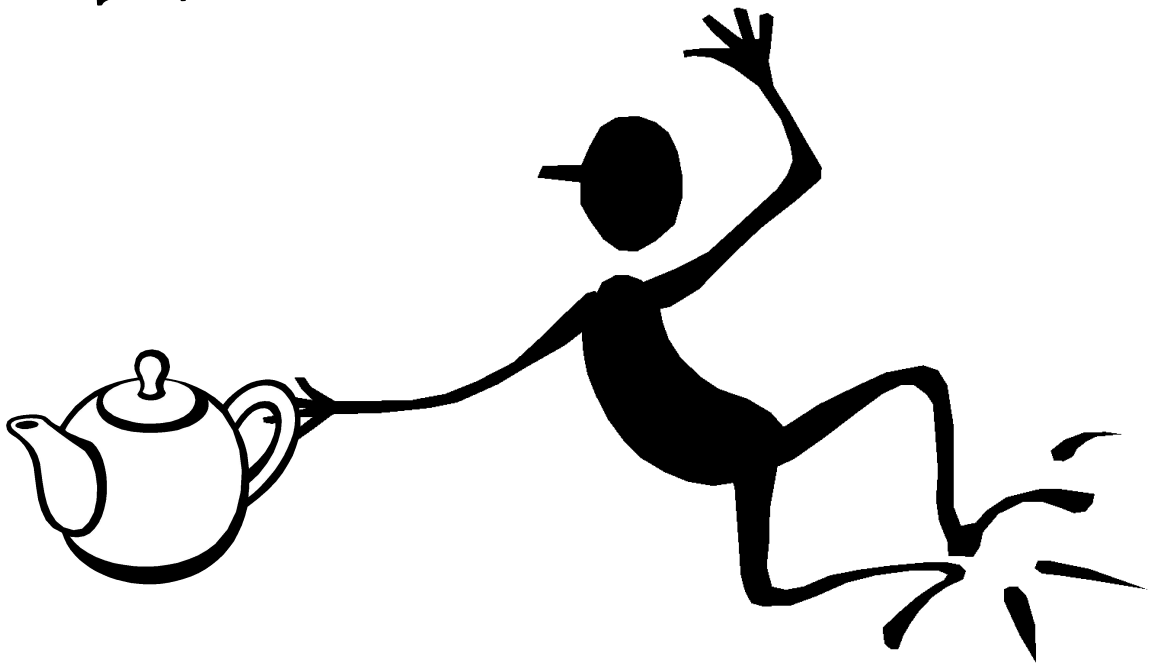
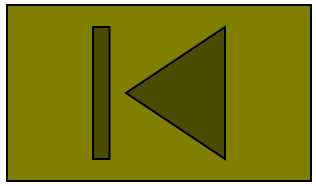
Основы языка Pascal для чайников



Оглавление

Основные понятия о языке Паскаль

- *Алфавит языка* 
- *Основные определения языка* 
- *Составные части программы* 

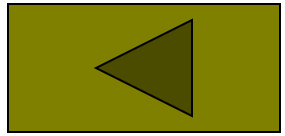


Алфавит языка

Алфавит стандартного языка Паскаль содержит следующие символы:

- 26 букв латинского алфавита
- арабские цифры
- 32 буквы русского алфавита
- специальные символы + - * / : , < > = . ‘ ; () [] {}

Русскими буквами поясняется текст в какой-либо конструкции языка Паскаль.



Основные определения языка

Рассмотрим задачу. Вычислить периметр прямоугольника со сторонами A и B.

На Кумире:

```
алг pr1
нач вещ A,B,P
  ввод A,B
  P:=(A+B)*2
  вывод нс,P
кон
```

На Паскале:

```
Program pr1;
 var A,B,P: integer; {описание данных}
Begin
 Read (A,B);          {ввод сторон A и B}
 P:=(A+B)*2;          {вычисление периметра}
 Writeln(P);          {вывод периметра}
End.
```

Программа начинается *со стандартного заголовка Program.*. После заголовка идет *описание данных.* За словом Var перечисляются все встречающиеся в задаче переменные A,B,P и указывается, что они - целого типа (integer). Далее между словами Begin и End располагаются *операторы*

Основные определения языка

Основные определения языка - это слова, элементы данных, комментарии.

Слова. В зависимости от назначения различают служебные слова и имена.

Служебное слово - это слово, которое в языке Паскаль имеет определенное смысловое значение. В нашей программе служебными словами являются Program, Var, Begin, End. Их используют только в том значении, которое зарание установлено в языке.

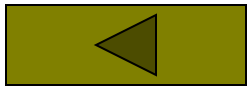
Имя (идентификатор) служит для обозначения каких-либо объектов. В языке Паскаль различают два вида имен: стандартные и даваемые пользователем.

Стандартные имена заложены в языке для обозначения стандартных объектов (например, стандартных программ, функций). Так, у нас в программе используются стандартные программы ввода-вывода, которые имеют стандартные имена Read, Writeln.

В качестве имени пользователя нельзя давать служебные слова и стандартные имена.Имя состоит из букв и цифр и обязательно начинается с буквы. В нашей программе периметр обозначен именем P. Вместо него можно было бы указать имя P1, Perim.

Основные определения языка.

Элементы данных. К ним относятся константы и переменные.
Переменные могут изменять свое значение в ходе выполнения программы. Так, в нашей программе переменными являются стороны прямоугольника и периметр. Значение переменных A и B могут изменяться в программе и в зависимости от них изменяется значение периметра.
Константа не изменяет своего значения в процессе выполнения программы, она м.б. задана явно своим значением или обозначена именем. В нашей программе явно задана константа 2.
Комментарий. Комментарий служит для пояснения программы или отдельных ее частей.Наличие комментариев делает программу более понятной и удобной для чтения. Комментарии заключаются в {}



Составные части программы

Программа на языке Паскаль состоит из заголовка, раздела описаний и раздела операторов:

PROGRAM имя;

раздел описаний

BEGIN

раздел операторов

END.

Заголовок содержит служебное слово PROGRAM, имя программы, задаваемое программистом. Заканчивается заголовок символом « ; ».

Раздел описаний предназначен для объявления всех встречающихся в программе данных и их характеристик (имена данных, их тип, возможные значения и др.). Этот раздел в свою очередь содержит: объявление меток, констант, типов, переменных, объявление процедур и функций. Они должны располагаться строго в названном порядке. Следует заметить, что не все перечисленные разделы обязательны в каждой программе. В простых программах могут потребоваться, например, только разделы:объявления констант и переменных.После каждого описания ставится символ « ; ».

Раздел операторов заключается в операторные скобки вида: BEGIN (начать) и END (окончить), при этом после END ставится точка. В разделе операторов записывается последовательность исполняемых операторов. Операторы отделяются друг от друга символом « ; ».

Составные части программы

Структура программы на языке Паскаль в общем виде:

PROGRAM имя;

LABEL - раздел меток;

CONST - раздел констант;

TYPE - раздел типов;

VAR - раздел переменных;

PROCEDURE - раздел процедур;

FANCTION - раздел функций;

BEGIN

оператор 1;

оператор 2;

... …

оператор n-1;

оператор n;

END.

Составные части программы.

Рассмотрим в качестве примера программу вычисления объма шара по формуле $V = (3/4)\pi R^2$, где R=0,2 -радиус шара

{Вычисление объема шара}

```
PROGRAM   PR1;
   CONST   PI=3.14;
   VAR      R:REAL; {радиус шара}
            V: REAL; {объем шара}
 BEGIN
   R:=0.2;
   V:=4*PI*R*R*R/3;
   WRITELN('объем шара=',V:8:3)
 END.
```

Программа состоит из трех частей: заголовка; раздела описаний; раздела операторов, заключенного в операторные скобки BEGIN - END. Перед началом программы и внутри нее имеются комментарии.

В разделе описания констант задано значение π=3.14, где в изображении числа ставится точка вместо запятой.

В разделе описания переменных объявлено, что имеются две переменные R и V, относящиеся к действительному типу (REAL), т.е. Они могут содержать любую дробную часть. Здесь же используются комментарии для пояснения физического смысла переменных R и V.

