Структура Паскаль - программы

Программа, написанная на Паскале может содержать следующие разделы:

- - заголовок программы;
- - раздел меток;
- - раздел констант;
- - раздел типов;
- - раздел переменных;
- - раздел процедур и функций;
- - раздел операторов.

- Не все разделы являются обязательными. Порядок размещения разделов произвольный, можно создавать несколько одинаковых разделов, но при этом должно соблюдаться правило, что в любом месте программы можно использовать лишь те элементы, которые были определены ранее по тексту или определены в стандарте языка.

Заголовок программы состоит из слова **Program** и следующим за ним через пробел именем программы. После заголовка ставится точка с запятой.

Раздел констант

Раздел констант начинается словом **Const,** за которым следует задание констант. Задание константы состоит из имени константы знака равенства и значения константы.

Например: **Const**

g = 9.8;

s = 1000;

- Значение константы задает тип константы.
- Так в приведенном примере **g** - вещественная константа, а **s** - целая.

- В любом месте программы, где будет использовано имя константы, при вычислениях будет подставлено ее значение.
- Величина константы в процессе вычислений не может изменяться.
- Естественно, что наряду с константами, заданными таким образом, в программе можно использовать явно записанные константы (значения).

Раздел типов

- Раздел типов начинается словом **Type**. В этом разделе можно сформировать новые типы.
- Задание типа состоит из имени типа, знака равенства и описания типа.

- Раздел переменных служит для приписывания типа всем переменным, используемым в программе.
- Этот раздел начинается словом **Var**, за которым следуют описания переменных, разделенные точкой с запятой.

- Описание переменной состоит из имени переменной, за которым через двоеточие следует приписываемый ей тип.
- Если один и тот же тип надо приписать нескольким переменным, их имена перечисляют через запятую, а затем после двоеточия указывают тип.

Например:
```
Var
  i,j,k : Integer;
  x,y,z : Real;
```

Раздел операторов

- Раздел операторов начинается словом **Begin** и заканчивается словом **End**, после которого ставится точка, означающая конец программы.
- Этот раздел является обязательным. В нём записываются операторы, реализующие соответствующий алгоритм.

Комментарии

- В любом месте программы могут быть помещены комментарии.
- Это любой текст, заключенный в фигурные скобки { }.
- Комментарии не влияют на выполнение программы, но облегчают ее чтение.
- Грамотно написанная программа должна содержать комментарии.

ОПЕРАТОРЫ

- Операторы языка задают те действия, которые надо выполнить для решения задачи.
- Идущие друг за другом операторы разделяются точкой с запятой.
- Все операторы Паскаля делятся на простые и структурированные.
- К простым операторам относятся такие, которые не содержат в своем составе других операторов.

Оператор присваивания

- С помощью этого оператора присваивается значение переменной.
- Оператор состоит из двух частей и знака присваивания между ними.
- Знак присваивания состоит из двоеточия и равенства **:=** .
- Слева от знака присваивания записывается переменная, справа - выражение.

- При выполнении оператора, значение выражения подсчитывается и полученный результат присваивается переменной.

- Тип переменной и выражения должны совпадать (или, по крайней мере, должны быть совместимы для присваивания).

Рассмотрим пример простейшей программы, подсчитывающей значение формулы

$V = g*t^2 / 2$, где

g - константа,

t - переменная, значение которой надо ввести.

```
Const  g = 9.8;
  Var  v,t : Real;
     Begin   {начало раздела операторов}
       Write('t = ');
       Readln(t);
           {оператор присваивания}
       v := g * t * t / 2 ;
       Writeln( 'v = ', v )     {вывод значения v}
  End.
```

Составной оператор

Составной оператор - это последовательность операторов, заключенная в операторные скобки **Begin** и **End**.

```
Begin
    Оператор 1;
    Оператор 2;
     ...
     Оператор N
End
```

Составной оператор употребляется в тех случаях, когда в соответствии с синтаксисом языка можно написать только один оператор, а по смыслу задачи надо выполнить ряд действий. В этом случае необходимые действия оформляют в виде составного оператора.

