок СУММА, которая делится на число учащихся N.

Описание записи сделано в разделе TYPE, где T — имя типа. В разделе переменных VAR указано, что переменная СПИСОК имеет тип T.

Раздел операторов состоит из двух циклов: один вложен в другой. Внешний цикл с параметром I выполняется столько раз, сколько имеется фамилий в ведомости (в данном случае N = 6). Внутренний цикл с параметром K предназначен для ввода фамилии и имени. На фамилию и имя отведено поле из 15 позиций. Лишние позиции поля заполняются пробелами. Внутри поля фамилию и имя можно располагать свободно, например

```
КОЛБАСКИН — КОЛЯ —
или КОЛБАСКИН — — КОЛЯ
или — КОЛБАСКИН — КОЛЯ
```


После имени следует оценка. Поскольку оценка является числом, а не символом, то перед ним игнорируются все пробелы. Это означает, что оценку можно вводить в любой позиции, начиная с 16-й.

В машинах типа ДВК перед вводом символьных данных ставится пустой оператор READLN. В некоторых версиях языка Паскаль символьный массив можно вводить без организации цикла, а просто с указанием имени массива в операторе ввода READ.

Программа имеет вид:

```

!   ОБРАБОТКА ВЕДОМОСТИ С ИСПОЛЬЗОВАНИЕМ ЗАПИСИ   !
-----*)
PROGRAM WEDMST(INPUT,OUTPUT);
CONST   N=6;                      (* ЧИСЛО УЧАЩИХСЯ В СПИСКЕ *)
        M=15;                     (* МАКСИМАЛЬНАЯ ДЛИНА ФАМ *)
TYPE    T = RECORD
        ФАМ : PACKED ARRAY[1..M] OF CHAR;
        БАЛЛ: INTEGER
END;
VAR      СПИСОК : T;
        I,K    : INTEGER; (* ПАРАМЕТРЫ ЦИКЛА * )
        СРБАЛЛ : REAL;    (* СРЕДНИЙ БАЛЛ * )
        СУММА  : INTEGER; (* СУММА ОЦЕНОК * )
BEGIN
  СУММА:=0;
  WRITELN('ВВЕДИТЕ ТАБЛИЦУ: ФАМИЛИЯ,ИМЯ,ОЦЕНКА');
  FOR I:=1 TO N DO
    BEGIN
      READLN;
      FOR K:=1 TO M DO READ (СПИСОК.ФАМ[K]);
      READ(СПИСОК.БАЛЛ);
      СУММА:=СУММА+СПИСОК.БАЛЛ;
    END;
  СРБАЛЛ:=СУММА/N;
  WRITELN('-----');
  WRITELN('СРЕДНИЙ БАЛЛ = ', СРБАЛЛ:5:2);
END.

```

```

=====
=
=      ВВЕДИТЕ ТАБЛИЦУ:  ФАМИЛИЯ,ИМЯ,ОЦЕНКА      =
=      КОЛБАСКИН КОЛЯ      4                      =
=      БУЛАХ  МИТЯ         5                      =
=      БУЛАХ  РОМА         5                      =
=      ЖУКОВ  СЕРЕЖА       5                      =
=      БУБЛИКОВА БУЛЯ      4                      =
=      МАЗЕИН  ДИМА        5                      =
=      -----
=      СРЕДНИЙ БАЛЛ =    4.67                      =
=
=====

```

Задача 45. Присвоить звание "ДОБРЫЕ МОЛОДЦЫ" всем тем молодцам, которые лучше всех поют, играют в футбол и солят капусту.

Из исходного списка С1, в котором содержатся фамилии всех молодцов и их характеристики, нужно сформировать другой список С2, состоящий из отличившихся молодцов. Для формирования списков используется массив массивов.

Программа имеет вид:

```
PROGRAM KAPUS(INPUT,OUTPUT);
CONST K=5;
TYPE TIP= RECORD
    ФИО: ARRAY[1..K,1..16]OF CHAR;
    ПЕВЕЧ: ARRAY[1..K]OF INTEGER;
    ФУТБОЛИСТ: ARRAY[1..K]OF INTEGER;
    КАПУСТА: ARRAY[1..K]OF INTEGER;
END;
VAR C1,C2: TIP;      (* СПИСКИ *)
    I,J,N: INTEGER;  (* ПАРАМЕТРЫ ЦИКЛОВ *)
BEGIN
    J:=0;
    WRITELN('ВВЕДИТЕ ФИМИЛИЮ И.О. И ПРИЗНАКИ:');
    FOR I:=1 TO K DO
    BEGIN
        READLN;
        FOR N:=1 TO 15 DO READ(C1.ФИО[I,N]);
        READ(C1.ПЕВЕЧ[I],C1.ФУТБОЛИСТ[I],C1.КАПУСТА[I]);
        IF (C1.ПЕВЕЧ[I]=5)AND
            (C1.ФУТБОЛИСТ[I]=5)AND(C1.КАПУСТА[I]=5)
            THEN BEGIN
                J:=J+1; C2.ФИО[J]:=C1.ФИО[I]
            END;
    END;
    IF J=0 THEN WRITELN('ТАКИХ МОЛОДЦОВ НЕТ')
    ELSE BEGIN
        WRITELN('-----');
        WRITELN('АЙ, ДА МОЛОДЕЦ !');
        FOR I:=1 TO J DO
            WRITELN(C2.ФИО[I]);
        END;
    END.
END.
```

```
=====
=
= ВВЕДИТЕ ФИМИЛИЮ И.О. И ПРИЗНАКИ: =
= МОРОКОВКИН М.М. 2 4 5 =
= РЕДИСКИН Р.Р. 5 4 3 =
= ЛОСЕВ Ю.А. 5 5 5 =
= ЛУКОВКИН Л.Л. 4 5 5 =
= КОЛДАЕВ В.Д. 5 5 5 =
= ----- =
= АЙ, ДА МОЛОДЕЦ ! =
= ЛОСЕВ Ю.А. =
= КОЛДАЕВ В.Д. =
=====
```



Задача 46. Дан многочлен

$$4A+5B-8C+16A-3A+9K-1A$$

Найти подобные члены для переменной A и вычислить суммарный коэффициент.

Один элемент многочлена можно считать записью, так как он состоит из данных различного типа — коэффициента (чисел) и буквы.

Введем обозначения: M1 — элемент многочлена (запись); M2 — результирующий элемент (запись); COEF — коэффициент (элемент записи); BUK — буква (элемент записи); ELEM — тип записи; SUM — сумма коэффициентов.

Программа имеет вид:

```

(*-----*)
I      ПОДОБНЫЕ ЧЛЕНЫ МНОГОЧЛЕНА      I
-----*)
PROGRAM EX4(INPUT,OUTPUT);
TYPE
  ELEM=RECORD
    COEF :INTEGER; (* КОЭФФИЦИЕНТ *)
    BUKWA :CHAR;   (* БУКВА      *)
  END;
VAR
  M1: ELEM; (* ВХОДНОЙ ЭЛЕМЕНТ *)
  M2: ELEM; (* ВЫХОДНОЙ ЭЛЕМЕНТ *)
  SUM: INTEGER; (* СУММА КОЭФФИЦИЕНТОВ *)
BEGIN
  SUM:=0;
  READLN;
  WRITELN('ВВЕДИТЕ МНОГОЧЛЕН :');
  WHILE NOT EOLN DO
    BEGIN
      READ(M1.COEF,M1.BUKWA);
      IF M1.BUKWA='A' THEN SUM:=SUM+M1.COEF
      END;
      M2.COEF:= SUM ;
      M2.BUKWA:= 'A';
      WRITELN('ПОДОБНЫЙ ЧЛЕН=',M2.COEF:3,M2.BUKWA)
    END.
  =====
  =
  =      ВВЕДИТЕ МНОГОЧЛЕН :      =
  =      4A+7P-3A+8A-2K+1P-5R-2B  =
  =      ПОДОБНЫЙ ЧЛЕН=  9A      =
  =====

```

Картина 4. ФАЙЛ — ЛЮБИМОЕ СЛОВО ПРОГРАММИСТА

Фока. Мы все время рассматриваем задачи, в которых во время выполнения программы данные поступают с клавиатуры, а результаты выводятся на экран дисплея. Поэтому ни исходные данные, ни результаты не сохраняются. Всякий раз при выполнении одной и той же программы, особенно во время отладки ее, приходится заново вводить исходные данные. А если их очень много? Но особенно мне жалко результаты: на экране дисплея они есть, а сохранить их не удастся. Вот было бы здорово, если бы удалось записать их на диск!

Дока. Такая возможность имеется. Для этого необходимо оформить исходные данные и результаты в виде файлов, которые хранятся на диске точно так же, как и программа. Под файлом понимают любой набор данных: исходные данные, программы, результаты, любые тексты.

Файл в языке Паскаль представляет собой последовательность элементов одного типа. В отличие от массива длина файла (т.е. число элементов) не задается, место элемента не определяется индексом. И если каждый элемент файла становится доступным только после перебора всех предыдущих элементов, то файл называется последовательным.

Необходимо заметить, что создание и обработка файлов зависят как от конкретной реализации типа компьютера, так и от версии языка Паскаль.

Данные могут храниться на диске как символные или во внутреннем коде машины. Поскольку с помощью экранного редактора текста см.раздел "Зри в корень") можно создавать, просматривать и корректировать символные данные, то мы их и рассмотрим.

Файл, хранимый на диске, часто называют внешним. Во время выполнения программы данные файла поступают в память ЭВМ и преобразуются в тот тип данных, который объявлен в программе.

Если программа взаимодействует с внешними файлами, то файлы должны быть описаны в программе либо явно в разделе переменных VAR, либо с использованием раздела типов TYPE.

Объявление файлов в разделе переменных имеет вид:

VAR *имя файла*: FILE OF *базовый тип*;

Пример:

```
VAR
  FT : FILE OF CHAR;      (* ФАЙЛ СИМВОЛЬНЫХ ДАННЫХ *)
  FINP : FILE OF REAL;    (* ФАЙЛ ДЕЙСТВИТЕЛЬНЫХ ДАННЫХ *)
  M : FILE OF INTEGER;    (* ФАЙЛ ЦЕЛЫХ ДАННЫХ *)
```

Файлы символных данных называются текстовыми файлами. Описание текстового файла FILE OF CHAR эквивалентно описанию TEXT. Поэтому в предыдущем примере текстовый файл FT можно объявить в виде

VAR *FT*: TEXT

Как было отмечено ранее, определение данных с использованием раздела описания типов TYPE делает программу более универсальной и упрощает отладку программы.

Объявление файлов в этом случае имеет вид

TYPE *имя типа* = FILE OF *базовый тип*;
VAR *имя переменной* : *имя типа*;

Например, для рассмотренного текстового файла FT описание такое:

```
TYPE T = FILE OF TEXT;
VAR FT : T;
```

Здесь T — введенное нами имя типа.

Рассмотрим основные три вида взаимодействия программы с внешними файлами: чтение, запись, чтение и запись файлов.

Чтение файла. Под чтением файла понимается ввод данных из внешнего файла, находящегося на диске, в оперативную память машины. Данные внешнего файла становятся доступными программе.

Для чтения файла в программе необходимо выполнить следующие действия: открыть файл для чтения (оператор RESET), ввести данные файла в программу (оператор READ), закрыть файл для чтения (оператор CLOSE).

Внешний файл, из которого читаются данные, часто называют входным файлом. Общая форма чтения файла имеет вид:



RESET (*имя 1*);

...

READ (*имя 1*, *параметры*);

...

CLOSE (*имя 1*);

Многократно отмечено наличие других операторов программы.

В некоторых версиях микроЭВМ (в частности, ДБК) оператор RESET может иметь несколько аргументов, например:

RESET (*имя 1*, '*имя 2*', '*имя 3*', *имя 4*).

Аргументы имеют следующие назначения:

имя 1 — файловая переменная, которая в программе объявлена как файл. Она устанавливает связь с физическим (конкретным) именем файла; *имя 2* — физическое (конкретное) имя файла, который хранится на диске; *имя 3* — расширение физического имени файла. Можно указывать любые три символа, например PAS, однако рекомендуется использовать слово DAT (обозначает — данные). Аргумент можно не указывать, тогда по умолчанию будет слово DAT; *имя 4* — переменная, которая автоматически получает значение, равное числу блоков, занимаемых файлом на диске. Аргумент можно не указывать, но если он указан, то нужно объявить переменную как целого типа. Для разных типов ЭВМ один блок может иметь различное число байтов. Например, блок может состоять из 512 байтов, а один символ занимать один байт памяти.

Оператор ввода для чтения данных файла обладает всеми свойствами обычного оператора ввода READ. Если оператор ввода имеет вид READ (*параметры*), то данные вводятся с клавиатуры, а если READ (*имя 1*, *параметры*), то данные вводятся из внешнего файла, хранящегося на диске.

Параметрами являются переменные. Каждая переменная получает значение одного элемента, считанного из файла. Это значение преобразуется автоматически в тот тип данных, который имеет переменная.

Поскольку по определению число элементов файла не задается, то в языке Паскаль введен признак конца файла. Стандартная встроенная функция:

EOF (*имя 1*)

используется для определения, достигнут ли конец файла: она принимает истинное значение, если достигнут конец файла и ложное — в противном случае.

Функцию EOF можно использовать в логических выражениях и, в частности, в операторах цикла, например:

```
WHILE NOT EOF(ИМЯ 1) DO
  BEGIN
    ...
  END
```

Пока не достигнут конец файла, повторяется циклическая часть программы, заключенная в операторные скобки BEGIN—END.

Элементы в файле хранятся по строкам. Для определения конца строки файла используется оператор цикла

WHILE NOT EOLN (*имя 1*) DO

а для перехода на новую строку файла — оператор READLN (*имя 1*).

Задача 47. Пусть на диске имеется файл с именем D1.PAS, в котором записана последовательность действительных чисел в нескольких строках:

```
0.54 _ 1.7 _ 4.56 _ 0.2
1.32 _ _ _ 1.524 _ _ 78.0 _ _ 0.98
5.6 _ 7.7
```

Необходимо эти данные сделать доступными программе и вычислить их сумму. Программа имеет вид:

```
PROGRAM A21 (INPUT,OUTPUT);
VAR
  F1 : TEXT;          (* ФАЙЛОВАЯ ПЕРЕМЕННАЯ *)
  L1 : INTEGER;        (* ДЛИНА ВХОДНОГО ФАЙЛА *)
  S : REAL;            (* ДЕЙСТВ. ПЕРЕМЕННАЯ *)
  SUM : REAL;          (* СУММА *)
BEGIN
  SUM:=0;
  RESET(F1,'D1','PAS',L1); (* ОТКРЫТИЕ ФАЙЛА *)
  WHILE NOT EOF(F1) DO      (* ПРОВЕРКА НА КОНЕЦ ФАЙЛА *)
    BEGIN
      WHILE NOT EOLN(F1) DO (* ПРОВЕРКА НА КОНЕЦ СТРОКИ *)
        BEGIN
          READ(F1,S);        (* ЧТЕНИЕ ДАННЫХ ИЗ ФАЙЛА *)
          SUM:=SUM + S
        END;
        READLN(F1)          (* ПЕРЕХОД НА НОВУЮ СТРОКУ В ФАЙЛЕ *)
      END;
      WRITE('СУММА=', SUM:8:3);
    CLOSE(F1)              (* ЗАКРЫТИЕ ФАЙЛА *)
  END.
```

Здесь F1 — файловая переменная, которая устанавливает связь с файлом D1.PAS. Оператор RESET открывает файл для чтения из него данных. Оператор READ (F1,S) применяется для считывания данных из внешнего файла в переменную S. Если оператор ввода стоит вне цикла, то переменная S получит только одно значение S=0.54. Оператор CLOSE служит для закрытия файла. После него данные файла становятся недоступными программе.

Значения вычисленной суммы выводятся на экран дисплея.