В разделе операторов прежде всего задается значение радиуса с помощью оператора **R:=0.2;** Далее непосредственно вычисляется значение объема шара

с помощью оператора **V:=4*PI*R*R*R/3;**

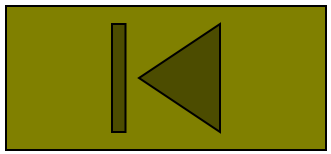
Вычесленное значение V хранится в памяти ЭВМ. Для вывода этого значения на экран дисплея используется оператор **WRITELN('объем шара=',V:8:3)**

который выводит на экран поясняющий текст **объем шара=**

и вычесленное значение V по формату 8:3 (где 8 - число позиций на изображение всего числа; 3 - число позиций на изображение дробной части). Таким образом, после выполнения программы на экране появится результат **объем шара= 0.033**

Стандартные типы данных

- *Данные целого типа*
- *Данные действительного типа*
- *Данные логического типа*
- *Данные символьного типа*
- *Описание констант и переменных стандартного типа*



Данные целого типа

Достоинством языка ПАСКАЛЬ является возможность использования широкого набора разных типов данных. Тип данных определяет возможное значение констант, переменных, функций, выражений, принадлежащих к этому типу, форму представления в ЭВМ и операции, которые могут выполняться над ними. Все типы данных можно разделить на простые и сложные.

П р о с т ы е типы - это стандартные и переменные типы данных. Стандартными являются целый INTEGER, действительный REAL, логический BOOLEAN и символьный CHAR типы данных. Переменные типы определяются пользователем ЭВМ. К ним относятся перечисляемый и ограниченный типы.

С л о ж н ы е типы данных представляют собой различные комбинации простых типов (массивы, множества, записи и файлы).

Рассмотрим стандартные типы данных, которые широко используются в программах.

Данные представляются в программе в виде констант и переменных. При выполнении программы в каждый момент времени любая переменная имеет некоторое значение. Это значение и переменная должны относиться к одному типу данных.

***Конст*анта целого типа (целая константа)** - любое десятичное число, записанное без точки. Если константа отрицательная, то перед ней должен стоять знак «-», если константа положительная, то знак «+» можно опустить.

Целая константа выражает некоторое число абсолютно точно. Она необходима в том случае, когда какую-то величину нельзя представить приближенно, например, число живых существ (людей, животных), количество предметов и т.д. Примеры констант целого типа: 14, -357, 0, 5390.

Данные целого типа

Диапазон целых чисел зависит от конкретного типа ЭВМ. Для микроЭВМ с двубайтовым словом числа чаще всего находятся в диапазоне от -32768 о+32767.

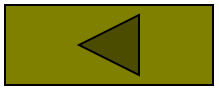
Переменные, принимающие в качестве своих значений константы целого типа, относятся также к целому типу (тип INTEGER). Над данными целого типа можно выполнять следующие арифметические операции, которые дают целый результат: + (сложение), - (вычитание), * (умножение), DIV - деление с отбрасыванием дробной части (получение целого частного при делении целого данного на целое), MOD - получение целого остатка при делении целого данного на целое.

Например, пусть , A, B, N - переменные целого типа, принимающие значения: A=25, B=2, N= -17. Тогда допустимы следующие операции:

A+51	(результат 76)
B-A	(результат -23)
B*N	(результат -34)
A DIV B	(результат 12)
A MOD B	(результат 1)

Операция MOD часто используется для определения, делится ли целое число X без остатка на 2, т. е. является 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.



Данные действительного типа

Константы действительного типа в языке ПАСКАЛЬ могут быть представлены в двух видах: с фиксированной точкой и с плавающей точкой.

Константы с ф и к с и р о в а н н о й т о ч к о й изображаются десятичным числом с дробной частью, которая может быть и нулевой. Дробная часть отделяется от целой с помощью точки, например 27.3, 5.0, -16.003, 200.59.

В математике для изображения очень больших и малых чисел используется запись числа с десятичным порядком. Например, число 680 000 000 можно записать $68 \bullet 10^{7}$ (7 - порядок числа), число 0,00000005 можно записать так $5 \bullet 10^{-8}$ (-8 - порядок числа).

В языке ПАСКАЛЬ также можно изображать числа с десятичным порядком. Они имеют вид mEp. Здесь m - мантиса; E - признак записи числа с десятичным порядком; p - порядок числа.В качестве m могут быть целые числа и действительные числа с фиксированной точкой. В качестве p могут быть только целые числа. Как мантиса так и порядок могут содержать знаки «+» или «-».

Константами с п л а в а ю щ е й т о ч к о й являются числа, представленные с десятичным порядком. Примеры чисел с плавающей точкой:

Математическая запись	Запись на языке ПАСКАЛЬ
$4 \bullet 10^{-5}$	$4E-5$
$0{,}62 \bullet 10^{4}$	$0.62E+4$
$-10{,}88 \bullet 10^{12}$	$-10.88E12$

Данные действительного типа

Следует отметить, что в языке ПАСКАЛЬ знак умножения не ставится. Рассмотрим, например, числа с плавающей точкой:

0.547E+3, 5.47E+2, 54.7E+1, 547.0E-1, 54700E-2

Эти записи представляют одно и то же число 547. Перемещая положение десятичной точки в мантисе (точка «плывет») и одновременно изменяя величину порядка, можно выбрать наиболее подходящее представление числа. Для микроЭВМ с двубайтовыми словами, как правило, самое малое по модулю число (не считая 0) 10^{-38} , а самое большое 10^{38}.

Перевод действительных чисел в двоичную систему может быть неточным, поэтому действительные числа могут быть представлены в памяти ЭВМ с некоторым приближением. Например, вместо ожидаемого числа 0.517 мы можем получить число 0.516999.

Переменные действительного типа (REAL) - это переменные, которые в качестве своих значений принимают числа с фиксированной или плавающей точкой.

Над данными действительного типа можно выполнять следующие операции, дающие действительный результат: + (сложение); - (вычитание); * (умножение; / (деление).

Данные логического типа

Логический тип данных часто называют *булевским* по имени английского математика Д. Буля, создателя особой области математики - математической логики.

В языке ПАСКАЛЬ имеются две логические константы: TRUE (*истина)* и FILSE *(ложь)*. Логическая переменная принимает одно из этих значений и имеет тип BOOLEAN.

Логические данные широко используются при проверки правильности некоторых условий и при сравнении величин. Результат может оказаться «истинным» или «ложным».

Для сравнения данных предусмотрены следующие операции отношений: < (меньше); <= (меньше или равно); = (равно); <> (не равно); >= (больше или равно); > (больше).

Если операцию отношения приложить к арифметическим данным, то получим логическое значение: отношение истинно или ложно. Например, отношение 5>3 (читается «пять больше трех?») дает истинный результат (TRUE); отношение 5=3 (читается «пять равно трем?») дает ложный результат (FALSE).

Над логическими данными допускаются следующие операции: OR - логическое сложение (ИЛИ); AND - логическое умножение (И); NOT - логическое отрицание (НЕ). Логические операции OR и AND выполняются над двумя величинами, а операция NOT - над одной.

Логическое сложение дает истинный результат, если хотя бы одна из логических величин (A или B) имеет истинное значение. Если обе величины (A или B) имеют ложное значение, то и результат операции будет ложным.

Логическо умножение дает истинный результат только в том случае, если обе величины истинны. Если хотя бы одна величина ложна, то результат будет ложным.

Логическое отрицание дает ложный результат, если величина имеет истинное значение, и наоборот.

Логический тип определяется таким образом, что FALSE < TRUE.

Данные логического типа.

Результаты операций над логическими данными:

A	B	NOT A	A OR B	A AND B
TRUE	TRUE	FALSE	TRUE	TRUE
TRUE	FALSE	FALSE	TRUE	FALSE
FALSE	TRUE	TRUE	TRUE	FALSE
FALSE	FALSE	TRUE	FALSE	FALSE

Например, результат операции **(C>10) OR (D<3)**
будет истинным: а) при C=12 и D=2; б) при C=12 и D=5; в) при C=8 и D=2;
будет ложным при C=12 и D=5.