Логические выражения

Логические выражения могут принимать два значения:

True – истина,

False – ложь.

Рассмотрим отношение **a > 10**.

Это выражение имеет значение **True**, если значение переменной **a** больше 10 и имеет значение **False**, если значение переменной **a** меньше или равно 10.

Примеры отношений:

b + c <= a * Exp(x),

a / (b + c) < Ln(x) + y.

Поскольку приоритет арифметических операций выше приоритета операций отношения, дополнительные скобки в этих отношениях можно не ставить.

Логические операции

Рассмотрим три логические операции:

NOT

AND

OR

Операция **NOT**

Операция **NOT** - отрицание одноместная операция изменяет истинностное значение следующего за ней операнда.

Так, например, если значение переменной
c есть **True**,

то **NOT c** - имеет значение **False**.

Операция **AND**

- Операция **AND** - логическое “И” (конъюнкция) двуместная операция. Результат этой операции имеет значение **True** только в том случае, когда оба операнда имеют значение **True.**

Операция **OR**

- Операция **OR** - логическое “ИЛИ” (дизъюнкция) двуместная операция. Результат этой операции имеет значение True, если хотя бы один операнд имеет значение **True**, - результат **False** в том случае, если оба операнда принимают значение **False.**

Из логических операций наивысший приоритет имеет операция **NOT**,

затем следует операция **AND ,**

наименьший приоритет имеет операция **OR**.

Для изменения порядка выполнение логических операций используются круглые скобки в обычном смысле.

Операндами в логических операциях могут быть:

- логические константы,
- переменные типа **Boolean**,
- функции типа **Boolean**,
- отношения, заключенные в круглые скобки.

Примеры логических выражений:

(F > B) OR (D <= 10)

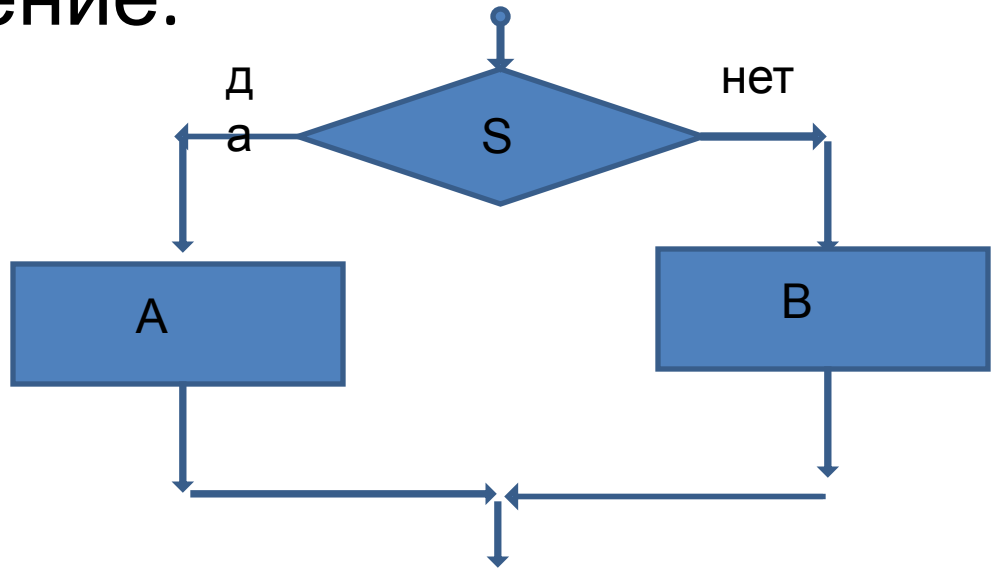
NOT (A > B)

(A > 1) AND (A < 10)

NOT ((A > 1) AND (A < 10)).