Задача 48. На диске имеется файл с именем DAN.DAT, в котором записана скороговорка

```
=====
=
=      СШИТ КОЛПАК НЕПОКОЛПАКОВСКИ,      =
=      ВЫЛИТ КОЛОКОЛ НЕПОКОЛОКОВСКИ,      =
=      НАДО КОЛПАК ПЕРЕКОЛПАКОВАТЬ,        =
=      ПЕРЕВЫКОЛПАКОВАТЬ,                  =
=      НАДО КОЛОКОЛ ПЕРЕКОЛОКОВАТЬ,        =
=      ПЕРЕВЫКОЛОКОВАТЬ                    =
=
=====
```

Требуется подсчитать число буквы "К" в этой скороговорке.

```
(*-----*)
I      СКОРОГОВОРКА      I
(*-----*)

PROGRAM KOLPAK(INPUT,OUTPUT);
VAR
  F : TEXT;          (* ФАЙЛОВАЯ ПЕРЕМЕННАЯ *)
  SUMMA: INTEGER;    (* ЧИСЛО БУКВЫ "К" *)
  SIM: CHAR;         (* СИМВОЛ *)
```



```

BEGIN
  RESET(F, 'DAN', 'DAT' );          (* ОТКРЫТИЕ ФАЙЛА *)
  SUMMA:= 0;
  WHILE NOT EOF(F) DO                (* ПРОВЕРКА КОНЦА ФАЙЛА *)
    BEGIN
      WHILE NOT EOLN(F) DO           (* ПРОВЕРКА КОНЦА СТРОКИ *)
        BEGIN
          READ(F, SIM);
          IF SIM = 'K' THEN SUMMA:= SUMMA+1
        END;
        READLN(F);                   (* ПЕРЕХОД НА НОВУЮ СТРОКУ В ФАЙЛЕ *)
      END;
    END;
  CLOSE(F);
  WRITE('ЧИСЛО БУКВЫ "K" В СКОРОГОВОРКЕ ', SUMMA:3)
END.
=====
      ЧИСЛО БУКВЫ "K" В СКОРОГОВОРКЕ      22
=====

```

Запись файла. Под записью файла понимается вывод результатов программы из оперативной памяти ЭВМ на диск, т.е. создание нового файла на внешнем устройстве.

Для записи файла в программе необходимо выполнить следующие действия: открыть файл для записи (оператор REWRITE), вывести данные (оператор WRITE), закрыть файл для записи (оператор CLOSE).

Внешний файл, в который записываются данные из программы, часто называют выходными.

Общая форма записи файла имеет вид:

```

REWRITE (имя 1);
...
WRITE (имя 1, параметры);
...
CLOSE (имя 1);

```

В некоторых версиях микроЭВМ (в частности, ДБК) оператор REWRITE может иметь несколько аргументов, например:

```

REWRITE (имя 1, 'имя 2', 'имя 3', имя 4).

```

Аргументы имеют следующее назначение: *имя 1* — файловая переменная, которая в программе объявлена как файл. Она устанавливает связь с физическим именем создаваемого файла; *имя 2* — физическое имя создаваемого файла, который будет храниться на диске; *имя 3* — расширение физического имени. Можно указывать любые три символа, однако рекомендуется слово DAT. Аргумент можно не указывать, тогда по умолчанию будет слово DAT; *имя 4* — переменная целого типа, означающая размер создаваемого файла. Размер, выраженный в блоках, задает сам программист. Если указать размер, равный -1, то для выходного файла отводится наибольшая свободная область на диске (например, L=-1).

Оператор вывода для записи данных в файл обладает всеми свойствами обычного оператора WRITE. Если оператор вывода имеет вид WRITE (*параметры*), то данные выводятся на экран дисплея, а если WRITE (*имя 1, параметры*), то данные записываются в файл, который хранится на диске.

Пример записи данных в файл. Пусть в программе переменные A и B получают следующие значения: A = 510, B = 16.2. Записать в файл с именем D2.PAS строку в виде

```

A=510 _ _ _ _ _ B=16.2

```


Программа имеет вид:

```
PROGRAM A21A(INPUT,OUTPUT);
  VAR
    F2 : TEXT;      (* ФАЙЛОВАЯ ПЕРЕМЕННАЯ *)
    L2 : INTEGER;    (* ДЛИНА ВЫХОДНОГО ФАЙЛА *)
    A : INTEGER;     (* ЦЕЛОЕ ДАННОЕ *)
    B : REAL;        (* ДЕЙСТ-Е ДАННОЕ *)
  BEGIN
    L2:=1;
    A:=510; B:=16.2;
    REWRITE(F2, 'D2', 'PAS', L2);
    WRITE(F2, 'A=', A:3, ' ', B:4:1);
    CLOSE( F2 )
  END.
```

Здесь F2 — файловая переменная. Оператор REWRITE открывает файл для записи в него данных. Оператор вывода WRITE выводит строку символов в соответствии с форматом. Оператор CLOSE закрывает файл для записи.

Поскольку для выходной строки A=510 _ _ _ _ B=16.2 достаточен один блок, то переменной L2 присваивается значение 1.

Чтение и запись файлов. Действия с файлами могут быть разнообразными: иногда требуется читать данные из одного файла и записывать их в другой, иногда — корректировать один и тот же файл. В программе по мере необходимости указываются операторы для чтения и записи файлов, а также для их закрытия.

П р и м е р чтения и записи файлов:

```
PROGRAM A22(INPUT,OUTPUT);
  TYPE T = TEXT;      (* ТИП ФАЙЛА - ТЕКСТОВЫЙ *)
  VAR
    T1,T2 : T;        (* ФАЙЛОВЫЕ ПЕРЕМЕННЫЕ *)
    S : CHAR;         (* СИМВОЛЬНОЕ ДАННОЕ *)
    R : INTEGER;       (* ЦЕЛОЕ ДАННОЕ *)
  BEGIN
    RESET( T1,'FINP' ); (* ОТКРЫТИЕ ФАЙЛА ДЛЯ ЧТЕНИЯ *)
    REWRITE( T2,'FOUT' ); (* ОТКРЫТИЕ ФАЙЛА ДЛЯ ЗАПИСИ *)
    READ( T1, S );      (* ЧТЕНИЕ ДАННОГО ИЗ ФАЙЛА *)
    .....             (* ДЕЙСТВИЯ С ПЕРЕМЕННЫМИ S,R *)

    WRITE( T2, R );     (* ЗАПИСЬ ЗНАЧЕНИЯ R В ФАЙЛ *)
    CLOSE( T2 );        (* ЗАКРЫТИЕ ФАЙЛА ДЛЯ ЗАПИСИ *)
    CLOSE( T1 )         (* ЗАКРЫТИЕ ФАЙЛА ДЛЯ ЧТЕНИЯ *)
  END.
```

Здесь вводятся данные из файла FINP.DAT (файловая переменная T1) и выводятся в файл FOUT.DAT (файловая переменная T2). Переменная символьного типа S принимает значения одного символа из файла. Значение некоторой переменной целого типа R записывается в файл. При открытии файлов заданы не все аргументы, т.е. они будут назначены по умолчанию.

При использовании последовательных файлов нельзя одновременно открывать один и тот же файл для чтения и записи. Если появляется необходимость работы с одним файлом и для чтения и для записи, то нужно сначала открыть файл, обработать данные и закрыть его. Затем открыть этот же файл заново для другой цели, обработать данные и закрыть его.

Задача 49. В файле DAN1.DAT записаны целые числа в нескольких строках. Нужно вычислить сумму элементов каждой строки и результаты вместе с исходными данными записать в файл DAN2.DAT.

Программа имеет вид:

```

(*-----*)
I  ЧТЕНИЕ ДАННЫХ ИЗ ФАЙЛА  DAN1.DAT      I
I  И ЗАПИСЬ В ФАЙЛ  DAN2.DAT             I
(*-----*)
PROGRAM A21BV(INPUT,OUTPUT);
VAR
  F1,F2 : TEXT;          (* ФАЙЛОВЫЕ ПЕРЕМЕННЫЕ *)
  LIN1,LIN2 : INTEGER;    (* ДЛИНА ФАЙЛОВ *)
  X : INTEGER;            (* ДЕЙСТВ-НАЯ ПЕРЕМЕННАЯ *)
  SUM : INTEGER;          (* СУММА КАЖДОЙ СТРОКИ *)
BEGIN
  SUM:=0;
  RESET( F1,'DAN1','DAT',LIN1 );
  LIN2:=-1; (* НАИБОЛЬШАЯ СВОБОДНАЯ ОБЛАСТЬ НА ДИСКЕ *)
  REWRITE(F2,'DAN2','DAT',LIN2);
  WHILE NOT EOF(F1) DO
    BEGIN
      SUM:=0;
      WHILE NOT EOLN(F1) DO
        BEGIN
          READ(F1,X);
          WRITE(F2,X:3);
          SUM:=SUM + X
        END;
      WRITELN(F2,' ' 3,'СУММА=',SUM:4);
      READLN(F1)
    END;
  WRITE('РЕЗУЛЬТАТ НАХОДИТСЯ В ФАЙЛЕ DAN2.DAT');
  CLOSE(F1);
  CLOSE(F2)
END.

```

ИСХОДНЫЕ ДАННЫЕ
В ФАЙЛЕ DAN1.DAT

```

=====
=  3  5  2  =
=  4  9  1  =
=  1  7  1  =
=  6  2  8  =
=====

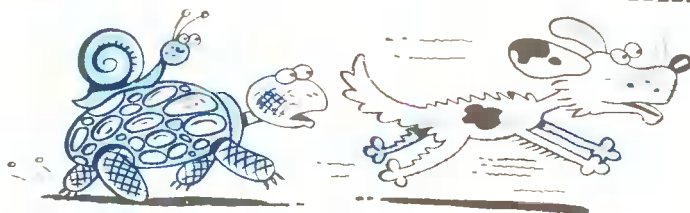
```

РЕЗУЛЬТАТ
В ФАЙЛЕ DAN2.DAT

```

=====
=  3  5  2  СУММА= 10  =
=  4  9  1  СУММА= 14  =
=  1  7  1  СУММА=  9  =
=  '   2  8  СУММА= 16  =
=====

```



Задача 50. В файле с именем VEL1.DAT хранится информация:

```

УЛИТКА           : 0.0015
ЧЕРЕПАХА         : 0.02
РЫБА              : 1
ПЕШЕХОД          : 1.4
КОННИЦА РЫСЬЮ    : 3.5
ПУХА              : 5
ЗАЯЦ              : 18
ОРЕЛ              : 24
ОХОТНИЧЬЯ СОБАКА : 25
ЗЕМЛЯ ПО ОРБИТЕ  : 30000

```


где указана скорость движения, м/с.

Необходимо создать файл с именем VEL2.DAT, в котором должна быть представлена следующая информация: заголовок таблицы; вся исходная информация; скорость движения, м/с и км/ч.

Таблица должна иметь вид:

ВИД		СКОРОСТЬ М/С	СКОРОСТЬ КМ/Ч
УЛИТКА	:	0.0015	0.0054
ЧЕРЕПАХА	:	0.0200	0.0720
РЫБА	:	1.0000	3.6000
ПЕШЕХОД	:	1.4000	5.0400
КОНИЦА РЫСЬЮ	:	3.5000	12.6000
МУХА	:	5.0000	18.0000
ЗАЯЧ	:	18.0000	64.8000
ОРЕЛ	:	24.0000	86.4000
ОХОТНИЧЬЯ СОБАКА	:	25.0000	90.0000
ЗЕМЛЯ ПО ОРБИТЕ	:	30000.0000	108000.0000

В комментариях программы даны все пояснения и назначения переменных. Для перевода единиц измерения м/с в км/ч использован коэффициент умножения 3,6.

Файловая переменная F1 обеспечивает связь с входным файлом, а F2 — с выходным.

Программа имеет вид:

```
(*-----*)
I          СКОРОСТЬ          I
(*-----*)
PROGRAM SKOR(INPUT,OUTPUT);
VAR
  F1 :TEXT;  (* ОХОТНОЙ ФАЙЛ      *)
  F2 :TEXT;  (* ВЫХОДНОЙ ФАЙЛ     *)
  V:REAL;    (* СКОРОСТЬ          *)
  SIM:CHAR;  (* РАБОЧАЯ ПЕРЕМЕННАЯ *)
BEGIN
  RESET(F1,'VEL1.DAT');
  REWRITE(F2,'VEL2.DAT');
  WRITELN(F2,'          ВИД          СКОРОСТЬ М/С  СКОРОСТЬ КМ/Ч');
  WRITELN(F2,'-----*-----*');
  WHILE NOT EOF(F1) DO          (* ПРОВЕРКА КОНЦА ФАЙЛА *)
    BEGIN
      WHILE NOT EOLN(F1) DO    (* ПРОВЕРКА КОНЦА СТРОКИ *)
        BEGIN
          READ(F1,SIM);
          WRITE(F2,SIM);
          IF SIM = ':' THEN BEGIN
            READ(F1,V);
            WRITE(F2,V:10:4);
            V:=V*3.6;
            WRITE(F2,' ':5, V:10:4)
          END
        ELSE;
        END;
        READLN(F1);          (* ПЕРЕХОД НА НОВУЮ СТРОКУ В ФАЙЛЕ F1 *)
        WRITELN(F2)         (* ПЕРЕХОД НА НОВУЮ СТРОКУ В ФАЙЛЕ F2 *)
      END;
    CLOSE(F1);
    CLOSE(F2);
    WRITE('РЕЗУЛЬТАТ СМОТРИ В ФАЙЛЕ VEL2.DAT')
  END.
```


Действие восьмое

КАК ОБЪЯТЬ НЕОБЪЯТНОЕ

Картина 1. ПРАВИЛА ХОРОШЕГО ТОНА

Дока. Ты любишь петь песни, Фока?

Фока. Конечно! Кто же их не любит петь? Но причем здесь песни, если мы занимаемся программированием?

Дока. А ты обратил внимание на то, что в тексте песни припев дается только один раз? Если нужно повторить его, то указывается одно слово: "Припев".

Фока. Правильно! Зачем повторять несколько раз одно и то же.

Дока. Этот же принцип действует и в программировании. Если какая-нибудь часть программы многократно повторяется, то ее оформляют отдельно от основной программы и называют подпрограммой. К ней обращаются при необходимости ее использования.

Более того, рекомендуется большую задачу разбивать на отдельные смысловые части (подпрограммы), программировать их отдельно, а затем объединять в единую программу. Использование подпрограмм считается хорошим стилем программирования. При этом уменьшается в целом объем программы и сокращается время на ее отладку.

Любая программа может содержать несколько подпрограмм, каждая из которых также может содержать подпрограммы. Для простоты изложения ограничимся подпрограммами, которые не содержат внутри себя другие подпрограммы, а вызов их осуществляется из основной программы.

Подпрограммы располагаются в разделе описаний основной программы. Структура программы с двумя подпрограммами P1 и P2 может иметь, например, такой вид:

```
PROGRAM ИМЯ(INPUT,OUTPUT);
```

```
(* РАЗДЕЛ ОПИСАНИЙ ОСНОВНОЙ ПРОГРАММЫ *)
```

```
LABEL - РАЗДЕЛ МЕТОК;  
CONST - РАЗДЕЛ КОНСТАНТ;  
TYPE - РАЗДЕЛ ТИПОВ;  
VAR - РАЗДЕЛ ПЕРЕМЕННЫХ;
```

```
-----  
I      ПОДПРОГРАММА P1      I
```

```
-----  
I      ПОДПРОГРАММА P2      I
```

```
(* РАЗДЕЛ ОПЕРАТОРОВ ОСНОВНОЙ ПРОГРАММЫ *)
```

```
BEGIN  
ОПЕРАТОРЫ;  
ВЫЗОВ ПОДПРОГРАММЫ P1;  
ОПЕРАТОРЫ;  
ВЫЗОВ ПОДПРОГРАММЫ P1;  
ОПЕРАТОРЫ;  
ВЫЗОВ ПОДПРОГРАММЫ P2;  
ОПЕРАТОРЫ;  
ВЫЗОВ ПОДПРОГРАММЫ P1;  
ОПЕРАТОРЫ;  
END.
```

Выполнение программы начинается с выполнения операторов основной программы. Как только появляется необходимость в выполнении подпрограммы, она вызывается по имени. Данные из основной программы передаются в под-

программу (входные данные), которая начинает выполняться. Затем результаты подпрограммы (выходные данные) передаются в основную программу в то место, откуда был сделан вызов подпрограммы, и продолжает выполняться основная программа.

Подпрограмма оформляется подобно основной программе, т.е. состоит из заголовка, раздела описаний и раздела операторов.

Все имена, представленные в разделе описаний основной программы, называются глобальными. Они действуют как в разделе операторов основной программы, так и в любой подпрограмме. Имена, представленные в разделе описаний подпрограммы, называются локальными. Они действуют только в рамках подпрограммы и недоступны операторам основной программы.

В языке Паскаль имеется два вида подпрограмм:

процедура (PROCEDURE), функция (FUNCTION). Рассмотрим каждый вид отдельно.



Картина 2. ПОКУПКА ПРОДУКТОВ

Дока. У нас с тобой, Фока, накопилось много дел. Давай в ближайшую субботу все переделаем? Составим план и строго по нему будем действовать.