Результат операции **(C>10) AND (D<3)** будет истинным при C=12 и D=2;
будет ложным: а) при C=12 и D=5; б) при C=8 и D=5; в) при C=8 и D=2.

Логические данные и операции над ними имеют важное значение в информатике, так как позволяют внести в расчеты элементы человеческой логики. При этом в теоретических расчетах в отличие от языка ПАСКАЛЬ часто вводят следующие обозначения логических констант: 1 - истина, 0 - ложь.

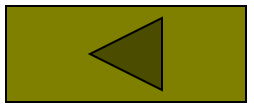
Данные символьного типа

Данные символьного типа позволяют представить в программе тексты и производить над ними некоторые редакционные операции, например, исправлять орфографические ошибки, вставлять и удалять отдельные буквы и слова. Кроме того, они дают возможность обрабатывать различные ведомости, документы, справочники и т.д.

Символьная, или *литерная константа* есть любой символ языка, заключенный в апострофы. Например, ’A’, ’+’, ’9’, ’:’.

Чтобы представить апостроф как символьную константу, его повторяют дважды: ””’. Символьная константа, как правило, занимает один байт памяти.

Символьная переменная (тип CHAR) - это переменная, принимающая значение символьной константы. Так, символы языка ПАСКАЛЬ упорядочены, то к символьным данным применяются операции сравнения, например ’A’ > ’B’.



Описание констант и переменных стандартного типа

Константы в программе могут быть заданы явно своим значением или обозначены именем. Если константа обозначена именем, она должна быть описана в разделе констант. Описание начинается со служебного CONST и имеет следующую форму записи: CONST *имя константы = значение ;*

Например, CONST N = 18;

В одном разделе допускается описание нескольких констант. Каждое описание заканчивается символом точка с запятой, например:

```
CONST
    NUM = 23;        { константа целого типа }
      B = 1.8E-3;    { действительного типа }
     PI = 3.14;      { действительного типа }
    SIM = 'R';       { символьного типа }
      L = TRUE;      { логического типа }
```

Описание констант и переменных стандартного типа

Любая переменная, встречающаяся в программе, должна быть описана в разделе переменных.

Описание начинается со служебного слова VAR и имеет следующую форму записи:
VAR *имя переменной : тип ;*

В одном разделе допускается описание нескольких переменных, например:

```
VAR
   B : INTEGER;     { переменная целого типа }
 SUM : REAL;        { действительного типа }
   K : CHAR;        { символьного типа }
 LOG : BOOLEAN;     { логического типа }
```

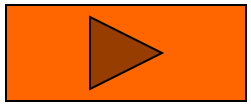
Если несколько переменных имеют одинаковый тип, то их можно объединить в список. Под *списком* понимается последовательность элементов (в нашем случае - переменных), разделенных запятой. Например:

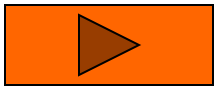
```
VAR
    R, V : REAL;
K, L, TOR : INTEGER;
```

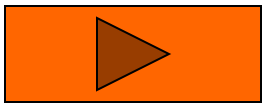
Здесь R, V - переменные действительного типа; K, L, TOR - переменные целого типа.

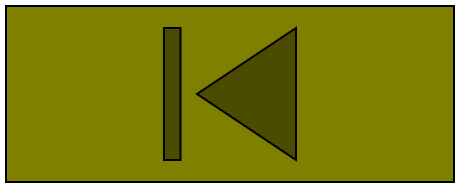
Выражения

- *Стандартные функции*
- *Арифметические выражения*
- *Логические выражения*









Стандартные функции

Элементарные функции, такие, как синус, косинус,квадратный корень, логарифм и другие, часто используются в различных задачах и вычислениях. Поэтому язык ПАСКАЛЬ имеет простые средства записи элементарных или, как их называют, стандартных функций. При этом не обязательно знать, как вычисляется стандартная функция, достаточно правильно записать ее вид.

Правила записи стандартных функций:

1. *Аргумент функции записывается в круглых скобках после имени функции.*

2. *Аргументом функции может быть константа, переменная или арифметическое выражение.*

Следует заметить, что в тригонометрических функциях синуса и косинуса аргумент может быть задан только в радианной мере. Если аргумент X дан в градусах, то для перевода его в радианы используется формула $X\pi/180$.

Стандартные функции

Рассмотрим некоторые стандартные функции:

SQRT(X) - вычисляет корень квадратный из аргумента X, что соответствует математической записи $\sqrt{X}$;

SQR(X) - вычисление квадрата аргумента X, что соответствует математической записи X^2

SIN(X) - вычисляет синус аргумента X, что соответствует арифметической записи sin x.

ABS(X) - вычисляет абсолютное значение (модуль) аргумента X, что соответствует математической записи $|X|$;

ORD(X) - определяет порядковый номер символа X, например, результатом выполнения операции ORD('R') будет число 82 (под этим номером в кодовой таблице находится символ R).

CHR(X) - определяет символ, стоящий по порядковому номеру X, например, результатом выполнения операции CHR(68) будет символ D (символ D находится в кодовой таблице под 68 номером).

PRED(X) - определяет предыдущий символ по отношению к X, например, результатом выполнения операции PRED('N') будет символ M.

SUCC(X) - определяет последующий символ по отношению к X, например, результатом выполнения операции SUCC('S') будет символ T.

При использовании функции PRED(X) и SUCC(X) необходимо помнить, что если в конкретной реализации языка нет для X предыдущего или последующего символа, то значение функции не определено. Аргументами этих функций не могут быть действительные данные, так как для них не существует понятий предыдущего и последующего элементов.

Арифметические выражения

Арифметический выражения строятся из арифметических констант, переменных, функций и операций над ними. Пример арифметического выражения:

A+B*T1/T2 - 2.3*SQRT(X).

Частным случаем выражения является константа, переменная или функция.

Все данные, входящие в арифметическое выражение должны быть одного типа. Однако во многих версиях языка допускается использование в одной операции данных целого и действительного типов. Результат операции при этом будет действительного типа. Например, разрешены операции A+N или A*2, где A - переменная действительного типа, а N - целого типа.

При составлении выражений следует выполнять следующие правила:

1. *Записывать все составные части выражений в одну строку. В выражениях двухэтажных и более верхние и нижние индексы не допускаются.* Например, формула

$$\frac{a_1 x_1 + b_2 x_2 + 5 a_3 x_3}{25d - 14f}$$

должна быть записана в виде следующего выражения:

(A1*X1+B2*X2+5*A3*X3)/(25*D-14*F).

Арифметические выражения

2. *Использовать скобки только одного типа - круглые. Применение фигурных и квадратных скобок в выражениях запрещается, так как они имеют особое назначение.* Поэтому алгебраической записи

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

в языке ПАСКАЛЬ соответствует выражение

a*(b+c*(d+e*(f+g))).

В правильно записанном выражении число открывающихся скобок всегда должно равняться числу закрывающихся скобок. При записи сложных выражений рекомендуется всегда проверять, соблюдено ли это правило.

3. *Нельзя записывать подряд два знака арифметических операций.* Например, выражение 3*A*B/-Z неверно. Его следует записать так: 3*A*B(-Z).

4. *Необходимо помнить, что вычисления выполняются слева направо в соответствии со старшинством операций.* Самой старшей является операция вычисления значения функции, например, SIN(X). Затем идут операции умножения и деления, а также целочисленные операции DIV и MOD. Операции сложения и вычитания обладают самым низким приоритетом и выполняются в последнюю очередь.

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

SIN(0.14+Z)

сначала вычисляется аргумент (0.14+Z), а затем значение функции SIN

Логические выражения

Логические выражения строятся 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 логических выражениях принят следующий приоритет операций:

1) NOT
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 скобках (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.

Логические выражения

Пример. Определить результат логического выражения

(a > 3) AND (b=a+6) OR NOT (c=4) при a=2, b=8, c=5.