Условный оператор

Этим оператором программируют базовую конструкцию алгоритмов – ветвление.



При выполнении программы, в зависимости от истинности условия выполняется та или иная ветвь программы.

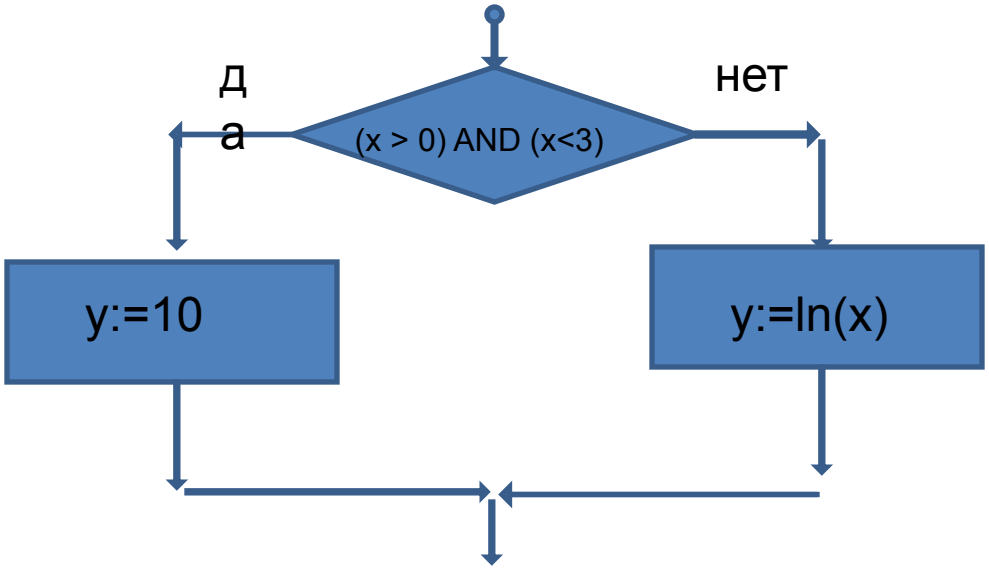
Записывается оператор в виде:

If S Then A Else B ,

где **S** - логическое выражение, истинность которого проверяется; **A** и **B** – операторы (простые или составные).

Выполняется такой оператор следующим образом: проверяется значение выражения **S**, если оно истинно, выполняется оператор **A**, иначе –

Пример.



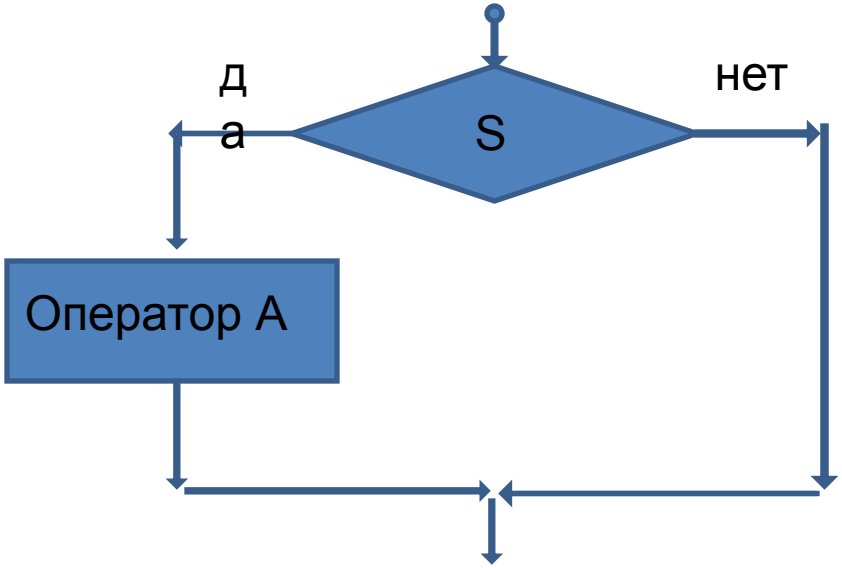
IF (x > 0) AND (x<3) Then y:=10

Else y:=ln(x);

В этом примере, если **x** находится в интервале (0,3), **y** получает значение 10, если **x** вне интервала, **y** получает значение **ln(x)**.