Фока. Вот так всегда! Только задумаешь отдохнуть, а тут тебе — много дел! Ну, ладно, помогу составить план.

План друзей на ближайшую субботу:

- сделать зарядку;
- купить продукты;
- съесть хлеб и джем;
- посмотреть телевизор;
- покататься на лыжах;
- накормить кур, петуха и кота;
- поджарить картошку и добавить соль;
- съесть картошку;
- посмотреть телевизор

Фока. План получился какой-то большой...

Дока. Да еще не указали подробно поход в магазин за продуктами. Давай составим план покупки продуктов отдельно от основного плана. Собственно он и будет **являться** процедурой:

BEGIN

купить ПРОДУКТ 1;

купить ПРОДУКТ 2;

купить ПРОДУКТ 3

END

Теперь нам осталось **объединить** оба плана и указать, какие продукты мы купим и какие съедем:

PROGRAM ПЛАН (INPUT, OUTPUT);

PROCEDURE ПРОДУКТЫ (ДЕНЬГИ, ПРОДУКТ ПРОДУКТ ПРОДУКТ);

BEGIN

ВЗЯТЬ ПРОДУКТ 1 (*ХЛЕБ*);

ВЗЯТЬ ПРОДУКТ 2 (*ДЖЕМ*);

ВЗЯТЬ ПРОДУКТ 3 (*СОЛЬ*);

ОТДАТЬ ДЕНЬГИ

END;

BEGIN

СДЕЛАТЬ ЗАРЯДКУ;

КУПИТЬ ПРОДУКТЫ

ПОСМОТРЕТЬ ТЕЛЕВИЗОР;

ПОКАТАТЬСЯ НА ЛЫЖАХ;

НАКОРМИТЬ ПЕТУХА, КУР, КОТА;

ПОДЖАРИТЬ КАРТОШКУ И ДОБАВИТЬ СОЛЬ;

СЪЕСТЬ КАРТОШКУ;

ПОСМОТРЕТЬ ТЕЛЕВИЗОР

END.



Что характерно для этого плана? Покупку продуктов мы оформили в виде процедуры с именем ПРОДУКТЫ, а в основной программе указали лишь одно действие: купить продукты. С помощью параметров мы устанавливаем взаимодействие с процедурой: отдаем деньги, а получаем продукты.

Параметры процедуры ДЕНЬГИ, ПРОДУКТ 1, ПРОДУКТ 2, ПРОДУКТ 3 называются **формальными**. Им соответствуют фактические параметры при вызове процедуры в основной программе: ДЕНЬГИ, ХЛЕБ, ДЖЕМ, СОЛЬ, так что ПРОДУКТ 1 — ХЛЕБ, ПРОДУКТ 2 — ДЖЕМ, ПРОДУКТ 3 — СОЛЬ.

Фока. Все ясно! Если мы оставим формальные параметры в том же порядке, а некоторые фактические параметры поменяем местами

ДЕНЬГИ,

ПРОДУКТ 1,

ПРОДУКТ 2,

ПРОДУКТ 3,

↑
ДЕНЬГИ,

↓
СОЛЬ,

↓
ДЖЕМ,

↓
ХЛЕБ

то получим мешанину: купленный первый продукт хлеб мы обозначим именем СОЛЬ, купленный третий продукт соль мы обозначим именем ХЛЕБ. Значит, когда мы вернемся из магазина, то по нашему плану мы в этом случае съедим соль с джемом, не так ли?

Дока. Ты совершенно прав! К тому же в жареную картошку добавим хлеб вместо соли.

Для того чтобы мешанина не происходила, необходимо следить за параметрами: каждый фактический параметр должен соответствовать своему формальному.

Но вернемся к программированию!

Оформление процедуры в общем виде выглядит следующим образом:

```
PROCEDURE ИМЯ( ФОРМАЛЬНЫЕ ПАРАМЕТРЫ );
```

```

-----
I      РАЗДЕЛ ОПИСАНИЙ      I
-----
BEGIN
-----
I      РАЗДЕЛ ОПЕРАТОРОВ    I
-----
END;
```

В заголовке указывается служебное слово PROCEDURE, далее идет имя процедуры, за которым следуют формальные параметры, заключенные в круглые скобки.

Раздел описаний процедуры подобен программе и состоит из разделов меток, констант, типов, переменных и, в свою очередь, процедур и функций. В языке Паскаль допускается отсутствие раздела описаний.

Раздел операторов заключается в операторные скобки BEGIN и END, причем после END в отличие от основной программы ставится точка с запятой.

Вызывается процедура по ее имени:

имя (фактические параметры)

С помощью фактических и формальных параметров данные передаются из программы в процедуру и из процедуры в программу.

В качестве формальных параметров могут быть только переменные с указанием их типа. В качестве фактических параметров могут быть константы, переменные, выражения без указания их типа.

Процедура может содержать несколько операторов и несколько результатов выполнения. Каждый результат обозначается своим именем. В основной программе после вызова процедуры мы можем пользоваться этими результатами, сохраняя те же имена или давая другие.

Задача 51. А сейчас рассмотрим пример программирования. Пусть в программе необходимо многократно вычислять площадь квадрата $S1=a^2$ и площадь прямоугольника $S2=a \cdot b$ при различных значениях сторон a и b . Оформим эти вычисления в виде процедуры, используя любые формальные обозначения:

```

PROCEDURE PLACE( X,Y:REAL;  VAR SK,SP:REAL);
BEGIN
  SK:=X*X;      (* КВАДРАТ      *)
  SP:=X*Y       (* ПРЯМОУГОЛЬНИК *)
END;
```


Вызвать эту процедуру можно, используя фактические параметры:

PLACE (A,B,S1,S2),

где значения A и B передаются в процедуру параметрам X и Y, а результаты ее выполнения SK и SP возвращаются параметрам S1 и S2.

Пример полной программы:

```
(*-----*)
I      ООФОРМЛЕНИЕ ПРОЦЕДУРЫ      I
(*-----*)
PROGRAM PT21 (INPUT,OUTPUT);
  VAR A,B,S1,S2: REAL;

  PROCEDURE PLACE( X,Y:REAL; VAR SK,SP:REAL);
  BEGIN
    SK:=X*X;      (* КВАДРАТ      *)
    SP:=X*Y      (* ПРЯМОУГОЛЬНИК *)
  END;

  BEGIN
    WRITE('ВВЕДИТЕ ЗНАЧЕНИЯ A,B: ');
    READ(A,B);
    PLACE( A,B,S1,S2 );  (* ВЫЗОВ "PLACE" *)
    WRITE( 'S1=',S1:5:2, ' ':6, 'S2=',S2:5:2)
  END.
=====
= ВВЕДИТЕ ЗНАЧЕНИЯ A,B:  2.5  4      =
= S1=  6.25      S2= 10.00      =
=====
```

При вызове процедуры вместо переменных A и B можно указать конкретные значения, например:

PLACE (2.5, 4, S1, S2).

Допускается одинаковое обозначение соответствующих формальных и фактических параметров. Так, возможна процедура:

```
PROCEDURE PLACE( A,B:REAL; VAR S1,S2:REAL);
  BEGIN
    S1:=A*A;      (* КВАДРАТ      *)
    S2:=A*B      (* ПРЯМОУГОЛЬНИК *)
  END;
```

Параметры процедуры делятся на отдельные виды, но в простых программах чаще всего используются параметры-значения и параметры-переменные. В формальных параметрах процедуры перед параметрами-переменными ставится слово VAR. Данные этих параметров передаются в обоих направлениях: из программы в процедуру и, наоборот, из процедуры в программу. В случае переопределения этих данных в процедуре, они все равно доступны программе. Если в процедуре перед формальными параметрами не стоит слово VAR, то такие параметры называются параметрами-значениями. Их можно передавать только в одном направлении: из программы в процедуру. Эти параметры не могут быть результатом выполнения процедуры. Таким образом, параметры-значения могут быть только входными для процедуры, а параметры-переменные как входными, так и выходными.

Пр и м е р . Пусть имеется процедура

```
PROCEDURE SUMMA( A:INTEGER;  VAR B:INTEGER);
BEGIN
  A:=A+3;
  B:=B+3
END;
```

Посмотрим, чему будут равны значения A и B в основной программе после выполнения следующих операторов:

```
A:=5;  B:=5;
SUMMA( A, B );
WRITE( A, B );
```

Здесь значения $A = 5$ и $B = 5$ передаются в процедуру, где вычисляются новые значения $A = 5 + 3 = 8$ и $B = 5 + 3 = 8$. Новое значение B передается в программу, а значение A — нет. Поэтому в основной программе будем иметь $A = 5$, $B = 8$.

Если в процедуру нужно передать в качестве параметра не просто одно значение, а массив, то в этом случае фактическим параметром является имя массива. При этом формальный параметр указывается после слова VAR вместе с типом массива. Само же описание массива делается в разделе TYPE основной программы.

Пр и м е ч а н и е . В языке Паскаль допускается использование процедур без параметров. В этом случае отсутствуют как формальные, так и фактические параметры.

Рассмотрим задачи с использованием процедур с массивами и без них.

Задача 52. Судьи по фигурному катанию на коньках не всегда справедливы, и в их оценках случается большой разброс. Требуется составить программу определения минимальной и максимальной оценок, данных судьями.

```
(*-----*)
I      ФИГУРНОЕ КАТАНИЕ      I
-----*)
PROGRAM MINMAX(INPUT,OUTPUT);
CONST N = 9;  (* ЧИСЛО СУДЕЙ *)
TYPE MASSIV = ARRAY[1..N]OF REAL;
VAR
  A  MASSIV;  (* МАССИВ ОЦЕНОК *)
  I: INTEGER; (* ПАРАМЕТР ЦИКЛА *)
  MAX : REAL; (* МАКСИМАЛЬНАЯ ОЦЕНКА *)
  MIN : REAL; (* МИНИМАЛЬНАЯ ОЦЕНКА *)

(* ПРОЦЕДУРА " MAXMIN"
-----*)
PROCEDURE MAXMIN( K:INTEGER;
                  VAR X: MASSIV;
                  VAR MAX,MIN: REAL );

  VAR J: INTEGER;
BEGIN
  MAX:=X[1];
  MIN:=X[1];
  FOR J:=1 TO K DO
  BEGIN
    IF X[J] > MAX THEN MAX:=X[J];
    IF X[J] < MIN THEN MIN:=X[J]
  END
END;
```



(* ОСНОВНАЯ ПРОГРАММА
-----*)

```
BEGIN
  WRITELN('КАКИЕ ОЦЕНКИ ДАЛИ СУДЬИ ? ');
  FOR I:=1 TO N DO READ(A[I]);
  MAXMIN(N,A,MAX,MIN); (* ВЫЗОВ ПРОЦЕДУРЫ *)
  WRITELN('МАКСИМАЛЬНАЯ ОЦЕНКА =',MAX:4:1);
  WRITE('МИНИМАЛЬНАЯ ОЦЕНКА =', MIN:4:1)
END.
-----
= КАКИЕ ОЦЕНКИ ДАЛИ СУДЬИ ? =
= 5.6 5.8 5.7 5.8 5.6 5.8 5.7 5.5 5.7 =
= МАКСИМАЛЬНАЯ ОЦЕНКА = 5.8 =
= МИНИМАЛЬНАЯ ОЦЕНКА = 5.5 =
-----
```

В процедуре MAXMIN определяются максимальная и минимальная оценки судей. Переменная J и формальные параметры процедуры K, X, MAX, MIN являются локальными. В основной программе происходит ввод значений массива оценок, вызов процедуры и вывод результатов. Константа N, тип MASSIV, а также переменные A, I, MAX и MIN являются глобальными.

Задача 53. Определить длину окружности C и площадь S круга, ограниченного этой окружностью, если радиус равен R. Определить удаление L центра окружности от начала координат O. Координаты центра окружности равны X и Y (рис.2).

Вычисление длины окружности, площади круга и удаления центра окружности оформим в виде процедуры с именем КРУГ. В основной программе организуем ввод исходных данных R,X,Y, затем вызов процедуры и, наконец, вывод результатов.

Программу составим в двух вариантах. В первом варианте процедура КРУГ имеет параметры: входные R,X,Y, выходные C,S,L. Соответственно вызов этой процедуры имеет параметры.

Второй вариант составлен без параметров. Программы двух вариантов имеют вид:

1 вариант:

```
(*-----
!      ВЫЧИСЛЕНИЕ ПАРАМЕТРОВ ОКРУЖНОСТИ      !
!-----*)
PROGRAM E24(INPUT,OUTPUT);
  VAR R : REAL; (* РАДИУС *)
      X,Y : REAL; (* КООРДИНАТЫ ЦЕНТРА *)
      C : REAL; (* ДЛИНА ОКРУЖНОСТИ *)
      S : REAL; (* ПЛОЩАДЬ КРУГА *)
      L : REAL; (* УДАЛЕНИЕ ЦЕНТРА *)

  (* ПРОЦЕДУРА 'КРУГ'
  -----*)
PROCEDURE КРУГ;
  CONST PI=3.14;
BEGIN
  C:= 2 * PI * R ;
  S:= PI * SQR( R );
  L:= SQRT( SQR ( X ) + SQR ( Y ))
END;
```

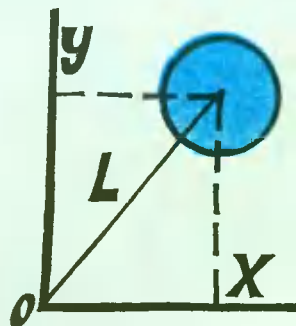


Рис.2. Окружность, удаленная от центра координат


```

      (* ОСНОВНАЯ ПРОГРАММА
      -----*)
BEGIN
  WRITELN('ВВЕДИТЕ ЗНАЧЕНИЯ R,X,Y:');
  READ( R,X,Y );
  КРУГ;      (* ВЫЗОВ ПРОЦЕДУРЫ 'КРУГ'*)
  WRITELN( 'ДЛИНА ОКРУЖНОСТИ = ' , C:6:2 );
  WRITELN( 'ПЛОШАДЬ КРУГА = ' , S:6:2);
  WRITELN( 'УДАЛЕНИЕ ЦЕНТРА = ' , L:6:2 )
END.
=====
=
=      ВВЕДИТЕ ЗНАЧЕНИЯ R,X,Y:
=      3 8 6
=      ДЛИНА ОКРУЖНОСТИ = 18.84
=      ПЛОШАДЬ КРУГА = 28.26
=      УДАЛЕНИЕ ЦЕНТРА = 10.00
=
=====

```

II вариант:

```

(*-----*
!      ВЫЧИСЛЕНИЕ ПАРАМЕТРОВ ОКРУЖНОСТИ      !
!-----*)
PROGRAM E24(INPUT,OUTPUT);
  VAR R : REAL; (* РАДИУС *)
      X,Y : REAL; (* КООРДИНАТЫ ЦЕНТРА *)
      C : REAL; (* ДЛИНА ОКРУЖНОСТИ *)
      S : REAL; (* ПЛОШАДЬ КРУГА *)
      L : REAL; (* УДАЛЕНИЕ ЦЕНТРА *)

  (* ПРОЦЕДУРА 'КРУГ'
  -----*)
  PROCEDURE КРУГ( R,X,Y:REAL; VAR C,S,L: REAL );
    CONST PI=3.14;
  BEGIN
    C:= 2 * PI * R ;
    S:= PI * SQR( R );
    L:= SQRT( SQR ( X ) + SQR ( Y ))
  END;

  (* ОСНОВНАЯ ПРОГРАММА
  -----*)
BEGIN
  WRITELN('ВВЕДИТЕ ЗНАЧЕНИЯ R,X,Y:');
  READ( R,X,Y );
  КРУГ(R,X,Y,C,S,L); (* ВЫЗОВ ПРОЦЕДУРЫ 'КРУГ'*)
  WRITELN( 'ДЛИНА ОКРУЖНОСТИ = ' , C:6:2 );
  WRITELN( 'ПЛОШАДЬ КРУГА = ' , S:6:2);
  WRITELN( 'УДАЛЕНИЕ ЦЕНТРА = ' , L:6:2 )
END.
=====
=
=      ВВЕДИТЕ ЗНАЧЕНИЯ R,X,Y:
=      3 8 6
=      ДЛИНА ОКРУЖНОСТИ = 18.84
=      ПЛОШАДЬ КРУГА = 28.26
=      УДАЛЕНИЕ ЦЕНТРА = 10.00
=
=====