Порядок выполнения операций:

а) операция сравнения a > 3 в первых скобках. Результат операции - FALSE, так как 2 < 3;

б) действия во вторых скобках в соответствии с приоритетом - сначала вычисляется выражение a+6, а затем сравнивается значение b со значением a+6. Результат операции - TRUE, так как 8=8;

в) операция сравнения c=4 в третьих скобках. Результат операции - FALSE, так как 5 ≠ 4;

г) операция NOT(c=4), равная NOT FALSE. Результат операции - TRUE;

д) операция AND над первыми и вторыми скобками (FALSE AND TRUE). Результат операции - FALSE;

е) операция OR над выражениями слева и справа от нее (FALSE OR TRUE). Результат операции - TRUE.

Внимание! В ПАСКАЛЕ нет операции возведения в степень. При необходимости ее использования применяют стандартные функции. Например, a^x заменяют выражением **EXP(x*LN(a));**

где EXP и LN - стандартные функции (экспонента и логарифм).

Основы программирования простых задач

- *Оператор присваивания*
- *Понятие о составном и пустом операторах. Назначение символа точка с запятой*
- *Простейший ввод и вывод данных*
- *Примеры программирования простых задач*
- *Создание собственных программ*

Оператор присваивания

Операторы языка ПАСКАЛЬ можно разделить на простые и сложные. Простые операторы не содержат внутри себя других операторов. Сложные (структурные) операторы представляют собой конструкции, содержащие простые операторы. К простым операторам в языке ПАСКАЛЬ относятся операторы: присваивания, перехода, пустой оператор, операторы ввода и вывода; к сложным — составной и условный операторы, операторы цикла, оператор выбора, оператор присоединения в записях.

Оператор присваивания — основной оператор любого языка программирования. Общая форма записи оператора V:=A

Здесь V – имя переменной; « := » – знак присваивания; A – выражение.

Данный оператор вычисляет значение выражения A, стоящего справа от знака операции присваивания := , и присваивает полученное значение переменной V, стоящей слева.

Следует обратить внимание на разницу между знаком операции присваивания « := » и обычным знаком равенства « = ». Это различие заключается не только в форме, но и в содержании. Например, в обычной математической записи выражение X=X+2 является неверным. Одноко запись оператора присваивания X:=X+2 правильна и означает следующее: к текущему значению переменной X (пусть до выполнения операции оно было равно 5) прибавляется число 2, и после выполнения данного оператора значение переменной X будет равно числу 7.

В частных случаях выражение в правой части оператора присваивания может принимать значение константы, имени переменной или имени функции. Например:

```
T := 527.475;
M := TEMP;
Y := SQRT(X);
```

Оператор присваивания

Оператор присваивания применим не только к арифметическим, но и к логическим и символьным данным. Например, если переменные L, M и N описаны как логические, то можно записать L:=M AND N.

Следовательно, если M TRUE, а N FALSE, то логическая переменная L получит значение FALSE.

При использовании оператора присваивания необходимо следить, чтобы переменная в левой части и выражение в правой части оператора были одного и того же типа. Так, если переменная D описана как переменная действительного типа

```
VAR
  D : REAL;
```

то оператор D := 'F'; ошибочен, так как в правой части оператора записана константа символьного типа. При трансляции этого оператора будет зафиксирована ошибка, и программа выполняться с такой ошибкой не будет.

Неверен также отрезок фрагмента программы

```
VAR A,B,C,D : REAL;
…
A:= (B<C) AND (D>C);
…
```

так как выражение в правой части оператора логического типа, а переменная A – действительного типа.

Оператор присваивания

Из этого общего правила существует одно исключение. Разрешается при целочисленном выражении использовать переменную действительного типа в левой части оператора присваивания. В этом случае значение вычисленного выражения будет преобразовано в действительный тип. Пусть, например, A и B – переменные целого тип, причем A=6, а B=5. Пусть переменная C описана как переменная действительного типа. Тогда в результате выполнения оператора

C := A*B

значение выражения целого типа A*B, равное 30, будет преобразовано в действительный тип 30.0

Примеры оператора присваивания:

```
Y := A+ROUND(B/3)*2;

SUM :=  SUM+X;

C5 := 2*K-SIN(PI/4-X);
```

Понятие о составном и пустом операторах. Назначение символа точки с запятой

Составной оператор — объединение нескольких операторов в одну группу.

Форма записи данного оператора:

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

В эту конструкцию служебные слова BEGIN *(начало)* и END *(конец)* называются *операторными скобками.*Слова BEGIN выполняет роль открывающей скобки, слово END — роль закрывающей скобки.

Составной оператор представляется как единый оператор. Его можно вставлять в любое место программы, где допускается один оператор. Любой из операторов составного оператора, в свою очередь, также может быть составным.

- После BEGIN и перед END не ставится точка с запятой; она ставится между операторами.

Понятие о составном и пустом операторах. Назначение символа точки с запятой

Пустой оператор — это оператор, не выполняющий никакого действия.

Пустому оператору соответствует отсутствие записи на том месте, где по правилам должен быть кокой-нибудь оператор. Например:

```
A := B;
R := 2;
;
K := 7.2; 3
```

Здесь третий оператор является пустым. Составной и простой операторы нередко применяются в условных операторах.

Возникает вопрос, где правильно поставить знак точка с запятой? Чтобы на него ответить, обратимся с обычному естественному языку. В любом перечне элементов между ними ставится запятая, 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 и перед закрывающей скобками запятая не указывается.

Понятие о составном и пустом операторах. Назначение символа точки с запятой

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

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

Подобная система введена и в языке ПАСКАЛЬ, только в нем роль круглых скобок выполняют операторные скобки BEGIN — END, вместо запятой ставится точка с запятой, а вместо элементов — операторы. Роль операторных скобок могут выполнять и другие служебные слова и конструкции языка.

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
```

Обратите внимание на правильную расстановку знака « ; ».

Простейший ввод и вывод данных

Ввод числовых данных. Для задания переменным их числовых значений можно использовать оператор присваивания, например:

A := 5;

BB := -6.143;

Однако в этом случае программа становится не универсальной, так как выполняется только при этих значениях переменных. Для выполнения программы при различных значениях переменных предназначен оператор ввода READ.

Как только во время выполнения программы встречается оператор READ, машина останавливается и ожидает ввода числовых значений. Когда числовые значения введены, процесс выполнения программы продолжается. Оператор ввода имеет вид

READ ($a_1, a_2, ..., a_n$)

где $a_1, a_2, ..., a_n$ – переменные, которые последовательно получают вводимые значения.

Обратите внимание: числовые значения вводятся после набора на экране дисплея всей программы и запуска ее на выполнение.

Пусть переменным A, B, C необходимо присвоить следующие значения в процессе выполнения программы: A=5, B=17, C=6.2. Оператор ввода примет вид

READ (A, B,C)

а числовые значения можно ввести следующим образом:

5 ↵ 17 ↵ 6.2 ↵

Простейший ввод и вывод данных

Если вновь повторить запуск программы, то можно ввести любые другие значения, например: 16 ↵ -4 ↵ -0.5 ↵

и переменные получат новые значения A=16, B= -4, C= -0.5, при которых будет выполняться программа. Ни один оператор программы в этом случае не изменится.

Допускается использование оператора ввода без параметров READLN, осуществляющего переход на новую строку при вводе данных. Дополнительно к этому имеется оператор ввода READLN ($a_1, a_2, ..., a_n$)

который сначала вводит значения $a_1, a_2, ..., a_n$, а затем осуществляет переход на новую строку. Этот один оператор равносилен использованию двух предыдущих операторов.

Вывод данных. Для вывода данных из памяти ЭВМ на экран дисплея предназначен оператор вывода WRITE. Форма записи оператора WRITE ($a_1, a_2, ..., a_n$)

где $a_1, a_2, ..., a_n$ являются в простом случае либо переменными, либо строкой символов, заключенных в апострофы.

Например, оператор

WRITE (‘значение B=‘ , B)

выводит на экран дисплея строку

значение B=

а затем значение переменной B.

Простейший ввод и вывод данных

Для вывода целых и действительных чисел можно указывать форматы в операторе WRITE. Формат указывается через двоеточие после переменной. Для действительных чисел формат состоит из двух величин. Первая величина обозначает общее поле выводимого значения, второе — после дробной части. Общее поле включает в себя отрицательный знак числа или пробел для положительного числа, количество цифр в целой части, точку и количество цифр в дробной части.Так, вывод значения Y в соответствии с форматом WRITE (Y:5:2) означает, что на изображение всего значения Y отведено пять позиций, из них две — на дробную часть. Например, если в результате выполнения программы значение Y равно 1.76, то в соответствии с рассмотренным форматом число на экране дисплея будет представлено в виде

Если формат отведен больше, чем количество позиций, занимаемых числом, то перед целой частью будет отведено соответствующее количество пробелов, а после дробной части – соответствующее количество нулей.

Пусть оператор вывода для значения Y=1.76 имеет вид WRITE (‘Y=‘ , Y:8:3)

Тогда информация на экране дисплея будет представлена следующим образом:

Простейший ввод и вывод данных

Для вывода целых чисел формат дробной части не указывается. Пусть необходимо вывести значение целого числа N=179. Оператор вывода можно представить в виде WRITE (‘N=‘ , N:5)

Здесь на изображение числа отведено три позиции. Если формат указан больше чем 3, например WRITE (‘N=‘ , N:5)

то перед числом будет отведено два пробела: N = ¬¬179

а для отрицательного числа – один пробел: N = ¬-179

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 ($a_1, a_2, ..., a_n$) осуществляет сначала вывод на экран дисплея значений $a_1, a_2, ..., a_n$, а затем – переход на новую строку. Таким образом, данный оператор эквивалентен двум операторам WRITE($a_1, a_2, ..., a_n$); WRITELN

Операторы ввода и вывода используются практически во всех программах.

Перед вводом данных рекомендуется давать пояснительный текст с помощью оператора WRITE. Этим самым устанавливается диалог пользователя и машины. Например, для ввода значений A, B, C лучше указать