- Перед **Else** точку с запятой ставить не надо, так как точка с запятой указывает на конец оператора, а этот условный оператор заканчивается дальше.

Возможен сокращенный вариант логического оператора.



В этом случае оператор имеет вид:

If S Then A;

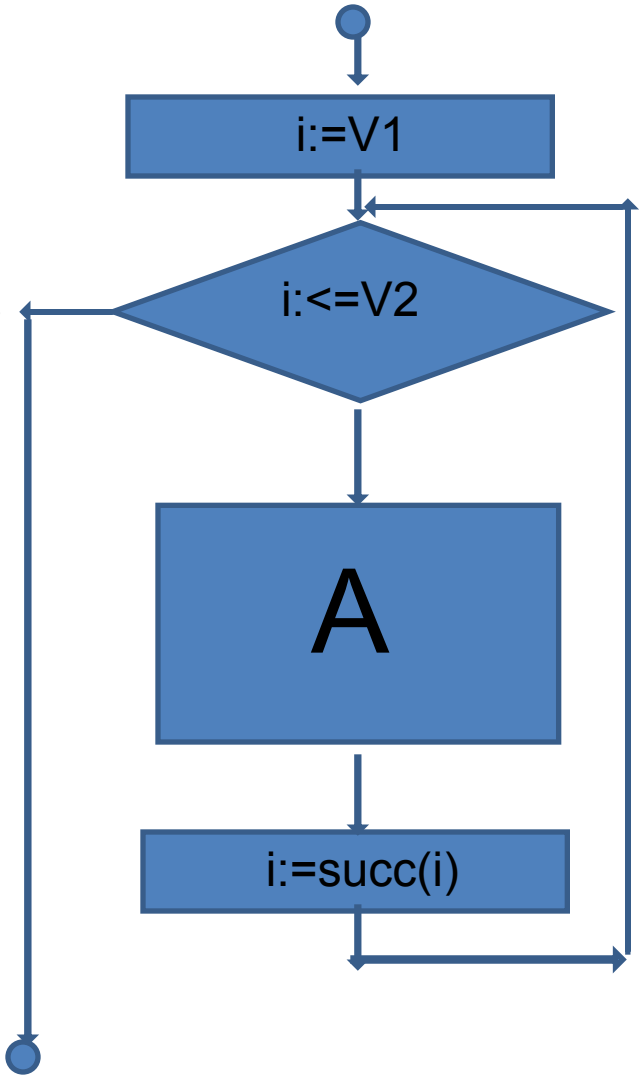
Оператор цикла с параметром

For i:=V1 To V2 Do A, где

i – параметр цикла,
переменная перечислимого типа;

V1 и **V2** - выражения того же типа,
что и параметр цикла;

A - оператор (тело цикла).