```


Задача 54. Приведем пример программы с двумя процедурами: одна предназначена для ввода значений массива, другая — для вычисления суммы произведения элементов массива.

```
(*-----*)
I  ОФОРМЛЕНИЕ ПРОЦЕДУРЫ С МАССИВАМИ I
-----*)
PROGRAM PT13(INPUT,OUTPUT);
  TYPE MASSIV= ARRAY[1..20]OF REAL;
  VAR   A,B: MASSIV;
        SUM,PR: REAL;

      (* ПРОЦЕДУРА "ВВОД"
      -----*)
PROCEDURE ВВОД( N:INTEGER; VAR X:MASSIV);
  VAR I: INTEGER;
BEGIN
  WRITELN('ВВЕДИТЕ ЗНАЧЕНИЯ МАССИВА:');
  FOR I:=1 TO N DO  READ(X[I]);
END;

      (* ПРОЦЕДУРА "СУММА"
      -----*)
PROCEDURE СУММА( N:INTEGER;
                VAR X:MASSIV;
                VAR SUM,PR:REAL);
  VAR I:INTEGER;
BEGIN
  SUM:=0;  PR:=1;
  FOR I:=1 TO N DO
    BEGIN
      SUM:=SUM+I;
      PR:=PR*I
    END
  END
END;

      (* ОСНОВНАЯ ПРОГРАММА
      -----*)
BEGIN
  ВВОД(8,A);          (* ВЫЗОВ ПРОЦЕДУРЫ *)
  СУММА(8,A,SUM,PR);  (* ВЫЗОВ ПРОЦЕДУРЫ *)
  WRITELN('SUM=', SUM:7:2, ' ' ':3,'PR=', PR);
  WRITELN;
  ВВОД(15,B);         (* ВЫЗОВ ПРОЦЕДУРЫ *)
  СУММА(15,B,SUM,PR); (* ВЫЗОВ ПРОЦЕДУРЫ *)
  WRITELN('SUM=', SUM:7:2, ' ' ':3,'PR=', PR)
END.

=====
=      ВВЕДИТЕ ЗНАЧЕНИЯ МАССИВА:      =
=      4 3 1 1 2 6 2 1                  =
=      SUM=  36.00    PR=  4.032000E+04  =
=                                          =
=      ВВЕДИТЕ ЗНАЧЕНИЯ МАССИВА:      =
=      3 6 1 2 7 1 2 1 1 4 2 2 1 2 2   =
=      SUM= 120.00    PR=  1.307674E+12  =
=====
```

В основной программе описан тип массива, размерность которого 20, реально используются массивы А размерностью 8 и В размерностью 1. Обращение к процедурам происходит дважды, сначала с одним массивом, затем с другим.

Задача 55. Упорядочить имена FOCA, DOCA, NICA, MIC в соответствии с латинским алфавитом.


```

(*-----*)
I      упорядочить имена      I
(*-----*)

PROGRAM LB1(INPUT,OUTPUT);
CONST N = 5;      (* ЧИСЛО ИМЕН *)
      L = 4;      (* ДЛИНА ИМЕНИ *)
VAR   I: INTEGER;
      NAME: ARRAY[1..N,1..L] OF CHAR;

      (* ПРОЦЕДУРА СОРТИРОВКИ
      -----*)

PROCEDURE SORT;
VAR   STROKA: ARRAY[1..L] OF CHAR;
      PR: BOOLEAN;
      J: INTEGER;

BEGIN
  PR:=TRUE;
  WHILE PR DO
  BEGIN
    PR:=FALSE;
    FOR J:=1 TO N-1 DO
      IF NAME[J]>NAME[J+1] THEN
        BEGIN
          STROKA:=NAME[J];
          NAME[J]:=NAME[J+1];
          NAME[J+1]:=STROKA;
          PR:=TRUE
        END
      END
    END
  END;
  (* ОСНОВНАЯ ПРОГРАММА
  -----*)

BEGIN
  WRITELN('УКАЖИТЕ НЕУПОРЯДОЧЕННЫЕ ИМЕНА:');
  READLN;
  FOR I:=1 TO N DO READLN(NAME[I]);
  SORT;      (* ВЫЗОВ ПРОЦЕДУРЫ СОРТИРОВКИ *)
  WRITELN('УПОРЯДОЧЕННЫЕ ИМЕНА:');
  FOR I:=1 TO N DO WRITELN(NAME[I])
END.
=====
      УКАЖИТЕ НЕУПОРЯДОЧЕННЫЕ ИМЕНА:      =
      FOCA                                  =
      DOCA                                  =
      NIKA                                  =
      MICA                                  =
      MIC                                   =
      УПОРЯДОЧЕННЫЕ ИМЕНА:                  =
      DOCA                                  =
      FOCA                                  =
      MIC                                   =
      MICA                                  =
      NIKA                                  =
=====

```

**КАК БУДУТ
УПОРЯДОЧЕНЫ
ИМЕНА:
ФОКА, ДОКА,
НИКА,
МИКА, МИК?**

Картина 3. ПОКУПКА МОЛОКА

Дока. После стольких трудов так хочется попить молочка!

Фока. И мне тоже! Я сейчас быстренько сбегая в магазин за молоком.

Дока. Это отличная идея, но сначала попытаемся оформить ее в виде под-
программы-функции, а уж если получится, тогда и попьем молочка!

Чтобы понять суть подпрограммы функции, вернемся к покупке продукта (см. картину 2 этого действия), где мы использовали процедуру. Поскольку в данном случае продукт покупки только один — молоко, то процедуру можно упростить. Вместо указания "купить ПРОДУКТЫ" дадим конкретное указание "купить МОЛОКО", при этом функция должна получить имя МОЛОКО

PROGRAM ПЛАН(INPUT,OUTPUT);

FUNCTION МОЛОКО(ДЕНЬГИ);

BEGIN

ВЗЯТЬ МОЛОКО;
ОТДАТЬ ДЕНЬГИ

END;

BEGIN

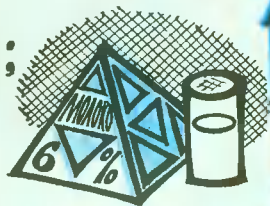
ПОЧИСТИТЬ ЗУБЫ;

КУПИТЬ

ВЫПИТЬ

РАЗОБРАТЬ ТЕМУ О ПОДПРОГРАММАХ

END.



Функция, как и процедура, может содержать несколько операторов, но результат ее выполнения только один. Этот единственный результат обозначается именем функции и передается в основную программу.

Фока. А в этом примере может быть несколько входных параметров?

Дока. Конечно! Например, можно принести в магазин бидон, чтобы в нем налили молока, или сдать пустые бутылки из-под молока:

PROGRAM ПЛАН(INPUT,OUTPUT);

FUNCTION МОЛОКО(ДЕНЬГИ,СОСУД);

BEGIN

СДАТЬ СОСУД;
ВЗЯТЬ МОЛОКО;
ОТДАТЬ ДЕНЬГИ

END;

BEGIN

ПОЧИСТИТЬ ЗУБЫ;

КУПИТЬ

ВЫПИТЬ

РАЗОБРАТЬ ТЕМУ О ПОДПРОГРАММАХ

END.



Главное здесь то, что из магазина мы можем принести только молоко. Если же мы купим еще и конфеты, то нам придется иметь дело с процедурой, но не с функцией.

В общем виде функция записывается следующим образом:

FUNCTION ИМЯ(ФОРМАЛЬНЫЕ ПАРАМЕТРЫ) : ТИП;

I	РАЗДЕЛ ОПИСАНИЙ	I
BEGIN		
I	РАЗДЕЛ ОПЕРАТОРОВ	I
END;		

Поскольку результат обозначается именем функции, то после формальных параметров указывается тип функции, который должен совпадать с типом результата вычислений. Вызывается функция по ее имени с указанием фактических параметров. При этом вызов функции можно делать непосредственно внутри выражения, подобно тому, как используются стандартные встроенные функции, например синус SIN(X).

Задача 56. Сколько раз бывало: видишь перед собой гору, она кажется совсем близко, но идешь-идешь к ней... Впрочем, если известна высота горы и угол возвышения, то можно определить расстояние по формуле $S=H/\operatorname{tg}\beta$.

Если угол задается в градусах, то в программе его нужно перевести в радианы: $x = \beta \cdot \pi / 180$. Поскольку в языке Паскаль нет греческих букв β и π , то заменяем их соответственно на BETTA и PI. Стандартная функция тангенса также отсутствует в языке Паскаль, поэтому вместо него используем отношения $\sin x / \cos x$.

```
(*)-----(*)  
I      РАССТОЯНИЕ ДО ГОРЫ      I  
-----*)  
PROGRAM PT4(INPUT,OUTPUT);  
CONST PI=3.14159;  
VAR    H:REAL; (* ВЫСОТА *)  
        BETTA:REAL; (* УГОЛ ВОЗВЫШЕНИЯ *)  
  
(*  
                                     *  
                                     I  
                                     I  
                                     I   H  
                                     I  
D      *      ) BETTA      I  
/I\ -----I  
/\          S              (*)  
  
(* ФУНКЦИЯ ВЫЧИСЛЕНИЯ РАССТОЯНИЯ S  
-----*)  
FUNCTION S( H:REAL; BETTA:REAL):REAL;  
VAR X:REAL;  
BEGIN  
X:= BETTA * PI/180;  
S:=H/( SIN(X)/COS(X) )  
END;
```



(* ОСНОВНАЯ ПРОГРАММА
-----*)

```
BEGIN
  WRITE('ЧЕМУ РАВЕН УГОЛ ВОЗВЫШЕНИЯ? ');
  READLN(BETTA);
  WRITE('ЧЕМУ РАВНА ВЫСОТА? ');
  READLN(H);
  WRITE('РАССТОЯНИЕ = ', S(H,BETTA):10:4)
END.
```

```
-----
=   ЧЕМУ РАВЕН УГОЛ ВОЗВЫШЕНИЯ? 30   =
=   ЧЕМУ РАВНА ВЫСОТА? 10           =
=   РАССТОЯНИЕ =      17.3205        =
-----
```

Вызов функции осуществляется непосредственно в операторе вывода: S (H,BETTA).

Задача 57. Найти разность двух факториалов $F=m!-K!$. Вспомним, что $n!$ представляет собой произведение n чисел натурального ряда $1 \cdot 2 \cdot 3 \dots n$. Вычисление факториала $n!$ оформим в виде функции FACT, а затем будем обращаться к ней со значениями m и K .

(*-----
I ОФОРМЛЕНИЕ ФУНКЦИИ I
-----*)

```
PROGRAM PT11(INPUT,OUTPUT);
  VAR F,M,K : INTEGER;
```

(* ФУНКЦИЯ "FACT"
-----*)

```
FUNCTION FACT(N:INTEGER): INTEGER;
  VAR P,I:INTEGER;
BEGIN
  P:=1;
  FOR I:=2 TO N DO
    P:=P*I;
  FACT:=P;
END;
```

(* ОСНОВНАЯ ПРОГРАММА
-----*)

```
BEGIN
  WRITE('ВВЕДИТЕ ЗНАЧЕНИЯ M,K: ');
  READ(M,K);
  F:=FACT(M)-FACT(K);
  WRITELN('F=', F:5)
END.
```

```
-----
=   ВВЕДИТЕ ЗНАЧЕНИЯ M,K: 7 4   =
=   F= 5016                     =
-----
```

Внутри функции имена N,P,I являются локальными. Результат вычисления факториала обозначается именем функции FACT.

В основной программе переменные F,M,K являются глобальными. При вычислении значения F происходит дважды обращение к функции: FACT(M) и FACT(K), причем это делается в правой части оператора присваивания.

Задача 58. Дан массив целых чисел $\{A_i\}$, где $i=1,2,3, \dots, M$. Пусть M равно 15. Вычислить сумму элементов с 1-го по 12-й и сумму элементов с 8-го по 15-й:

$$A_1 + A_2 + A_3 + \dots + A_{12}$$

и

$$A_8 + A_9 + A_{10} + \dots + A_{15}.$$

Затем найти произведение этих сумм. Вычисление суммы оформить как функцию SUMMA.

Программа имеет вид:

```

(*-----*
!      ВЫЧИСЛЕНИЕ СУММЫ ЦЕЛЫХ ЧИСЕЛ      !
!      С ИСПОЛЬЗОВАНИЕМ ФУНКЦИИ           !
!-----*)
PROGRAM E26( INPUT,OUTPUT ) ;
CONST M = 15;  (* РАЗМЕР МАССИВА *)
VAR
  A : ARRAY [1..M] OF INTEGER; (* МАССИВ *)
  P : INTEGER;      (* ПРОИЗВЕДЕНИЕ СУММ *)
  J : INTEGER;      (* ПАРАМЕТР ЦИКЛА *)

  (* ФУНКЦИЯ "SUMMA"
  -----*)
FUNCTION SUMMA( N, K :INTEGER ) :INTEGER;
  VAR I, S : INTEGER;
BEGIN
  S:= 0;
  FOR I:= N TO K DO
    S:=S + A[I];
  SUMMA:= S
END;

  (* ОСНОВНАЯ ПРОГРАММА
  -----*)
BEGIN
  WRITELN('ВВЕДИТЕ ЗНАЧЕНИЯ МАССИВА "A":');
  FOR J:= 1 TO M DO READ( A[J] );
  P:= SUMMA( 1,12 ) * SUMMA( 8,15 );
  WRITELN;
  WRITELN('ПРОИЗВЕДЕНИЕ РАВНО : ',P:6 )
END.

=====
=
=      ВВЕДИТЕ ЗНАЧЕНИЯ МАССИВА "A":      =
=      2 3 1 4 5 2 7 1 2 3 3 1 2 5 4      =
=
=      ПРОИЗВЕДЕНИЕ РАВНО :      714      =
=
=====

```

Картина 4. ОБРАЩЕНИЕ К САМОМУ СЕБЕ

Дока. Фока, ты можешь обратиться к самому себе?

Фока. Могу! В некоторых случаях я так и поступаю. Если мне становится рутинно и ничего не хочется делать, я обращаюсь к самому себе: "Уважаемый Фока, перестань хандрить и займись делом!" Иногда мне это помогает.

Дока. В программировании имеется тоже возможность обращения процедуры или функции к самой себе. При этом получается интереснейший эффект: циклическую часть программы можно составить без операторов цикла.

Способ обращения процедуры или функции к самой себе называют рекурсией.

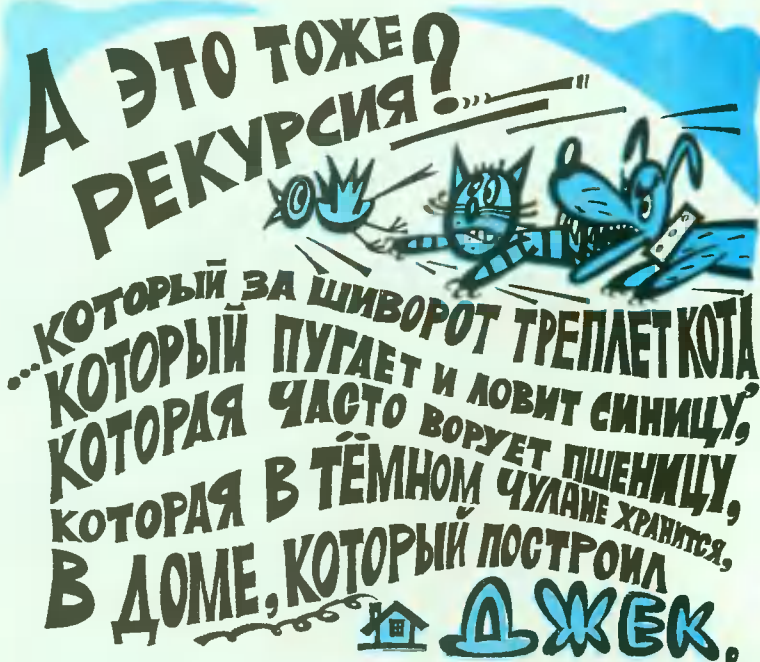
С помощью рекурсии удобно представлять те задачи, которые сводятся на подзадачи того же типа, но меньшей размерности. Например, вычисление факториала можно представить опять-таки через факториал:

$$n! = n \cdot (n-1)!$$

Представление факториала в виде последовательности операции умножения является итерацией:

$$n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot n.$$

Итерация программируется с помощью циклов.



Задача 59. Рассмотрим вычисление факториала с использованием рекурсивной функции, а также для сравнения — с применением цикла (следует напомнить, что $0! = 1$ и $1! = 1$).