```
WRITE (‘введите значения a, b, c’);
READ (a, b, c);
```

Таким образом, перед вводом числовых значений a, b, c на экране появится сообщение

введите значения a, b, c

после чего можно вводить значения, например: 5↵17↵6.2↵

Если пояснительный текст не давать, то пользователь зачастую забывает, значения каких переменных нужно вводить. Особенно это сказывается при выполнении больших программ с большим количеством операторов ввода.

Примеры программирования простых задач

Пример 1. Вычислить объем шара V с радиусом R по формуле $V = (4/3)\pi R^3$.

Составим программу так, чтобы можно было вычислять объем шара при любых значениях радиуса. Для этого воспользуемся не оператором присваивания R:=0.2, а оператором ввода READ (R), который позволит вводить любые значения R во время выполнения программы. Вспомним при этом, что значения R и V не должны выходить за пределы 10^{+38}.

```
PROGRAM Pr1;
  {вычисление объема шара}
   CONST   PI=3.14;
   VAR   R, V : REAL;
BEGIN
   WRITELN (‘введите значение радиуса R:’);
    READ (R);
    V:= 4*PI*R*R*R/3;
    WRITELN;
    WRITELN (‘РЕЗУЛЬТАТ’);
    WRITELN (‘объем шара =‘ , V:8:3);
    readln
END.
```

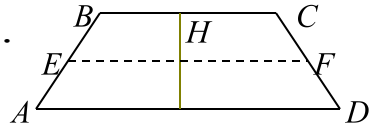
Информация на экране дисплея во время выполнения программы:

введите значение радиуса R:
0.2

РЕЗУЛЬТАТ
объем шара= ␣␣␣0.033

Примеры программирования простых задач

Пример 2. Вычислить площадь трапеции, представленной на рисунке.



Обозначим площадь трапеции через S. По определению , S=EF•H, где EF=(BC+AD)/2 – средняя линия, равная полусумме оснований трапеции; H – высота.

Составим программу.

```
PROGRAM PR2;
        {вычисление площади трапеции}
   VAR      BC,  AD, H  :  REAL; { входные переменные}
                        S : REAL;   { выходная  переменная (результат)}
                       EF : REAL;    {промежуточная переменная (средняя линия)}
BEGIN
    WRITELN (‘введите значения BC, AD, H:’);
    READ (BC, AD, H);
    EF := (BC+AD)/2;    {средняя линия}
     S:=EF*H;
     WRITELN;
     WRITE (‘площадь трапеции= ’, S:7:2);
    readln
END.
```

Информация на экране дисплея во время выполнения программы

введите значения BC, AD, H:

6 ↵
10 ↵
5 ↵

площадь трапеции = ¬¬40.00

Примеры программирования простых задач

Пример 3. Вычислить сопротивление цепи, состоящих из резисторов, соединенных:
последовательно $R_{посл} = R_1 + R_2$

параллельно $R_{парал} = \frac{R_1 \cdot R_2}{R_1 + R_2}$

Обозначим R_1 через R1; R_2 через R2, $R_{посл}$ через RPOS; $R_{парал}$ через RPAR.
Составим программу.

```
PROGRAM  PR;
    VAR          R1, R2 : REAL; { входные данные }
                 RPOS, RPAR: REAL; { выходные данные}
BEGIN
    WRITELN (‘введи значение R1’); READ (R1);
     WRITELN (‘введи значение R2’); READ (R2);
     RPOS:=R1+R2;
     RPAR:=R1*R2/(R1+R2);
     WRITELN;
      WRITELN (‘последовательная цепь, R=’ , RPOS:8:2);
      WRITELN (‘параллельная цепь, R=’ , RPAR:8:2);
      readln
END.
```

Информация на экране дисплея во время выполнения программы

```
введите значение R1
20 ↵
введите значение R2
30 ↵

последовательная цепь, R=  50.00
параллельная цепь, R=   12.00
```

Примеры программирования простых задач

Пример 4. Вычислить арифметические выражения $Y = \sqrt{X}$,$R = \sin(X + \pi / 4)$ и остаток от деления целого числа K на целое число N.

Даны значения действительного типа π , X. Обозначим π через PI, остаток от деления K на N – через OST.

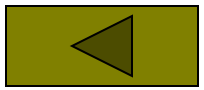
Составим программу.

```
PROGRAM  PR4;
{Арифметические выражения}
   CONST  PI=3.14;
   VAR         Y, R, X : REAL;
               K, N, OST : INTEGER;
BEGIN
   WRITELN (‘              Добрый день!  ‘);
    WRITELN (‘Введите значения X, K, N  :’);
    READ (X, K, N);
    Y := SQRT (X);
    R := SIN( X+PI/4);
    OST := K  MOD  N;
    WRITELN (‘РЕЗУЛЬТАТЫ  ВЫЧИСЛЕНИЙ:’);
    WRITELN (‘Y=’ , Y:6:2 );
     WRITELN (‘R=’ , R:6:2);
    WRITELN (‘ОСТАТОК = ’, OST : 3);
    WRITE (‘        ДО СВИДАНИЯ  !’);
    readln
END.
```

Информация во время выполнения программы имеет вид:

```
              Добрый  день!
Введите значения  X, K, N  :
6.25 ↵
17   ↵
3    ↵

РЕЗУЛЬТАТЫ ВЫЧИСЛЕНИЙ:
Y= ¬ ¬2.50
R= ¬ ¬ 0.68
OST= ¬ ¬ 2
       ДО СВИДАНИЯ  !
```



Создание собственных программ

Задача: Вычислить значения F и R , где $F = \sqrt{N}$, $R = D^2$ при следующих данных: а) N=47, D=2,5

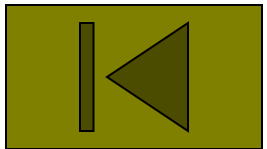
б) N=1225, D=16

в) N=6.25, (R не вычислять)

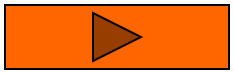
Решение задач на ЭВМ:

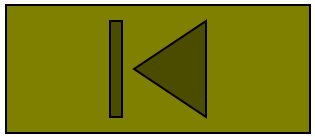
1. Запустить транслятор языка ПАСКАЛЬ.
2. Набрать текст программы в окне редактора.
3. Запустить программу на выполнение (RUN). Если на экране появилось сообщение об ошибках, то необходимо исправить ошибки и вновь запустить программу на выполнение.
4. После запуска программы на выполнение и исправления всех ошибок нужно ввести исходные данные по пункту а) и проверить правильность полученных результатов: F = 7.0 , R =6.25
5. Повторить решение задачи при новых исходных данных по пункту б) и проверить результат. F=35.0, R=256.00
6. Изменить программу (убрать команды, работающие с переменной R) и проверить правильность результатов. F= 2.5
7. Выход из ПАСКАЛЯ – Alt +X

ЗАПУСК ПАСКАЛЯ



Управляющие конструкции языка

- *Условный оператор* 
- *Оператор выбора* 
- *Оператор перехода* 
- *Примеры программирования вычислительных процессов с разветвлением* 



Условный оператор

Условный оператор используется в тех случаях, когда вычисления могут пойти по различным путям, в зависимости от выполнения или невыполнения определенных условий.

<u>Пример:</u> Вычислить арифметическое выражение *y = x+1,* если x < 0, либо *y=2x,* если x ≥ 0.

Здесь сначала проверяется условие: будет ли аргумент x положительным или отрицательным. В зависимости от этого условия вычисляется только одно значение: либо *y=x+1*, либо *y= 2x.* В языке ПАСКАЛЬ для таких вычислений 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 IF (*если),* THEN *(тогда),* ELSE *(иначе)* -- служебные слова, *оператор 1, оператор 2* **-- простые или составные операторы. Если логическое выражение истинно, тогда выполняется оператор 1, иначе (если логическое выражение ложно) -- оператор 2. В качестве оператора 1 и 2 могут быть также условные операторы. Условный оператор относится к сложным, так как в его состав входят другие операторы. Для нашего примера условный оператор имеет вид:**

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

Здесь значение X должно быть определено до выполнения оператора IF.

Условный оператор

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

Если в нашем примере описать B как логическою переменную и описать ее значение отдельно: B:=X<0 то условный оператор будет иметь вид

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

<u>ПРИМЕР.</u> Вычислить A=N+40, если значение больше 15, но меньше 25. При всех других значениях вычислить B=M+1. Условный оператор имеет вид

```
IF  ( N > 15)  AND  ( N <25)  THEN  A:= N+40
ELSE  B:= M+1
```

Обратите внимание на логическое выражение: когда выполняются одновременно оба условия N<15 и N<25 только тогда вычисляется значение A=N+40. Значение переменных N и M определяют до выполнения оператора.

Условный оператор

Правила написания программы позволяет записывать ее в свободной форме. Однако для удобства восприятия программы, особенно большой и сильно разветвленной, рекомендуется слово ELSE писать под тем 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:

$$X=\begin{cases} 1, \text{ если } A=B \text{ и } C<D \\ 2, \text{ если } A=B \text{ и } C>=D \\ 3, \text{ если } A<>B. \end{cases}$$

Условный оператор

Иногда 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.

Действия условного оператора можно расширить путем использования 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
```

Перед служебным словом ELSE не ставится знак точки с запятой.

Условный оператор

Внутри составных операторов могут быть также условные операторы. Глубина вложенности может быть ограничена в различных версиях языка .

Например, если A>B, то нужно вычислить три оператора:

Y1=7, Y2=A, Y3=A+B.

Если A<=B, то нужно вычислить T1=2A 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
```

Здесь используется два составных оператора. Возможны случаи когда используется один составной оператор, а другой простой. Начинающие программисты часто допускают такую ошибку: после служебного слова 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.

Условный оператор

В языке ПАСКАЛЬ допускается и краткая форма условного оператора :

IF *логическое условие* THEN *оператор_1*

Если логическое условие истинно, то выполняется оператор 1; иначе (если логическое выражение ложно) выполняется оператор расположенный в программе после условного оператора IF. Например,

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

В этом фрагменте представлено два независимых оператора. Один из них условный, другой -- оператор присваивания. Если условие A>15 истинно, то вычисляется значение Y. Если условие A>15 ложно, т.е. A<=15, то сразу выполняется оператор присваивания Z:=SUM+1.

Краткой формой условного оператора нужно пользоваться осторожно, так как может нарушиться вся структура при вложенных условных операторах. Вместо краткой формы рекомендуется использовать всегда полную форму, но после слова ELSE ничего не ставить (говорят, указывается пустой оператор). Так, для рассмотренного примера лучше составить следующую конструкцию:

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

Допускается использование пустого оператора и после слова THEN. Этот же пример можно записать и так:

```
IF A<=15 THEN
        ELSE Y:=X-7;
        Z:=SUM+1;
```

Здесь в логическом выражении знак операции отношения “ > ” сменился на противоположный “ <= “. В соответствии с этим поменялись местами операторы, расположенные после слов THEN и ELSE (пустой оператор и оператор Y:=X-7).

Условный оператор

Пример 5. Определить, попадает ли точка *a* с координатами X*a* и Y*a* внутрь круга с радиусом *R* . Центр круга совпадает с началом координат.

Очевидно, что точка *a* находится внутри круга, если ее расстояние от начала координат $\sqrt{x_a^2 + y_a^2}$ будет меньше радиуса окружности R , т.е. Если

$$\sqrt{x_a^2 + y_a^2} < R \quad \text{или} \quad x_a^2 + y_a^2 < R^2$$

Программа решения этой задачи может быть такой:

```
PROGRAM  PR5;
VAR  XA, YA, R, L:REAL;
BEGIN
  WRITE ('введите значения  XA, YA, R');
  READ ( XA, YA, R);
  L:=SQR(XA)+SQR(YA);
  IF  L <  SQR( R )
        THEN  WRITELN ('Точка находится внутри круга ')
        ELSE  WRITELN ('Точка находится вне круга')
END.
```

Здесь условный оператор сравнивает расстояние L точки A от начала координат с радиусом круга R. Если оно меньше радиуса R, то выполняется группа THEN, т.е. печатается сообщение ТОЧКА НАХОДИТСЯ ВНУТРИ КРУГА. В противном случае выполняется сообщение ТОЧКА НАХОДИТСЯ ВНЕ КРУГА

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 относится к сложным и имеет следующую форму записи:

CASE ***выражение*** **OF**
константа1: оператор1;
константа2: оператор2;
. . .
константаN: операторN
END

Здесь CASE (в случае), OF (из), END (конец) - служебные слова.

Оператор выбора действует следующим образом.

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 за пределы группы.

Выражение может быть любым стандартным типом, кроме действительного (REAL). В соответствии с этим и константа не может быть действительного типа. Тип константы должен совпадать с типом выражения.

Оператор выбора

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, если значение K+1 будет равно 11, 4 или 7, то выполнится один соответствующий оператор и управление будет передано за пределы оператора выбора.

Переменная K должна быть объявлена как переменная целого типа. Кроме того, K, X, A, B должны получить значения до выполнения оператора CASE.

Оператор выбора

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 символ один из знаков ‘+’, ‘-’, ‘*’, ‘/’ , то переменная P получит значение 1. Если значением S будет символ ‘A’ или ‘B’, то P получит значение 2. Если значение S будет знак точки ‘ . ’ , то переменной P будет присвоено значение 3.

Оператор выбора

Пример 6. 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.