И в том и в другом случае вычисление факториала оформлено как подпрограмма FUNCTION. В основной программе задается значение n , вызывается подпрограмма и выводятся результаты вычислений.

```
(* -----  
! ИТЕРАТИВНОЕ ВЫЧИСЛЕНИЕ ФАКТОРИАЛА !  
-----*)  
PROGRAM FI(INPUT,OUTPUT);  
  VAR  FAC : REAL;  
       N : INTEGER;  
  
      (* ФУНКЦИЯ ВЫЧИСЛЕНИЯ ФАКТОРИАЛА  
      -----*)  
FUNCTION FACTORIAL(N:INTEGER):REAL;  
  VAR  F : REAL;  
       I : INTEGER;  
  
BEGIN  
  IF (N=0) OR (N=1) THEN FACTORIAL:=1  
  ELSE BEGIN  
    F:=1;  
    FOR I:=2 TO N DO  
      F:=F*I;  
    FACTORIAL:=F;  
  END;  
END;
```



```

      (* ОСНОВНАЯ ПРОГРАММА
      -----*)
BEGIN
  WRITELN(' ВВЕДИТЕ ЗНАЧЕНИЕ N ');
  READ(N);
  FAC:=FACTORIAL(N);  (* ВЫЗОВ ФУНКЦИИ FACTORIAL *)
  WRITELN(' ФАКТОРИАЛ =',FAC);
END.
(*-----
  ! РЕКУРСИВНОЕ ВЫЧИСЛЕНИЕ ФАКТОРИАЛА !
  -----*)
PROGRAM FR(INPUT,OUTPUT);
  VAR  FAC : REAL;
      N : INTEGER;

      (* ФУНКЦИЯ ВЫЧИСЛЕНИЯ ФАКТОРИАЛА
      -----*)
FUNCTION FACTORIAL(N:INTEGER):REAL;
  BEGIN
    IF (N=0) OR (N=1) THEN FACTORIAL:=1
    ELSE FACTORIAL:=FACTORIAL(N-1) * N ;
  END;

      (* ОСНОВНАЯ ПРОГРАММА
      -----*)
BEGIN
  WRITELN(' ВВЕДИТЕ ЗНАЧЕНИЕ N ');
  READ(N);
  FAC:=FACTORIAL(N);  (* ВЫЗОВ ФУНКЦИИ FACTORIAL *)
  WRITELN(' ФАКТОРИАЛ =',FAC);
END.

```

Задача 60. В 1228 г. итальянский математик Фибоначчи сформулировал интересную задачу: "Некто поместил пару кроликов в некоем месте, огороженном со всех сторон стеной, чтобы узнать, сколько пар кроликов родится при этом в течение года, если природа кроликов такова, что через месяц пара кроликов производит на свет другую пару, а рождают кролики со второго месяца после своего рождения".

Эта задача сводится к последовательности чисел

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

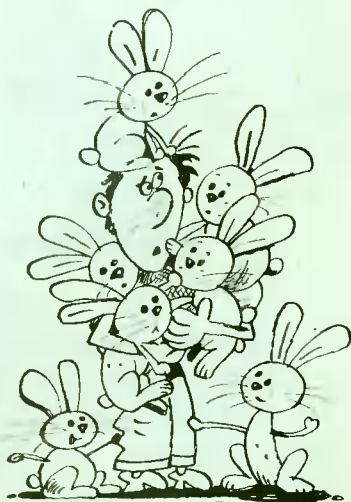
где каждый последующий член равен сумме двух предыдущих, за исключением первых двух членов.

Программу можно составить с применением рекурсивной функции FIB:

```

(*-----
  I          НУ И КРОЛИКИ !          I
  -----*)
PROGRAM КРОЛИК(INPUT,OUTPUT);
  VAR  KR:INTEGER; (* ЧИСЛО КРОЛИКОВ *)
      N:INTEGER; (* ЧИСЛО МЕСЯЦЕВ *)
      (* ----- *)
FUNCTION FIB(N:INTEGER):INTEGER;
  BEGIN
    IF (N=1)OR(N=2) THEN FIB:=1
    ELSE FIB:=FIB(N-1)+FIB(N-2)
  END;
  (* ----- *)

```




```

BEGIN
  WRITE('ВВЕДИТЕ ЗНАЧЕНИЕ N: ');
  READ(N);
  KR:=FIB(N);  (* ВЫЗОВ ФУНКЦИИ *)
  WRITE('ЧИСЛО КРОЛИКОВ =', KR:5)
END.
=====
=   ВВЕДИТЕ ЗНАЧЕНИЕ N:  12   =
=   ЧИСЛО КРОЛИКОВ =  144   =
=====

```

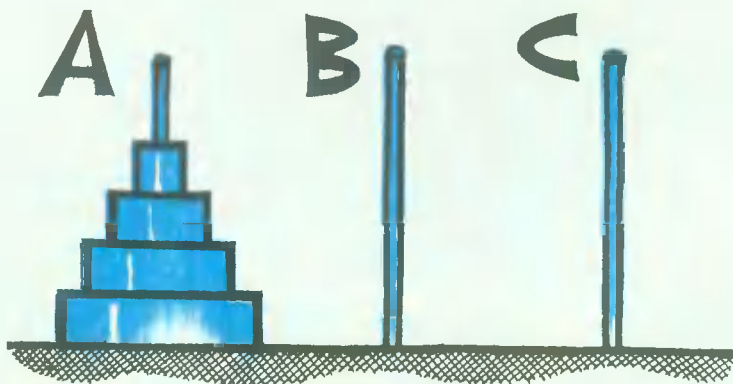
Первый вызов функции происходит в основной программе, а затем начинается рекурсивный вызов внутри функции.

Задача 61. Ханойская башня. В одной из древних легенд говорится: "В храме Бенареса находится бронзовая плита с тремя алмазными стержнями. На один из стержней бог при сотворении мира нанизал 64 диска разного диаметра из чистого золота. Наибольший диск лежит на бронзовой плите, а остальные образуют пирамиду, сужающуюся сверху. Это — башня Браммы. Работая день и ночь, жрецы переносят диски с одного стержня на другой, следуя законам Браммы:

- 1) диски можно перемещать с одного стержня на другой только по одному;
- 2) нельзя класть больший диск на меньший;
- 3) при переносе дисков с одного стержня на другой можно использовать промежуточный третий стержень, на котором диски должны находиться тоже только в виде пирамиды.

Когда все 64 диска будут перенесены с одного стержня на другой, наступит конец света".

Эта древняя легенда породила задачу о Ханойской башне: переместить n дисков со стержня А на стержень В, используя промежуточный С и соблюдая законы Браммы.



Если в программе использовать рекурсивную процедуру, то программа становится удивительно маленькой: два рекурсивных вызова процедуры и оператор вывода результатов перемещения дисков. В процедуре используются формальные обозначения дисков X,Y,Z и число дисков N. Результаты получены при $N=4$ и при фактических обозначениях стержней А, В, С.


```

(*-----*)
1      ХАНОЙСКАЯ БАШНЯ      1
(*-----*)
PROGRAM HANOI(INPUT,OUTPUT);
VAR
  N: INTEGER;
  X,Y,Z : CHAR;

  (* ПРОЦЕДУРА "HAN"
  -----*)
PROCEDURE HAN(N: INTEGER; VAR X,Y,Z: CHAR);
BEGIN
  IF N>0 THEN
    BEGIN
      HAN(N-1,X,Z,Y);      (* РЕКУРСИЯ *)
      WRITELN('ПЕРЕМЕСТИТЬ ДИСК СО СТЕРЖНЯ ',
              X, ' НА СТЕРЖЕНЬ ', Y);
      HAN(N-1,Z,Y,X)      (* РЕКУРСИЯ *)
    END
  END;

  (* ОСНОВНАЯ ПРОГРАММА
  -----*)
BEGIN
  WRITELN('ВВЕДИТЕ ЗНАЧЕНИЯ  N,X,Y,Z :');
  READLN(N,X,Y,Z);
  HAN(N,X,Y,Z);      (* ПЕРВЫЙ ВЫЗОВ "HAN" *)
END.
=====
=
= ВВЕДИТЕ ЗНАЧЕНИЯ  N,X,Y,Z :
= 4ABC
= ПЕРЕМЕСТИТЬ ДИСК СО СТЕРЖНЯ А НА СТЕРЖЕНЬ С
= ПЕРЕМЕСТИТЬ ДИСК СО СТЕРЖНЯ А НА СТЕРЖЕНЬ В
= ПЕРЕМЕСТИТЬ ДИСК СО СТЕРЖНЯ С НА СТЕРЖЕНЬ В
= ПЕРЕМЕСТИТЬ ДИСК СО СТЕРЖНЯ А НА СТЕРЖЕНЬ С
= ПЕРЕМЕСТИТЬ ДИСК СО СТЕРЖНЯ В НА СТЕРЖЕНЬ А
= ПЕРЕМЕСТИТЬ ДИСК СО СТЕРЖНЯ А НА СТЕРЖЕНЬ С
= ПЕРЕМЕСТИТЬ ДИСК СО СТЕРЖНЯ А НА СТЕРЖЕНЬ В
= ПЕРЕМЕСТИТЬ ДИСК СО СТЕРЖНЯ С НА СТЕРЖЕНЬ В
= ПЕРЕМЕСТИТЬ ДИСК СО СТЕРЖНЯ С НА СТЕРЖЕНЬ А
= ПЕРЕМЕСТИТЬ ДИСК СО СТЕРЖНЯ В НА СТЕРЖЕНЬ А
= ПЕРЕМЕСТИТЬ ДИСК СО СТЕРЖНЯ С НА СТЕРЖЕНЬ В
= ПЕРЕМЕСТИТЬ ДИСК СО СТЕРЖНЯ А НА СТЕРЖЕНЬ С
= ПЕРЕМЕСТИТЬ ДИСК СО СТЕРЖНЯ А НА СТЕРЖЕНЬ В
= ПЕРЕМЕСТИТЬ ДИСК СО СТЕРЖНЯ С НА СТЕРЖЕНЬ В
=
=====

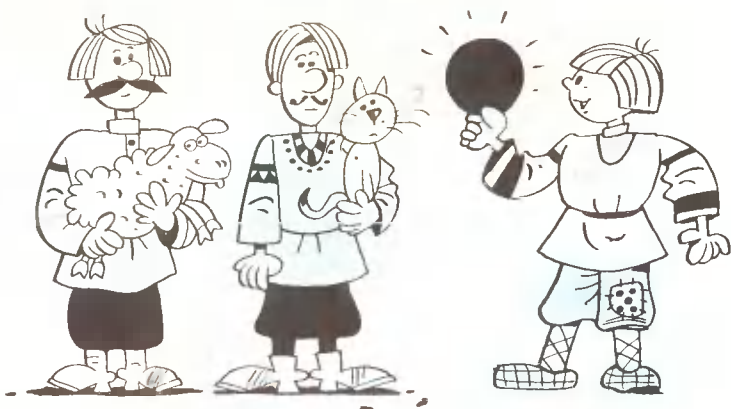
```

Картина 5. СКАЗ ПРО ТО, КАК ТРИФОН АВТОМОБИЛИ РАЗМЕЩАЛ

А было это давным-давно... В некотором царстве, в некотором государстве жили-были старик со старухой, и было у них три сына. Как сыновья подросли, отправил их старик по белу свету в разные стороны счастья искать.

Старшему сыну дал в дорогу овцу, среднему — кота, а младшему, Трифону, ничего не досталось. Думал, думал старик, да и говорит:

— А возьми-ка ты, Трифон, в дорогу две дискетки, кои в сундуке хранятся: одна со стандартным программным обеспечением, другая — чистая, для души.



Взял Трифон дискетки, да и отправился в путь-дорогу. Близко ли, далеко, низко ли, высоко ли, вдруг видит Трифон – отель стоит на курьих ножках, а перед отелем человек горькими слезами заливается.

Спрашивает Трифон:

— О чем печалишься, мил человек?

— Да как мне не печалиться, коли задал мне хозяин задачу непосильную. Видишь ли ты перед отелем стоянку для автомобилей? Да видишь ли, как стоят автомобили?

Каждый вечер к отелю подъезжают автомобили, в которых находится по три человека. Гости останавливаются на ночлег, утром завтракают, да и опять отъезжают.

Хозяин дал указ: каждый автомобиль может занимать любое свободное место на стоянке, так как стоянка не размечена. Стоянка занимается полностью и в один ряд, а которым гостям не хватит места — пусть проезжают далее, автомобилей-то много больше, чем мест на стоянке.

Длина стоянки — восемь сажень, ширина каждого автомобиля — одна сажень. В каждом автомобиле равным счетом по три гостя. Дал мне хозяин 4550 руб. на год, стало быть на 365 дней, чтобы каждому гостю я обязательно приготовил завтрак ценою в 70 коп.

Указ я должен выполнить полностью, а не выполняю — не сносить мне головушки кудрявой.

Стал я думать думать: каждый день на стоянке может остановиться разное число автомобилей, но самое меньшее - четыре (иначе стоянка не будет заполнена), а самое большее - восемь. Стало быть, в среднем шесть автомобилей. За год их получится: $6 \cdot 365 = 2190$ шт. Ежели завтрак стоит 0.70 руб., да в каждом автомобиле по три гостя, то от хозяина я должен получить 4599 руб. И откуда же я возьму 49 руб.? Не могу выполнить указ, от того и печалюсь.

Тут Трифон и говорит:

— Не кручинься раньше времени, мил человек, ложись-ка спать. Утро вечера мудренее.

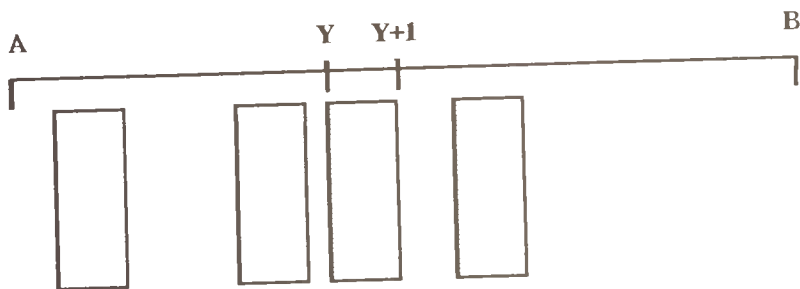
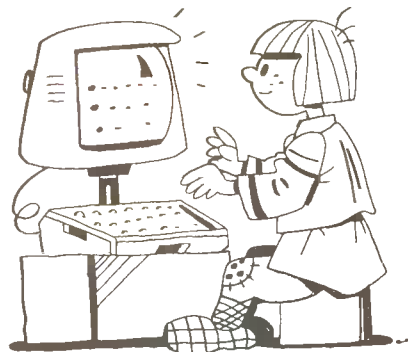
А когда солнце встало из-за леса, спрашивает Трифон:

— Мил человек, да есть ли в отеле персональный компьютер и транслятор с языка Паскаль?

— Да нешто мы не люди, — обиделся мил человек. — Да где это видано, чтобы в отелях компьютеры не водились?

Смекнул Трифон:

— Дай-ка я смоделирую весь процесс размещения автомобилей. Коли первый автомобиль занял случайно место на стоянке, то он разделил всю стоянку A,B на две части (A,Y) и (Y+1,B). Эти части не зависят друг от друга. Стало быть, каждую такую часть снова можно считать стоянкой, но меньшей длины. В каждой части можно опять попытаться разместить автомобиль. Ежели автомобиль не помещается — получается тривиальное решение, равное нулю, а если помещается, то процесс размещения продолжается.



— Белы светушки! — хлопнул себя по лбу Трифон. — Да ведь это же типичная рекурсия!

— И то правда, — поддакнул мил человек. — Да вот незатяе: коим же образом можно получать случайные значения Y?

Для этого надо воспользоваться стандартной функцией RAND, которая генерирует случайные числа с равномерным распределением в интервале (0,1). С помощью RAND можно получать значения в интервале (A,B):

$$Y = A + (B - A - 1) * \text{RAND},$$

где 1 — ширина автомобиля.