```
PROGRAM PR6;
 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 -  THUESDAY ');
    5 : WRITELN(' ПЯТНИЦА -  FRIDAY ');
                  6 : WRITELN(' СУББОТА -  SATURDAY ');
                  7 : WRITELN(' ВОСКРЕСЕНЬЕ -  SUNDAY ');
END;
  END.
```

Результат выполнения:

ВВЕДИТЕ НОМЕР ДНЯ НЕДЕЛИ :

5

ПЯТНИЦА-FRIDAY

Оператор перехода

В языке ПАСКАЛЬ принят естественный порядок выполнения программы: все операторы выполняются последовательно один за одним в том порядке, как они записаны. Однако в практике программирования задач возникает необходимость нарушения последовательности выполнения операторов. Например необходимо обойти участок программы, а вернутся к нему позже. Для этого предназначен оператор перехода, который имеет следующую форму записи:

GOTO *метка*

Метка представляет собой любое целое число без знака в диапазоне 1-9999. Это число записывается перед помечаемым оператором и отделяется от него двоеточием:

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

Здесь после оператора GOTO 32 выполняется оператор с меткой 32.

Оператор перехода

Следует отметить, что оператор, следующий за оператором перехода, также должен быть помечен. Иначе все операторы в программе между оператором GOTO и оператором с меткой 32 будут лишними, так как доступа к ним нет и они никогда не будут выполнятся .

Оператор перехода является простым оператором, так как в его состав не входят другие операторы.

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

LABEL *метка;*

Допускается объявлять список меток:

LABEL *метка_1, метка_2, ..., метка_n;*

Для рассмотренного примера объявления меток выглядит следующим образом:

LABEL 10, 32;

Оператор перехода

В простых программах оператор перехода не вызывает никаких затруднений. В сложных программах скачки при выполнении операторов программы проследить трудно. При этом ясность и понятность программы могут быть нарушены. По этому в программах желательно избегать операторов перехода. В ряде случаев это удается 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 ;
```

Можно заменить такой конструкцией языка ПАСКАЛЬ :

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

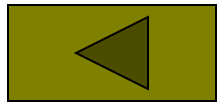
Оператор перехода

Однако в некоторых случаях оператор безусловного перехода оказывается весьма полезным. Например, пусть необходимо прекращать выполнение программы, если встечаются так называемые ситуации прерывания выражения, содержащие функции логарифма LN или корня квадратного SQRT от отрицательных аргументов, или необходимо выполнить деление на выражение, которое обращается в нуль :

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

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

          S:= T - COS (PT);
IF     S = 0     THEN     GOTO   100
                ELSE   R := 257 *  W / S ;
100 : WRITE (‘Выполнение программы прекращается  ‘);
```



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

Примеры программирования вычислительных процессов с разветвлением

Пример 7 : Даны целочисленные значения A и B . Если A=B , вывести Y=3. Если A<B, вывести Y=2. Если A>B вывести Y=3.Составим программу.

Программа.

```
Program Pr6;
  Var
        a,b : integer;   { вводимые данные }
        y   : integer;   { результат    }
  begin
    writeln('введите значения A и B ');
      readln(a,b);
    if  a = b  then  y:=1
                else   if   a < b   then   y:=2
                                      else   y:=3 ;
    writeln('Результат : ');
    write('y=',y)
end.
```

Информация во время выполнения программы имеет вид:

введите значения A и B

9

34

результат :

y=2

Примеры программирования вычислительных процессов с развилкой

Пример 8: Определить, принадлежит ли точка с координатами X ,Y прямоугольнику с координатами X1,X2,Y1,Y2. Координаты точки и прямоугольника действительного типа. Точка принадлежит прямоугольнику, если одна координата точки имеет значение $X \ge X_1$ и $X \le X_2$ и если другая координата точки имеет значение $Y \ge Y_1$ и $Y \le Y_2$

Тест:

Входные данные:

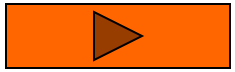
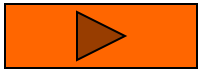
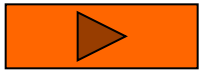
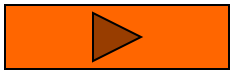
x1=3; x2=19; y1=5; y2=9

Выходные данные:

точка принадлежит прямоугольнику

```
Program   Pr7;
  var  x1,x2,y1,y2:real;{ коор-ты прямоугольника }
       x,y :real;          {  координаты   точки }
    begin
      writeln(‘ введите  x1,x2, y1,y2 ‘);
      readln(x1,x2,y1,y2);
      writeln(‘ введите x,y ‘);
      readln(x,y);
      if  ( x>=x1) and (x<=x2) and (y>=y1) and (y<=y2)
                then   writeln(‘ точка принадлежит прямоугольнику ‘);
                else     writeln(‘ точка не принадлежит прямоугольнику ‘);
end.
```

Организация циклических процессов

- *Оператор цикла с предварительным условием* 
- *Оператор цикла с последующим условием* 
- *Оператор цикла с параметром* 
- *Вложенные циклы* 
- *Программирование циклических вычислительных процессов* 

Оператор цикла с предварительным условием

При решении многих задач вычислительный процесс имеет циклический характер Это означает, что часть операторов многократно выполняется при разных значениях переменных.

В языке ПАСКАЛЬ имеется три вида операторов цикла :

1. **Оператор с предварительным условием (предусловие);**
2. **Оператор с последующим условием (постусловие);**
3. **Оператор цикла с параметром .**

Операторы для записи циклов являются сложными, так как в их состав входят другие операторы.

Для всех операторов цикла характерна следующая особенность. Повторяющиеся вычисления записываются всего один раз. Вход в цикл возможен только через его начало. Переменные оператора цикла должны быть определены до входа в циклическую часть. Необходимо предусмотреть выход из цикла: или по естественному его окончанию, или по оператору перехода. Если этого не предусмотреть, то циклические вычисления будут повторяться бесконечно. В этом случае говорят что произошло “зацикливание” программы.

Оператор цикла с предварительным условием

Цикл с предварительным условием (с предусловием) используется, как правило, в тех случаях, когда заранее известно число повторений цикла.

Форма записи цикла с пред условием:

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

Здесь WHILE *(пока)* и DO *(выполнить)* --- служебные слова.

Оператор цикла действует следующим образом. Предварительно прверяется значение логического выражения. Пока оно истинно выполняются, операторы циклической части. Как только оно становится ложным, происходит выход из цикла. Если с самого начала значение логического выражения ложно, то операторы циклической части не выполняются ни разу.

Возможен случай, когда в циклической части стоит оператор перехода, передающий управление за пределы цикла. В такой ситуации цикл может завершится до его естественного окончания.

Оператор цикла с предварительным условием

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*

Рассмотрим 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<=3n+1 является истинным, выполняются операторы циклической части. Переменные A и N, а также логическое выражение принимают следующие значения в процессе выполнения этой части программы:

A	**1**	**3**	**5**	**7**
N	**1**	**2**	**3**	**4**
2A<=3N+1	**2<=4**	**6<=7**	**10<=10**	**14<=13**
	Истинно	Истинно	Истинно	Ложно

При A=7 и N=4 логическое выражение становится ложным и управление передается за пределы цикла (т.е. за END).

Оператор цикла с последующим условием

Цикл с последующим условием, как правило, используется в тех случаях, ког - да заранее не известно число повторений цикла. Оператор цикла имеет вид :

REPEAT
операторы циклической части программы **;**
UNTIL ***логическое выражение ;***

Здесь REPEAT (*повторить)* и UNTIL *(до тех пор пока)* -- служебные слова. 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 циклическая часть, а затем проверяется условие.

Обратите внимание, 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 надобности заключать операторы циклической части в скобки вида BEGIN --- END. В то же время и дополнительное наличие скобок не является ошибкой.

Оператор цикла с последующим условием

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

Пример: Вычислить значение функции Y=X*X при X=8, 6, 4, 2.

Фрагмент программы имеет вид:

```
X:=8;
REPEAT
   Y:=X*X;
   WRITELN(X:3,Y:5);
    X:=X-2;
UNTIL  X<0
```

Здесь сначала задается первое значение аргумента X=8. Внутри циклической части выполняются следующие действия: вычисляется значение Y при текущем значении X; выводится на экран дисплея значение и X и Y; вычисляется новое значение аргумента X путем вычитания числа 2 из предыдущего значения X.

Циклическоя частьб программы повторяется до тех пор, пока выражение X<2 не станет истинным. В процессе выполнения этой цасти программы переменные принимают следующие значения:

X	**8**	**6**	**4**	**2**	**0**
Y	**64**	**36**	**16**	**4**	**_**
Выражение X<2 истинно или ложно	**False**	**False**	**False**	**False**	**TRUE**

Оператор цикла с параметром

Оператор цикла с параметром используется в тех случаях, когда заранее известно, сколько раз должна повторится циклическая часть программы. Оператор цикла имеет вид:

```
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 значения m1 до конечного значения m2 включительно.

В качестве параметра цикла может быть только переменная, в качестве m1 и m2 могут быть выражения, за исключением действительного типа (REAL);

Чаще всего параметр цикл i используют как переменную целого типа, а шаг его изменения равен +1 или -1. Если значение параметра увеличивается, то шаг его изменения +1. Если значение параметра цикла уменьшается, то шаг его изменения -1 и в операторе цикла FOR вместо служебного слова TO записывается служебное слово DOWNTO.

Оператор цикла с параметром

Рассмотрим использование оператора цикла с параметром:

Пример: Пусть имеется фрагмент программы с переменными целого типа.

```
FOR   I:=1   TO   5  DO
   BEGIN
      A:=2*I;
      B:=2*I+1;
      WRITELN(A:3, B:3)
END
```

Циклическая часть программы выполняется повторно пять раз, при этом параметр цикла i изменяет свое значение от 1 до 5. В результате выполнения программы переменная получает следующие значения:

I	**……..**	**1**	**2**	**3**	**4**	**5**
A	**……..**	**2**	**4**	**6**	**8**	**10**
B	**……..**	**3**	**5**	**7**	**9**	**11**

Оператор цикла с параметром

Фрагмент программы с убыванием значений параметра цикла от 5 до 1 имеет вид:

```
FOR  I:=1  DOWNTO  1  DO
  BEGIN
    A:=2*I;
    B:=2*I+1;
    WRITELN(A:3, B:3)
  END;
```

В процессе выполнения программы переменные принимают следующие значения:

I	**…….**	**5**	**4**	**3**	**2**	**1**
A	**…….**	**10**	**8**	**6**	**4**	**2**
B	**……..**	**11**	**9**	**7**	**5**	**3**

Оператор цикла с параметром

Если циклическая часть 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 m1 > m2, то циклическая часть не выполниться ни разу.

После естественного завершения цикла значение параметра цикла не определено. Это означает, что при последнем выполнении циклической части значение i =m2, а после ухода за пределы цикла значение i теряется.

Оператор цикла с параметром

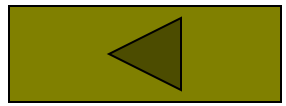
Пример: Напечатать все буквы латинского алфавита.

Так как буквы латинского алфавита упорядочены, то можно составить программу, где используется цикл с параметрами символьного типа:

```
PROGRAM  Pr9;
   Var  sim:char;
BEGIN
   writeln('Латинский алфавит');
   For  sim:='A'  TO  'Z'  DO
        WRITE('  ',sim)
END.
```

Результат выполнения программы:

```
Латинский алфавит
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
```



Вложенные циклы

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. Рассмотрим на примере структуру вложенных циклов.

Пример: Вычислить значение переменной Y=2K+N при всех значениях переменных N=1, 2, 3 и K=2, 4, 6, 8.

Обратите внимание на то, что если перебирать все значения N и K получим 12 значений Y.

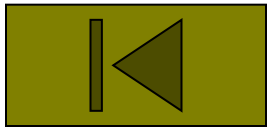
Составить программу можно следующим образом: N - параметр внешнего цикла, K - параметр внутреннего цикла. Тогда при одном значении N переменная K будет принимать значения 2, 4, 6, 8. Предполагается, что все переменные N, K, Y целого типа.

Фрагмент циклической части программы имеет вид:

Вложенные циклы

```
PROGRAM PR;
  VAR N,K,Y: INTEGER;
BEGIN
   FOR N:=1 TO 3 DO
      BEGIN
        K:=2;
        WHILE K<=8 DO
           BEGIN
             Y:=2*K+N;
             WRITELN (N:4, Y:4);
             K:=K+2
           END
      END
END.
```

Здесь внешний цикл организован с использованием оператора FOR, а внутренний - с использованием оператора WHILE.



В процессе выполнения вложенных циклов переменные получат следующие значения:

N	1	1	1	1	2	2	2	2	3	3	3	3
K	2	4	6	8	2	4	6	8	2	4	6	8
Y	5	9	13	17	6	10	14	18	7	11	15	19

Программирование циклических вычислительных процессов

Пример: Даны действительные числа. Вычислить их среднее арифметическое.

```
{ среднеарифметическое}
Program CA;
  var  i:integer;  { параметр цикла}
       n: integer;  { количество чисел}
       s :real;     { вводимое число}
        t: real;    { среднее арифметическое}
begin
  t:=0;
  writeln('введите количество чисел');
  read(n);
  for i:=1 to n do
     begin
       write('введите число ');
       read(s);
       t:=t+s;
     end;
 t:=t/n;
 writeln('----------------------------------------');
 writeln('среднее арифметическое=', t:5:2);
end.
```

Результат выполнения программы:

введите количество чисел
6
введите число
8
введите число
7
введите число
9
введите число
7
введите число
9

среднее арифметическое = 7.50

Программирование циклических вычислительных процессов

Пример: Вычислить объем каждого из нескольких шаров, а затем найти их суммарный объем. Известно, что радиус первого шара - R, радиус каждого последующего шара больше предыдущего на величину DR; радиус последнего шара - RK. Объем шара обозначим V, суммарный объем - VM.

```
Program  e10 ;
 const   pi=3.14;
 var
   r,rk: real;          (* радиус  *);
   v,vm: real;          (*  объём *);
    dr:real;           (*  изменение радиуса  *);
begin
    writeln(  ' введите начальное значение радиуса r :  ' );
    read(R);
    writeln( ' введите конечное значение радиуса rk : ' );
    read(rk);
     writeln(  ' введите шаг изменения радиуса  dr : ')
     read(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 VM=', VM:8:3)
end.
```

Результат выполнения программы:

введите начальное значение радиуса r :
0.2
введите конечное значение радиуса rk :
1.8
введите шаг изменения радиуса dr :
0.4

```
r=  0.20     v=  0.033
r=  0.60     v=  0.904
r=  1.00     v=   4.187
r=  1.40     v= 11.488
r=  1.80     v  24.417
**************************
общий объем  VM= 41.029
```

Программирование циклических вычислительных процессов

Пример: Дан произвольный текст. Признаком конца текста считается нажатие клавиши Enter. Подсчитать общее количество введенных символов текста и число букв T в тексте. Пояснение: так как заранее не известно, сколько раз будет выполняться цикл, для его организации используется оператор WHILE. Условием окончания цикла является проверка конца строки. Пока не обнаружен конец строки (NOT EOLN), цикл продолжает выполняться.

```
program e3 ;
  var   buck:char;
          k,n :integer;
begin
  k:=0;  n:=0;
  writeln('Введите текст');
  while  not eoln  do
     begin
       read(buk);
       n:=n+1;
       if  buk='T'  then k:=k+1;
     end;
 writeln;
 writeln('Количество символов  в тексте = ', n:3);
 writeln('Число буквы T в тексте =', k:3)
end.
```

Результат выполнения программы:

Введите текст

PTSPTWETPOGZPRTLO

Количество символов в тексте = 17

Число букв T в тексте = 4

Программирование циклических вычислительных процессов

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

```
Program pr;
 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.
```

Результат:

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

Программирование циклических вычислительных процессов

Пример: В последовательности целых чисел подсчитать количество четных и нечетных чисел. Для определения четности воспользуемся стандартной функцией ODD. Если число не делится на 2, то результат этой функции - истинный; иначе - ложный. Организуем цикл с помощью оператора WHILE , в который введем стандартную функцию EOLN: цикл выполняется до тех пор, пока в вводимой строке чисел не встретится символ конца строки Enter.

```
Program e4;
  var
    kn,kc,x: integer;
begin
   kc:=0;  kn:=0;
   writeln('Вводите числа ');
while not eoln do
        begin
       read(x);
       if   odd(x)  then kn:=kn+1
                          else  kc:=kc+1;
        end;
writeln('результат : ');
writeln(' Количество четных чисел', kc:4);
writeln('Количество нечетных чисел ', kn:4);
end.
```

Результата выполнения программы:

Вводите числа

5
7
33
16
13
4
10
71
9

Результат:

Количество четных чисел 3

Количество нечетных чисел 6

EXIT