Да если тебе, мил человек, все понятно в моих рассуждениях, то опусти очи свои на программу, которая составлена на языке Паскаль. Программа состоит из раздела описаний, рекурсивной функции и раздела операторов основной программы.

```
(*-----*
! МОДЕЛИРОВАНИЕ МЕСТОПОЛОЖЕНИЯ АВТОМОБИЛЕЙ !
*-----*)
PROGRAM MODEL1(INPUT,OUTPUT);
  CONST N=365;      (* ЧИСЛО ДНЕЙ В ГОДУ *)
        RUBL=0.7;  (* СТОИМОСТЬ ЗАВТРАКА *)
        GOST=3;    (* ЧИСЛО ГОСТЕЙ В АВТОМ-ЛЕ *)

  VAR
    A,B:REAL;      (* РАЗМЕРЫ СТОЯНКИ *)
    SUM:REAL;      (* КО-ВО АВТОМ-ЛЕЙ ЗА ГОД *)
    DENG:REAL;     (* СУММА ДАНЕГ *)
    I,KOL:INTEGER; (* РАБОЧИЕ ПЕРЕМЕННЫЕ *)
```



```

      (* ФУНКЦИЯ "AUTO"
      -----*)
FUNCTION AUTO(A,B:REAL): INTEGER;
  VAR Y:REAL;
BEGIN
  IF (B-A)<1 THEN KOL:=0
  ELSE BEGIN
    Y:=A+(B-A-1)*RAND;
    KOL:=AUTO(A,Y)+AUTO(Y+1,B)+1
  END;
  AUTO:=KOL
END;

      (* ОСНОВНАЯ ПРОГРАММА
      -----*)
BEGIN
  SUM:=0;
  WRITELN('ВВЕДИТЕ ЗНАЧЕНИЯ А,В :');
  READLN(A,B);
  FOR I:=1 TO N DO
    SUM:=SUM+AUTO(A,B);
  WRITELN('ЧИСЛО АВТОМОБИЛЕЙ =' ,SUM:10:2);
  DENGА:=SUM*RUBL*GOST;
  WRITE('КОЛИЧЕСТВО ДЕНЕГ =' ,DENGА:10:2)
END.

```

1.  $Y=3,76$

2.  $Y=0,01$

3.  $Y=2,23$

4.  $Y=1,04$

5.  $Y=5,60$

6.  $Y=6,80$

— Верно ли я уразумел, — говорит мил человек, — что функция моделирует случайное размещение автомобилей в течение одного вечера: то ли это будет 5 автомобилей, то ли 8, то ли 6?

— Верно, мил человек, верно. Так что для подсчета всех автомобилей за год нужно повторить моделирование 365 раз, для чего и организован цикл FOR в основной программе.

Да ежели тебе все остальное понятно, то я запускаю программу на выполнение!

— Батюшки мои! — удивился мил человек. — Уже и ответ готов. Это что же мы видим на экране дисплея?! Автомобилей-то за год меньше, чем я считал в среднем. Стало быть, хозяин мне, напротив, дал лишку денег, вот радость-то какая! Однако, братец, меня и сомнения берут. Что ежели повторить моделирование заново, ну, скажем, с другим генератором случайных чисел, но тоже с равномерным распределением в интервале (0,1). Получу ли я тогда лишек денег?

— Непременно получишь, но, может, не-много меньше, а может, и больше.

И пригласил на радостях мил человек Трифона к завтраку царскому, а у самого глаза разгорелись и мысли побежали в разные стороны.

— А что, братец, — заговорил мил человек, — что ежели в каждом автомобиле будут сидеть не по три человека, а случайно от одного до пяти,

будет ли у меня еще лишек денег? Да будет ли к тому же еще, ежели ширина автомобиля будет тоже случайна от 0,5 до 1,5 сажень?

Задумался Трифон... Кто поможет ему?

Присказка. Если в данной версии языка Паскаль нет генератора случайных чисел, то его можно создать по схеме:

$$x_{i+1} = (ax_i + b) \bmod m; \text{ RAND} = x_{i+1}/m;$$

где x_0 , a , b — небольшие целые числа; m — как можно большое целое число; \bmod — определение целого остатка. Нужно следить, чтобы значение x было в диапазоне $0 \leq x < m$. Удачно подобрав x_0 , a , b , m , можно получить значения RAND с равномерным распределением в интервале $(0,1)$.



Действие последнее

ДОКА, ФОКА И КСЮША КИСЕЛЕВА

Картина единственная. ВОТ ТАК КСЮША КИСЕЛЕВА!

Ксюша. Здравствуйте, Дока и Фока! Помогите, пожалуйста, мне решить задачу.

Дока и Фока (в один голос). С удовольствием! Слушаем тебя внимательно, Ксюша.

Ксюша. Задача из болгарского математического фольклора:

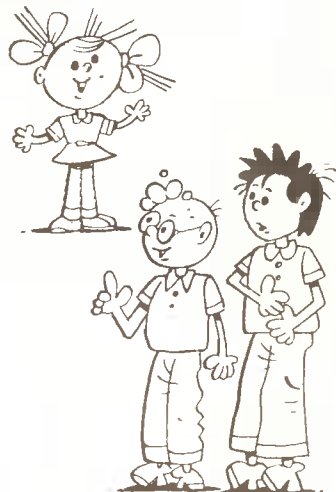
Мальчик и поросенок весят столько, сколько пять ящичков. Поросенок весит столько, сколько четыре кошки. Две кошки и поросенок весят столько, сколько три ящичка.

Сколько кошек уравновесят мальчика?


Дока и Фока (весело в один голос). Эту задачу можно решить устно, но мы сейчас быстренько составим программу на языке Паскаль и решим ее на компьютере.

Оба задумались...

И одновременно с большим удивлением в один голос:

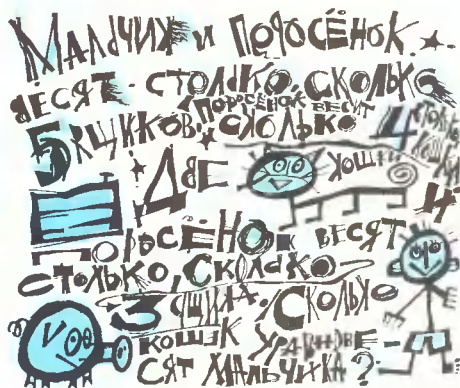


"А КАК СОСТАВИТЬ ПРОГРАММУ ЭТОЙ ЗАДАЧИ НА ЯЗЫКЕ ПАСКАЛЬ?"

(*А ЧТО ДУМАЕТЕ ВЫ, УВАЖАЕМЫЕ ЧИТАТЕЛИ?*)
(*FARE YOU WELL! ВСЕГО ХОРОШЕГО!*)
(*GOOD BYE! ДО СВИДАНИЯ!*)
END 

ЗРИ В КОРЕНЬ

Картина 1. ДИАЛОГ С ДВК



Дока. Мы рассмотрели составление самых разнообразных задач на языке Паскаль, но нам еще нужно научиться общаться с компьютером: как его подготовить к работе, как ввести программу и данные, как исправить ошибки и т.д.

Фока. Я с нетерпением жду этого момента! Давай скорее начнем решать задачи на ДВК!

Дока. Согласен! Приступаем...

Диалоговый вычислительный комплекс (ДВК) состоит из микроЭВМ, дисплея, дисководов с гибкими дисками и устройства печати. Как физическое устройство ДВК, так и его программное обеспечение непрерывно совершенствуются. Рассмотрим операционную систему ОС ДВК.

Для удобства работы на ДВК рекомендуется иметь по крайней мере два гибких диска:

системный, на котором записана операционная система, трансляторы с языков программирования и сервисные (обслуживающие) программы; рабочий, который используется для хранения информации пользователя (например, программ или каких-либо текстов).

Такое распределение является условным, так как допускается хранение любой информации на любом диске.

В операционной системе ОС ДВК для работы с языком Паскаль имеются две системные программы: PASCAL.SAV (транслятор с языка Паскаль) и PASCAL.OBJ (системная библиотека языка Паскаль). Как правило, первая из них располагается на системном диске, а вторая, из-за нехватки памяти, располагается на рабочем диске.

Для создания текста программы, а также для его корректировки (т.е. внесения различных изменений) предназначен экраный редактор текста. Широкое распространение получили редакторы: EDIK, EDIT, NED, K15, K52, SCREEN и др.

Наиболее простым для начинающих пользователей является редактор SCREEN, который входит в состав стандартного программного обеспечения ДВК и, как правило, располагается на системном диске. Полное имя редактора SCREEN.SAV, сокращенное — S.SAV (иногда SCR.SAV).

Однако возможности остальных редакторов текста значительно больше, поэтому именно они пользуются популярностью у программистов. Сначала мы рассмотрим основные функции редактора SCREEN, а затем в конце этого раздела – редакторов EDIK и NED.

Решение задачи на ЭВМ можно условно разбить на три этапа: подготовка ДВК к работе; создание и корректировка текста программы с помощью редактора текста; выполнение программы. Рассмотрим каждый этап в отдельности.

Подготовка ДВК к работе. Для подготовки ДВК к работе необходимо выполнить следующие действия:

1) ознакомиться с клавиатурой ДВК (см. Приложение 1);

2) включить электропитание устройств в следующем порядке: дисплей, дисководы, микроЭВМ.

На экране дисплея появляется символ @, означающий приглашение к началу работы. В случае сбоя повторить включение микроЭВМ. Будем считать, что магнитные диски физически готовы к работе, т.е. они отформатированы, инициализированы и на системном диске имеется стандартный набор программного обеспечения ОС ДВК;

3) вставить диски в дисководы. Нулевой (верхний) дисковод имеет обозначение (физическое имя) MX0, а первый (нижний) – имеет обозначение MX1. Можно вставлять любой диск в любой дисковод. Во всех обозначениях принят латинский регистр.

Пусть для определенности системный диск стоит в нулевом дисковом, а рабочий — в первом;

4) загрузить систему, для чего набрать символ X0, если системный диск расположен в нулевом дисковом, или набрать X1, если он расположен в первом дисковом.

В некоторых версиях ДВК загрузка имеет вид

В
MX0

После загрузки на экране появляется служебная информация и ожидается ввод даты;

5) ввести дату по образцу на экране и нажать клавишу возврата каретки ВК (в некоторых версиях ДВК клавишу ВВОД). Если дата не нужна, то можно просто нажать клавишу ВК.

На экран появляется имя рабочего дисковода MX1 или точка. Установился режим монитора, который позволяет вводить системные команды, т.е. команды монитора.

Все команды монитора набираются по верхнему латинскому регистру и не рекомендуется переключать регистры во время набора символов команды.

В случае сбоя или при необходимости выхода в монитор необходимо дважды одновременно нажать две клавиши

С	У
---	---

 и

Ц	С
---	---

при этом первую клавишу можно долго удерживать в нажатом положении, а вторую нажимать однократно.

П р и м е ч а н и е . Как правило, системное и рабочее устройства устанавливаются автоматически. Если системный диск находится в нулевом дисковом, то при загрузке X0 нулевой дисковод считается системным. Системное устройство имеет стандартное обозначение (логическое имя) SY.

Если рабочий диск находится в первом дисковом, то первый дисковод считается рабочим. Рабочее устройство имеет логическое имя DK.

Если рабочее устройство автоматически не устанавливается (на экране вместо MX1 появляется MX0), то необходимо набрать команду монитора

ASS _ MX1 _ DK BK

после которой на экране появится MX1 и все устройства полностью готовы к работе.

б) Посмотреть каталог (оглавление) дисков. Команда

DIR _ MX0 : BK

предназначена для просмотра каталога диска в нулевом дисковом. Так как в нашем случае в нулевом дисковом находится системный диск, то необходимо убедиться в наличии транслятора с языка Паскаль PASCAL.SAV и редактора текста S.SAV (возможно, SCR.SAV). Команда

DIR _ MX1: BK

предназначена для просмотра каталога диска в первом дисковом. Поскольку в нашем случае в первом дисковом находится рабочий диск, то необходимо убедиться в наличии системной библиотеки PASCAL.OBJ и свободной памяти для программы.

Управление движением и остановом информации на экране в режиме команд монитора происходит следующим образом:

для останова движущихся строк необходимо одновременно нажать клавиши

CY и C
S

для продолжения просмотра необходимо одновременно нажать клавиши

CY и Я
Q

После появления на экране MX1 или точки можно переходить к новому режиму.

Создание и корректировка файла программы с помощью редактора текста SCREEN. В операционной системе ОС ДБК информация, хранящаяся на диске, представлена в виде файлов. Под файлом понимается произвольный набор данных. Так, любая системная программа является файлом. В частности, можно создавать файл программы, файл исходных данных, файл какого-либо текста. Каждый файл обозначается именем. Имя может содержать не более шести любых символов. Оно может совпадать и не совпадать с именем программы на языке Паскаль после слова PROGRAM.

Для вызова любого файла, расположенного на системном диске, используется команда

R _ имя файла BK

Для вызова любого файла, расположенного на рабочем диске, используется команда

RUN _ имя файла BK

Рассмотрим основные сведения при создании и корректировке файла программы с помощью редактора SCREEN (или просто S). Для удобства изложения обозначим файл программы именем PRIM (полное имя PRIM.PAS).

Порядок действий для создания файла впервые (т.е. данная программа не была записана на диске) представлен на рис.3. Каждая строка текста программы заканчивается нажатием клавиши BK.

! ПОРЯДОК ДЕЙСТВИЙ !	! ПОЯСНЕНИЯ !
! .R S [BK]	! ВЫЗОВ РЕДАКТОРА SCREEN !
! ВХОДНОЙ ФАЙЛ> [BK]	! ВХОДНОГО ФАЙЛА НЕТ !
! ВЫХОДНОЙ ФАЙЛ> PRIM [BK]	! ИМЯ СОЗДАВАЕМОГО ФАЙЛА !
! !	! ИНФОРМАЦИЯ ПОЛЬЗОВАТЕЛЯ !
! ЭКРАН ЧИСТ, НАБИРАЙТЕ !	! !
! ТЕКСТ ПРОГРАММЫ !	! !
! !	! !
! ОДНОКРАТНО ПОСЛЕДОВАТЕЛЬНО !	! !
! НАЖМИТЕ КЛАВИШИ: !	! !
! [↑] [0] !	! КОНЕЦ РАБОТЫ РЕДАКТОРА !
! !	! И ЗАПИСЬ ФАЙЛА НА ДИСК !







Рис. 3. Создание файла с помощью редактора SCREEN

В дальнейшем для внесения различных изменений в созданную программу необходимо пользоваться режимом корректировки файла (рис.4).

! ПОРЯДОК ДЕЙСТВИЙ !	! ПОЯСНЕНИЯ !
! .R S [BK]	! ВЫЗОВ РЕДАКТОРА SCREEN !
! ВХОДНОЙ ФАЙЛ> PRIM [BK]	! ИМЯ ВЫЗЫВАЕМОГО ФАЙЛА !
! ВЫХОДНОЙ ФАЙЛ> [BK]	! ИМЯ ФАЙЛА СОХРАНЯЕТСЯ !
! СКОЛЬКО СТРОК ПРОПУСТИТЬ> 0 [BK]	! ПРОСМОТР ФАЙЛА С ПЕРВОЙ !
! !	! СТРОКИ !
! НА ЭКРАНЕ ПРОГРАММА, !	! ИНФОРМАЦИЯ ПОЛЬЗОВАТЕЛЯ !
! КОРРЕКТИРУЙТЕ ТЕКСТ !	! !
! !	! !
! ОДНОКРАТНО ПОСЛЕДОВАТЕЛЬНО !	! !
! НАЖМИТЕ КЛАВИШИ: !	! !
! [↑] [0] !	! КОНЕЦ РАБОТЫ РЕДАКТОРА !
! !	! И ЗАПИСЬ ФАЙЛА НА ДИСК !

Рис. 4. Корректировка файла с помощью редактора SCREEN

Любые изменения текста программы выполняются с помощью маркера (курсора), который можно перемещать в нужную позицию текста, используя клавиши управления (см. Приложение 1):

-  перемещение маркера вправо;
-  перемещение маркера влево;
-  перемещение маркера вверх;
-  перемещение маркера вниз;
-  перемещение маркера в левую верхнюю позицию экрана;
-  сдвиг строки (части строки) вправо от маркера (используется для вставки символов внутри строки);

- ← сдвиг строки (части строки) влево к маркеру (используется для удаления символов внутри строки);
- ↓ вставка новой строки (строка, под которой стоит маркер, сдвигается вниз и появляется пустая строка);
- ↑ удаление всей строки (маркер стоит под удаляемой строкой, на месте удаленной строки будет следующая за ней);
- ↓ для просмотра текста от начала к концу нажать и удерживать клавишу;
- ↑ для просмотра текста от конца к началу нажать и удерживать клавишу;
- ЗБ стирание предыдущего символа;
- ВК установка маркера в начало новой строки.

Выполнение программы. После создания файла программы на языке Паскаль необходимо оттранслировать, скомпоновать и запустить программу на выполнение. Эта последовательность действий отражена на рис.5, где имя файла программы обозначено через PRIM.

ПОРЯДОК ДЕЙСТВИЙ		ПОЯСНЕНИЯ
R PASCAL	[ВК]	ТРАНСЛЯЦИЯ С ПАСКАЛЯ НА
*PRIM, TT:=PRIM	[ВК]	МАКРОАССЕМБЛЕР;
R MACRO	[ВК]	ТРАНСЛЯЦИЯ С МАКРОАССЕМБЛЕРА
*PRIM=PRIM	[ВК]	В ОБЪЕКТНЫЕ КОДЫ;
*НАЖАТЬ КЛАВИШУ [СУ, Ч/С]		ВЫХОД В МОНИТОР;
R LINK	[ВК]	КОМПОНОВКА С СИСТЕМНОЙ
*PRIM=PRIM,PASCAL	[ВК]	БИБЛИОТЕКОЙ PASCAL.OBJ ;
*НАЖАТЬ КЛАВИШУ [СУ, Ч/С]		ВЫХОД В МОНИТОР;
RUN PRIM	[ВК]	ЗАПУСК ПРОГРАММЫ НА
(ВВОД ДАННЫХ ПО ОПЕРАТОРУ READ)		ВЫПОЛНЕНИЕ;

Рис. 5. Протокол выполнения программы

Пояснения к протоколу.

1. Команда R _ PASCAL означает вызов транслятора с языка Паскаль (PASCAL.SAV). После появления звездочки * на экране необходимо указать строку

PRIM, TT:= PRIM

Здесь справа от знака равенства указывается имя входного файла (на языке Паскаль), а слева — имя выходного файла (на языке Макроассемблера). Выходной файл можно обозначить иначе. Для вывода на экран листинга программы вместе с диагностическими сообщениями указывается TT:

Признак ошибки обозначается звездочками в левой части экрана и символом "—" на том месте оператора программы, где допущена ошибка и где после этого встречается точка с запятой. Вслед за этим выдается сообщение о виде ошибки и просмотр листинга продолжается. В конце листинга сообщается общее число ошибок, например при наличии двух ошибок

ERRORS DETECTED : 2

Управление просмотром листинга происходит следующим образом:

для останова просмотра листинга необходимо одновременно нажать клавиши

СУ

и

С
S

для продолжения просмотра необходимо одновременно нажать клавиши

СУ

и

Я
Q

Для повторного просмотра листинга нужно снова выполнить трансляцию с языка Паскаль.

При наличии ошибок трансляции необходимо в режиме корректировки исправить все ошибки (см.рис.4), затем вновь выполнить трансляцию с языка Паскаль.

Этот процесс продолжается до тех пор, пока не будут исправлены все ошибки, после чего на экране появится сообщение

ERRORS DETECTED: 0

2. При отсутствии ошибок трансляции с языка Паскаль можно переходить к трансляции с языка Макроассемблер в объектные коды.

Команда R _ MACRO означает вызов транслятора с языка Макроассемблер. После появления звездочки на экране указывается строка

PRIM = PRIM

где справа от знака равенства указывается имя входного файла (на языке Макроассемблера), а слева — имя выходного файла (в объектных кодах). Как только файл в объектных кодах готов, на экране появляется сообщение об ошибках

ERRORS DETECTED: 0

Если после трансляции с языка Паскаль не было ошибок, а после трансляции с Макроассемблера они появились, то это чаще всего означает, что не хватает памяти на диске для выходного файла. На экране вновь появляется звездочка и для выхода в монитор необходимо одновременно нажать две клавиши

СУ

и

Ц
С

3. При отсутствии ошибок в объектных кодах можно вызывать компоновщик (редактор связей) командой R _ LINK. После появления звездочки указывается строка

PRIM = PRIM, PASCAL

где справа от знака равенства компонуются в единый модуль файл в объектных кодах с системной библиотекой языка Паскаль PASCAL.OBJ. Получаемый слева от знака равенства файл готов к выполнению. В общем случае в строке справа от знака равенства можно указывать через запятую несколько файлов, все они будут объединяться в один модуль. Если PASCAL.OBJ записан не на рабочем диске, а на системном, то нужно указать это строкой

PRIM = PRIM,SY: PASCAL

По окончании компоновки на экране появляется звездочка, после чего можно выходить в монитор для перехода к новому режиму.

4. Запуск программы на выполнение осуществляется командой

RUN _ PRIM

Если в программе на языке Паскаль были операторы ввода READ, то необходимо ввести с клавиатуры исходные данные в полном соответствии с операторами ввода.

Замечания:

а) если необходимо повторить решение задачи при новых исходных данных, не изменяя текста программы, достаточно вновь запустить программу на выполнение командой

RUN _ PRIM

б) если необходимо изменить текст программы, то нужно в режиме редактирования (см.рис.4) внести все изменения и повторить протокол выполнения программы (см.рис.5).

Картина 2. НАКОНЕЦ-ТО ПОЛУЧУ РЕЗУЛЬТАТ!

Дока. Вот сейчас, Фока, мы подошли к самому главному моменту: решим задачу на ДВК и получим результат.

Фока. Ну наконец-то, я получу результат!

Задача 62. Вычислить значения F и R, если

$$F = \sqrt{N}, R = \Delta^2.$$

Переменные N и D принимают следующие значения:

- а) N = 49, D = 2,5;
- б) N = -49, D = 2,5;
- в) N = 1125, D = 16;
- г) N = 6,25 (R не вычислять).

Программа на языке Паскаль имеет вид:

```
(*-----*  
!                                     !  
-----*  
PROGRAM PRIM(INPUT,OUTPUT);  
  VAR  
    F,N,R,D:REAL;      (* ДЕЙСТВИТЕЛЬНЫЕ ДАННЫЕ *)  
BEGIN  
  WRITELN('ВВЕДИТЕ ЧЕРЕЗ ПРОБЕЛ ЗНАЧЕНИЯ N,D, :');  
  READ(N,D);  
  F:=SQRT(N);  
  R:=SQR(D);  
  WRITELN('РЕЗУЛЬТАТЫ ВЫЧИСЛЕНИЙ :');  
  WRITE('F=', F:5:1, ' ':5, 'R=', R:6:2);  
END.
```

Методика решения задачи.

1. С помощью редактора текста создать файл программы с именем PRIM и записать его на диск (см. рис.3).

2. Выполнить трансляцию с языка Паскаль (см.рис.5). Если на экране появилось сообщение об ошибках, то необходимо исправить ошибки в режиме корректировки (см. рис.4) и вновь повторить трансляцию с языка Паскаль.

При отсутствии ошибок трансляции можно выполнять весь протокол (см. рис.5),

3. После запуска программы на выполнение командой

RUN _ PRIM BK

нужно ввести исходные данные N = 49 и D = 2,5 в виде строки

49 _ 2.5 BK

и затем проверить правильность полученных результатов:

```
-----
!      ВВЕДИТЕ ЧЕРЕЗ ПРОБЕЛ ЗНАЧЕНИЯ N,D :      !
!      49 2.5                                     !
!      РЕЗУЛЬТАТЫ ВЫЧИСЛЕНИЙ :                   !
!      F= 7.0      R= 6.25                         !
-----
```

4. Повторить решение задачи при исходных данных $N=-49$, $D=2,5$. Для этого достаточно ввести команду

RUN _ PRIM BK

и данные в виде

-49 _ 2.5 BK

Поскольку машина не может извлечь корень квадратный из отрицательного числа, то выдается сообщение:

```
-----
!      ВВЕДИТЕ ЧЕРЕЗ ПРОБЕЛ ЗНАЧЕНИЯ N,D :      !
!      -49 2.5                                    !
!      SQRT OF NEGATIVE                          !
-----
```

5. Повторить решение задачи при новых исходных данных $N=1225$ и $D=16$. Для этого достаточно ввести команду

RUN _ PRIM BK

и данные в виде

1225 _ 16 BK

затем убедиться в правильности полученных результатов:

```
-----
!      ВВЕДИТЕ ЧЕРЕЗ ПРОБЕЛ ЗНАЧЕНИЯ N,D :      !
!      1225 16                                    !
!      РЕЗУЛЬТАТЫ ВЫЧИСЛЕНИЙ :                   !
!      F= 35.0      R= 256.00                      !
-----
```

6. В режиме корректировки (см. рис.4) изменить программу следующим образом: удалить вычисление значения R. Для этого надо удалить в разделе описания переменные R и D, при вводе удалить D, всю строку вычисления R, а при выводе удалить R. В результате получится программа:

```
(*-----
!                                     ПРИМЕР                                     !
!-----*)
PROGRAM PRIM(INPUT,OUTPUT);
  VAR
    F,N:REAL;      (* ДЕЙСТВИТЕЛЬНЫЕ ДАННЫЕ *)
BEGIN
  WRITELN('ВВЕДИТЕ ЗНАЧЕНИЕ N : ');
  READ(N);
  F:=SQRT(N);
  WRITELN('РЕЗУЛЬТАТЫ ВЫЧИСЛЕНИЙ : ');
  WRITE('F=', F:5:1 );
END.
```


Поскольку программа изменилась (в отличие от предыдущих пунктов, где менялись только исходные данные), то необходимо вновь повторить протокол выполнения (см.рис.5). После запуска программы на выполнение ввести одно значение N=6,25 в виде

6.25

и убедиться в правильности полученных результатов:

```

-----
!   ВВЕДИТЕ ЗНАЧЕНИЕ N :           !
!   6.25                         !
!   РЕЗУЛЬТАТЫ ВЫЧИСЛЕНИЙ :      !
!   F=  2.5                      !
-----

```

7. По окончании решения задачи на ЭВМ получено четыре файла:

PRIM.PAS — файл исходного текста на языке Паскаль;

PRIM.MAC — файл программы на языке Макроассемблер;

PRIM.OBJ — файл в объектных кодах;

PRIM.SAV — загрузочный файл для запуска программы на выполнение.

Кроме того, при корректировке программы в редакторе сохраняется старый файл (PRIM.BAK).

Необходимо убедиться, что на рабочем диске имеются все файлы, для чего посмотреть каталог диска командой

DIR _ MX1:

П р и м е ч а н и е. При создании и корректировке файла программы на языке Паскаль наиболее часто встречаются ошибки (не видимые на экране) из-за излишнего переключения русского и латинского регистров. Не разрешается переключать регистры внутри слов, составных знаков операций, числе и т.д. Регистры переключаются там, где допускается пробел, например:

SUMMA СУММА

(* КОММЕНТАРИЙ *)

WRITE (' СУММА = ',S:10);

<=.

Если константа имеет своим значением русское слово (например, в операторе CASE, в перечисляемом типе и т.д.), то запрещается использовать буквы Ч, Ю, Щ, Ъ.

Другая особенность ДВК связана с вводом символьных данных при четырехрегистравом режиме клавиатуры: нижнем регистре НР, верхнем ВР, русском РУС и латинском ЛАТ. При нажатии клавиш русского или латинского регистра во время ввода символьных данных, их коды (16В и 17В, см.Приложение 2) будут введены в память ЭВМ, хотя они на экране дисплея никак не отражаются.

Пусть, например, с помощью оператора READ (S1, S2) символьная переменная S1 должна получить значение '+', а переменная S2 — значение 'A'. Однако при нажатии клавиши ЛАТ перед вводимыми данными

+ A

переменная S1 получит значение кода латинского регистра, а переменная S2 — значение '+’.

В общем случае при вводе символьных данных за счет переключения регистров общее число символов будет увеличено. Например, при вводе строки из 12 символов (включая пробел)

ШINS+R _ ZБПNRW

фактически строка состоит из 16 символов

РУС 1 Ш 2 ЛАТ 3 N 4 S 5 + 6 R 7 _ 8 Z 9 РУС 10 Б 11 П 12 ЛАТ 13 N 14 R 15 W 16

Чтобы избавиться от этого недостатка, рекомендуется проверять каждый вводимый символ: если он является кодом латинского или русского регистра, то ввод повторяется до тех пор, пока не будет введен заданный символ. При этом код русского или латинского регистра переводится в соответствующий ему символьный тип с помощью встроенной функции CHR (см. Приложение 5):

CHR(14) и CHR(15)

По этой же причине не допускается переключение регистров внутри символьной константы между апострофами. Примеры правильного набора символьных констант:

ЛАТ 'L'; РУС 'П'.

Картина 3. ПОЙДУ К ДРУГОМУ РЕДАКТОРУ

Создание и корректировка файла программы с помощью редакторов EDIK и NED. Редактор текста EDIK (или просто E) обладает большими возможностями по сравнению с редактором SCREEN. Для удобства изложения обозначим файл программы именем PRIM (полное имя PRIM.PAS).

Порядок действий для создания файла впервые (т.е. данная программа не была записана на диске) представлен на рис.6. В дальнейшем для внесения различных изменений в созданную программу необходимо пользоваться режимом корректировки файла (рис.7).

!	ПОРЯДОК ДЕЙСТВИЙ	!	ПОЯСНЕНИЯ	!
!	1. R EDIK [BK]	!	ВЫЗОВ РЕДАКТОРА EDIK	!
!	2. *PRIM.PAS/C [BK]	!	ИМЯ СОЗДАВАЕМОГО ФАЙЛА	!
!	! ЭКРАН ЧИСТ.	!	ИНФОРМАЦИЯ ПОЛЬЗОВАТЕЛЯ	!
!	! НАБИРАЙТЕ ТЕКСТ	!		!
!	3. ОДНОКРАТНО ПОСЛЕДОВА-	!		!
!	ТЕЛЬНО НАЖМИТЕ КЛАВИШИ:	!		!
!	[ПРЕФИКС], [E]	!	КОНЕЦ РАБОТЫ С ФАЙЛОМ	!
!	EXIT? [BK]	!	И ЗАПИСЬ ФАЙЛА НА ДИСК	!
!	4. * [СУ, Ц/С]	!	ВЫХОД ИЗ РЕДАКТОРА	!

Рис. 6. Создание файла с помощью редактора EDIK

ПОРЯДОК ДЕЙСТВИЯ	ПОЯСНЕНИЯ
1. R EDIK [BK]	ВЫЗОВ РЕДАКТОРА EDIK
2. * PRIM.PAS [BK]	ИМЯ КОРРЕКТИРУЕМОГО ФАЙЛА
НА ЭКРАНЕ ТЕКСТ, КОРРЕКТИРУЙТЕ ЕГО	ИНФОРМАЦИЯ ПОЛЬЗОВАТЕЛЯ
3. ОДНОКРАТНО ПОСЛЕДОВА- ТЕЛЬНО НАЖМИТЕ КЛАВИШИ:	
[ПРЕФИКС], [E]	КОНЕЦ РАБОТЫ С ФАЙЛОМ
EXIT? [BK]	И ЗАПИСЬ ФАЙЛА НА ДИСК
4. * [СУ, Ц/С]	ВЫХОД ИЗ РЕДАКТОРА









Рис. 7. Корректировка файла с помощью редактора EDIK


















После вызова редактора командой R_E, на экране дисплея появляться звездочка *, что означает запрос о файле.

Для создания нового файла нужно указать полное имя файла с ключом "/С". Затем набрать текст программы, заканчивая каждую строку текста нажатием клавиши возврата каретки BK. Чтобы записать файл на диск, необходимо нажать клавиши префикса " \ " и символа "E", а после появления вопроса "Exit?" нужно нажать клавишу BK. Файл запишется на диск и на экране вновь появится звездочка — запрос о файле, создаваемом или существующем. Если редактор больше не нужен, то можно выйти из него, нажав клавиши выхода в монитор.

Для корректировки существующего файла необходимо после появления звездочки указать имя файла, после чего на экране появится текст программы. Нужно откорректировать текст и снова записать программу на диск. На экране вновь появится звездочка — запрос о файле. Если редактор больше не нужен, то можно выйти в монитор, а если еще нужен, то указать имя файла (создаваемого или существующего). Допускается выход из редактора без записи файла на диск. В этом случае вместо символа "E" указывается символ "Q".

Любые изменения текста программы выполняются с помощью маркера (курсора), который можно перемещать в нужную позицию текста, используя клавиши управления (см. Приложение 1):

-  перемещение маркера вправо;
-  перемещение маркера влево;
-  перемещение маркера вверх;
-  перемещение маркера вниз;
-  сдвиг строки (части строки) влево к маркеру, часто используется для удаления символов внутри строки;
-  пробел, сдвиг строки (части строки) вправо от маркера;
-  удаление всей строки (маркер стоит под удаляемой строкой, на месте удаленной строки будет следующая за ней);
-  вставка новой строки (строка, под которой стоит маркер, сдвигается вниз и появляется пустая строка, на место которой можно ввести новую строку);













-  просмотр текста от начала к концу (нажать и удерживать клавишу);
-  просмотр текста от конца к началу (нажать и удерживать клавишу);
-  клавиша не используется (при случайном нажатии ее необходимо для продолжения работы нажать одновременно клавиши  );
-  стирание предыдущего символа;
-  ввод новой строки или перенос остатка строки (справа от маркера) на новую строку;
-  установка маркера в начало новой строки;
-  префикс;
-   перемещение маркера в конец текста;
-   перемещение маркера в начало текста;
-   перемещение маркера в начало строки;
-   перемещение маркера в конец строки.

Основные функции редактора NED для создания и корректировки файла программы практически полностью совпадают с редактором EDIK, за исключением:

вместо редактора EDIK вызывается редактор NED;
клавишей префикс является 

Отличаются редакторы более сложными своими функциями, которые мы здесь не рассматриваем.

Приложение 1. КЛАВИАТУРА ДВК

Алфавитно-цифровая клавиатура:

ВР — верхний регистр,
НР — нижний регистр,
ЛАТ — латинский регистр,
РУС — русский регистр.

;	!	"	#	¤	%	&	/	()	0	=	ТАБ	ГТ
Й	Ц	У	К	Е	Н	Г	Ш	Щ	З	Х	:	ПС	ВК
Ј	С	У	К	Е	Н	Г	Ш	Щ	З	Х	:		
СУ	Ф	Ы	В	А	Р	О	Л	Д	Ж	Э	>	ЗБ	
	Ф	У	В	А	Р	О	Л	Д	Ж	Э	>		
ВР	Я	Ч	С	М	И	Т	Ь	Б	Ю	<	/		НР
	Q	Ч	С	М	И	Т	Ь	Б	Ю	<	/		
РУС	ПРОБЕЛ												ЛАТ

Клавиатура управления маркером

Приложение 2. УПОРЯДОЧЕННАЯ ПОСЛЕДОВАТЕЛЬНОСТЬ СИМВОЛОВ (КОДЫ КОИ-7)

Восьмеричный код	Номер символа	Обозначение	Наименование
040	32	—	Пробел
041	33	!	Восклицательный знак
042	34	"	Кавычки
043	35	#	Номер
044	36	¤	Знак денежной единицы
045	37	%	Процент
046	38	&	Коммерческое "и"
047	39	'	Апостроф
050	40	(Левая круглая скобка
051	41)	Правая круглая скобка
052	42	*	Звездочка
053	43	+	Плюс
054	44	,	Запятая
055	45	-	Минус
056	46	.	Точка
057	47	/	Наклонная черта
060	48	0	
061	49	1	
062	50	2	
063	51	3	
064	52	4	
065	53	5	
066	54	6	
067	55	7	
070	56	8	
071	57	9	

Восьмеричный код	Номер символа	Обозначение	Наименование
072	58	:	Двоеточие
073	59	;	Точка с запятой
074	60	<	Меньше
075	61	=	Равно
076	62	>	Больше
077	63	?	Вопросительный знак
078	64	@	Коммерческое "ЭТ"
101	65	A	
102	66	B	
103	67	C	
104	68	D	
105	69	E	
106	70	F	
107	71	G	
110	72	H	
111	73	I	
112	74	J	
113	75	K	
114	76	L	
115	77	M	
116	78	N	
117	79	O	
120	80	P	
121	81	Q	
122	82	R	
123	83	S	
124	84	T	
125	85	U	
126	86	V	
127	87	W	
130	88	X	
131	89	Y	
132	90	Z	
133	91	[Левая скобка
134	92	\	
135	93]	Правая скобка
136	94	¬	Отрицание
137	95	_	Подчеркивание
140	96	Ю	
141	97	А	
142	98	Б	
143	99	Ц	

Восьмеричный код	Номер символа	Обозначение	Наименование
144	100	Д	
145	101	Е	
146	102	Ф	
147	103	Г	
150	104	Х	
151	105	И	
152	106	Й	
153	107	К	
154	108	Л	
155	109	М	
156	110	Н	
160	112	П	
161	113	Я	
162	114	Р	
163	115	С	
164	116	Т	
165	117	У	
166	118	Ж	
167	119	В	
170	120	Ь	
171	121	Ы	
172	122	З	
173	123	Ш	
174	124	Э	
175	125	Щ	
176	126	Ч	
177	127	ЗБ	Забой
Дополнительные символы:			
015	13	ВК	Возврат каретки
016	14	РУС	Русский регистр
017	15	ЛАТ	Латинский регистр

Приложение 3. СЛУЖЕБНЫЕ СЛОВА ЯЗЫКА ПАСКАЛЬ

AND, ARRAY, BEGIN, CASE, CONST, DIV, DO, DOWNT0, ELSE, END, FILE, FOR, FUNCTION, GOTO, IF, IN, LABEL, MOD, NIL, NOT, OF, OR, PACKED, PROCEDURE, PROGRAM, RECORD, REPEAT, SET, THEN, TO, TYPE, UNTIL, VAR, WHILE, WITH.

Приложение 4. СТАНДАРТНЫЕ ИМЕНА

ABS, ARCTAN, BOOLEAN, CHAR, CHR, COS, CLOSE, DISPOSE, EOF, EOLN, EXP, FALSE, FORWARD, GET, INPUT, INTEGER, LN, MAXINT, NEW, ODD, ORD, OUTPUT, PACK, PAGE, PRED, PUT, READ, READLN, REAL, RESET, REWRITE, ROUND, SIN, SQR, SQRT, SUCC, TEXT, TRUE, TRUNC, UNPACK, WRITE, WRITELN.

Приложение 5. ТАБЛИЦА ОСНОВНЫХ ВСТРОЕННЫХ ФУНКЦИЙ И НЕКОТОРЫХ ОПЕРАЦИЙ

Функция	Назначение	Тип аргумента	Тип функции
ABS(X)	Вычисление абсолютного значения X	REAL INTEGER	REAL INTEGER
SQR(X)	Вычисление квадрата X (X * X)	REAL INTEGER	REAL INTEGER
SIN(X)	Вычисление синуса X	REAL INTEGER	REAL REAL
COS(X)	Вычисление косинуса X	REAL INTEGER	REAL REAL
ARCTAN(X)	Вычисление арктангенса X	REAL INTEGER	REAL REAL
EXP(X)	Вычисление экспоненты (числа E) в степени X	REAL INTEGER	REAL REAL
EXP10(X)	Вычисление 10 в степени X	REAL INTEGER	REAL REAL
LN(X)	Вычисление натурального логарифма X	REAL INTEGER	REAL REAL
LOG(X)	Вычисление десятичного логарифма X	REAL INTEGER	REAL REAL
SQRT(X)	Вычисление квадратного корня из X	REAL INTEGER	REAL REAL
A DIV B	Вычисление частного при делении A на B	INTEGER	INTEGER
A MOD B	Нахождение остатка при делении A на B нацело	INTEGER	INTEGER
TRUNC(X)	Нахождение целой части X	REAL INTEGER	INTEGER INTEGER
ROUND(X)	Округление X в сторону ближайшего целого	REAL INTEGER	INTEGER INTEGER
ODD(X)	TRUE, если X - нечетное FALSE, если X - четное	INTEGER	BOOLEAN
ORD(X)	1. Нахождение порядкового номера элемента X	BOOLEAN Перечислимый	INTEGER INTEGER
	2. ORD(X) = X	INTEGER	INTEGER
	3. Определение номера символа языка Паскаль в десятичной системе счисления	CHAR	INTEGER
CHR(X)	Определение символа языка Паскаль по его порядковому номеру	INTEGER	CHAR
SUCC(X)	Нахождение элемента, являющегося следующим для данного в перечне допустимых элементов	INTEGER BOOLEAN CHAR	INTEGER BOOLEAN CHAR
PRED(X)	Нахождение элемента, являющегося предыдущим для данного в перечне допустимых элементов	Перечислимый	Перечислимый
FOF(X)	TRUE, если файл F находится в состоянии "конец файла" FALSE, если нет конца файла	Файловый	BOOLEAN
EOIN(X)	TRUE, если файл F находится в состоянии "конец строки" FALSE, если нет конца строки	Файловый	BOOLEAN
EOLN	TRUE, если при вводе с клавиатуры нажата клавиша <BK> FALSE, если при вводе с клавиатуры не нажата клавиша <BK>		BOOLEAN

СПИСОК ЛИТЕРАТУРЫ

1. Тарасов Б. Паскаль. Серия "Жизнь замечательных людей". — М.: Молодая гвардия, 1982.
2. Горстко А.Б., Кочкова С.В. Азбука программирования. — М.: Знание, 1988.
3. Семашко Г.Л., Салтыков А.И. Программирование на языке Паскаль. — М.: Наука, 1988.
4. Шаньгин В.Ф., Поддубный Л.М., Голубев-Новожилов Ю.С. Программирование на языке Паскаль. Программное обеспечение микроЭВМ. — Кн. 7. — М.: Высшая школа, 1988.
5. Бутомо И.Д., Самочадин А.В., Усанова Д.В. Программирование на алгоритмическом языке Паскаль для микроЭВМ. — Л.: Изд-во ЛГУ, 1985.
6. Грэхем Р. Практический курс языка Паскаль для микроЭВМ. — М.: Радио и связь, 1986.
7. Вирт Н. Алгоритмы + Структуры данных — Программы. — М.: Мир, 1985.
8. Прайс Д. Программирование на языке Паскаль. Практическое руководство. — М.: Мир, 1985.
9. Уилсон И.Р., Эддиман А.М. Практическое введение в Паскаль. — М.: Радио и связь, 1983.
10. Форсайт Р. Паскаль для всех. — М.: Машиностроение, 1986.
11. Перминов О.Н. Программирование на языке Паскаль. — М.: Радио и связь, 1988.
12. Медведев А. Поговорки/ Информатика и образование. — № 4. — М.: Педагогика, 1988.
13. Абрамов С.А., Зима Е.В. Начала программирования на языке Паскаль. — М.: Наука, 1987.
14. Йейсен К., Вирт Н. Паскаль. Руководство для пользователя и описание языка. — М.: Финансы и статистика, 1982.

ОГЛАВЛЕНИЕ

Предисловие	3
Действующие лица	4
Действие первое. ДОКА, ФОКА И ПАСКАЛЬ	5
Картина 1. Встреча друзей	5
Общие сведения о языке Паскаль. Пример простой программы.	
Картина 2. Забегая вперед	11
Несколько интересных задач с программами и ответами.	
Картина 3. Кирпичики, из которых строится здание	16
Основные символы и определения языка.	
Картина 4. Театральная афиша	18
Структура программы. Требования к стилю программирования. Примеры.	
Картина 5. Действующие данные	22
Константы и переменные. Их описание в программе.	
Действие второе. ДАЙТЕ МНЕ ТОЧКУ ОПОРЫ	24
Картина 1. Как нужно правильно выражаться?	24
Понятие выражения. Правила записи выражений.	
Картина 2. Зачем изобретать велосипед?	25
Стандартные функции.	
Картина 3. Точка опоры	26
Оператор присваивания.	
Картина 4. То густо, то пусто	27
Составной и пустой операторы. Назначение точки с запятой.	
Картина 5. Диалог с компьютером	29
Операторы ввода и вывода.	
Действие третье. ВИЗИТНАЯ КАРТОЧКА ДАННЫХ	35
Картина 1. Хороший или плохой характер у данных?	35
Типы данных в языке Паскаль.	
Картина 2. Данные целого типа	36
Описание данных, операции над ними. Ввод и вывод. Задачи.	
Картина 3. Данные действительного типа	39
Описание данных, операции над ними. Ввод и вывод.	
Картина 4. Два плюс три	43
Арифметические выражения. Правила записи. Задачи.	
Картина 5. Данные логического типа	47
Описание данных, операции над ними. Ввод и вывод.	
Картина 6. Истина или ложь?	49
Логические выражения. Задачи.	
Картина 7. Данные символьного типа	51
Описание данных, операции над ними. Ввод и вывод. Задачи.	
Действие четвертое. ПОЙДЕТ НАПРАВО — ПЕСНЬ ЗАВОДИТ, НАЛЕВО — СКАЗКУ ГОВОРИТ	57
Картина 1. Любишь кататься, люби и саночки возить	57
Ветвление. Условный оператор. Задачи.	

Картина 2.	Право выбора Оператор выбора. Задачи.	63
Картина 3.	Куда пойти? Оператор перехода. Примеры.	65
Действие пятое. БЕЛКА В КОЛЕСЕ		68
Картина 1.	Цикл, цикл, цикл Понятие цикла. Виды оператора цикла в языке Паскаль.	68
Картина 2.	Проверь — потом делай Оператор цикла с предварительным условием. Задачи.	68
Картина 3.	Сначала сделай — потом проверь Оператор цикла с последующим условием. Задачи.	74
Картина 4.	Делай ровно столько — сколько задано Оператор цикла с параметром. Задачи.	76
Картина 5.	Сколько матрешек в матрешке? Вложенные циклы, их структура. Задачи.	81
Действие шестое. НЕСТАНДАРТНЫЕ ТИПЫ		84
Картина 1.	Еще раз о типах Понятие о простых нестандартных типах.	84
Картина 2.	Перечисляемый тип Описание перечисляемого типа данных. Действия над данными. Задачи.	84
Картина 3.	Ограниченный тип Описание ограниченного типа данных и их применение.	87
Действие седьмое. СЛОЖНЫЕ ТИПЫ		89
Картина 1.	Зрители и пассажиры Понятие массива, его описание. Тип массива. Задачи. Упакованные и многомерные массивы. Задачи.	89
Картина 2.	Множество — это всегда много? Понятие множества, его описание. Операции над множествами. Задачи.	101
Картина 3.	Если бы я стал директором Понятие записи, состав записи, ее описание. Оператор присоединения. Задачи.	104
Картина 4.	Файл — любимое слово программиста Понятие файла, описание его. Чтение и запись файлов. Задачи.	110
Действие восьмое. КАК ОБЪЯТЬ НЕОБЪЯТНОЕ?		118
Картина 1.	Правила хорошего тона Понятие подпрограммы. Область действия имен.	118
Картина 2.	Покупка продуктов Процедуры. Задачи.	119
Картина 3.	Покупка молока Функции. Задачи.	127
Картина 4.	Обращение к самому себе Рекурсивные функции и процедуры. Задачи.	131
Картина 5.	Сказ про то, как Трифон автомобили размещал Трудная задача.	135
Действие последнее. ДОКА, ФОКА И КСЮША КИСЕЛЕВА		140
Картина единственная.	Вот так Ксюша Киселева!	140
Послесловие. ЗРИ В КОРЕНЬ		141
Картина 1.	Диалог с ДВК Подготовка ДВК к работе. Создание и корректировка файла программы с помощью редактора SCREEN. Выполнение программы.	141

Картина 2. Наконец-то получу результат!	147
Решение задачи на ДВК от начала до конца.	150
Картина 3. Пойду к другому редактору	
Создание и коррективировка файла программы с помощью редакторов EDIK и NED.	152
Приложение 1. Клавиатура ДВК	153
Приложение 2. Упорядоченная последовательность символов (коды КОИ-7)	155
Приложение 3. Служебные слова языка Паскаль	155
Приложение 4. Стандартные имена	156
Приложение 5. Таблица основных встроенных функций и некоторых операций	157
Список литературы	

Литература для детей

Научно-популярная библиотека школьника

**ПОДДУБНАЯ ЛЮДМИЛА МИХАЙЛОВНА
ШАНЬГИН ВЛАДИМИР ФЕДОРОВИЧ**

МНЕ НРАВИТСЯ ПАСКАЛЬ

Руководитель группы МРБ И. Н. Суслова

Редактор О. В. Воробьева

Художественный редактор Н. С. Шеин

Обложка художника В. И. Полухина

Технический редактор Т. Н. Зыкина

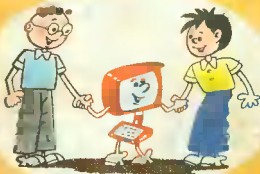
Корректор Н. Л. Жукова

ИБ № 2288

Подписано в печать 18.11.92 Формат 70×100 1/16 Бумага типограф. № 2 Гарнитура литер. Печать офсетная
Усл.печ.л. 13,0 Усл.кротт. 26,65 Уч.-изд.л. 10,60 Тираж 50000 экз. Изд. № 23129 Зак. № 2010.С=104

Издательство "Радио и связь". 101000 Москва, Почтамт, а/я 693

Московская типография № 4 Министерства печати и информации РФ Москва, 129041, Б.Переславская 46



"РАДИО И СВЯЗЬ"