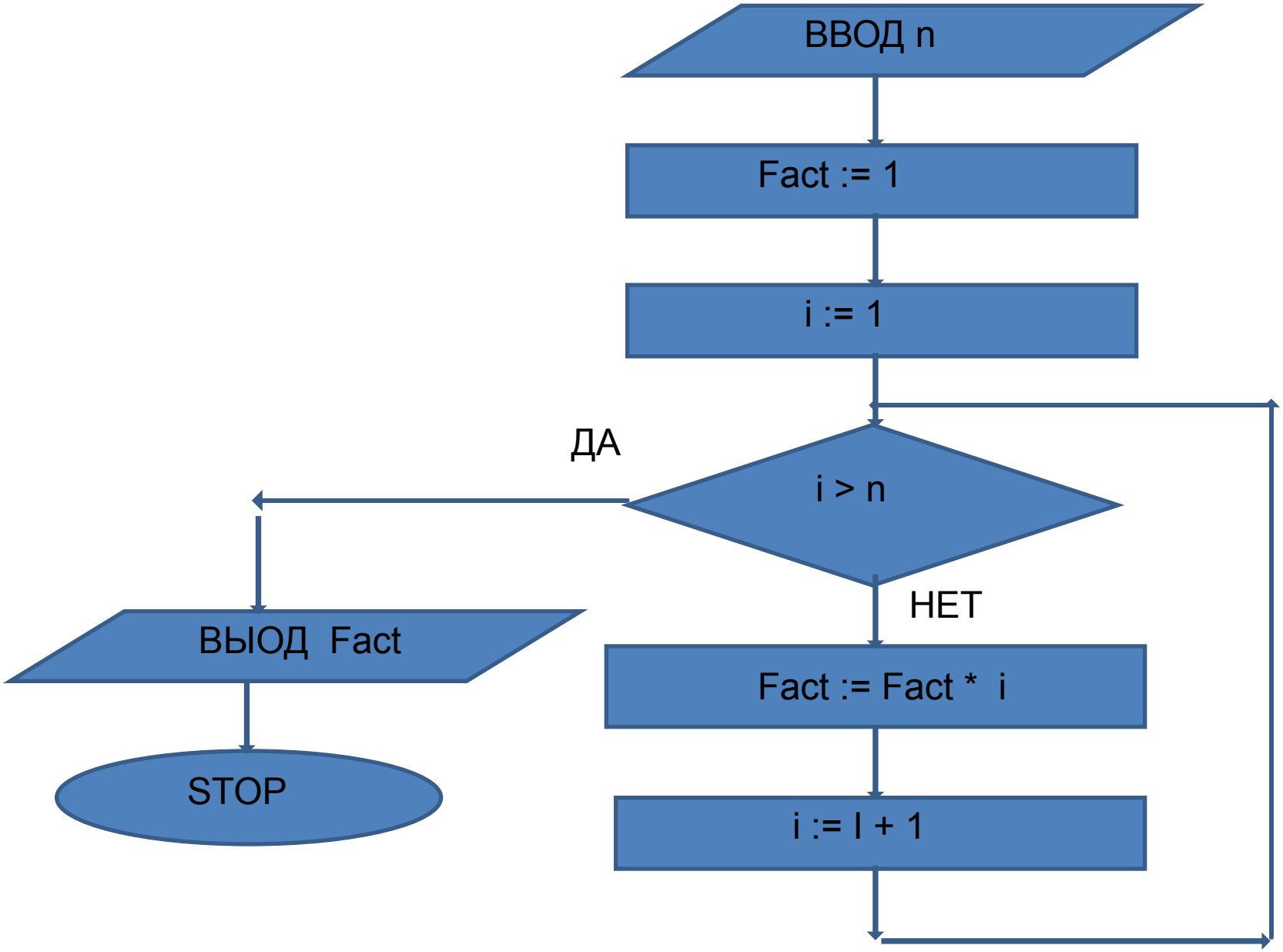
Работа оператора.

- Вычисляются значения выражений **V1** и **V2**.
- Переменная **i** (параметр цикла) получает значение **V1** и выполняется тело цикла.
- Затем переменная **i** получает новое значение, а именно следующее значение того перечислимого типа, к которому она принадлежит (т.е. значение функции **Succ(i)**).
- Если это новое значение меньше значения **V2**, то выполняется еще раз тело цикла.
- И так до тех пор, пока значение **i** не превысит значение **V2**, при этом осуществляется выход из цикла.

- Следует иметь в виду, что значения выражений **V1** и **V2** вычисляются один раз при входе в цикл, и даже если некоторые переменные, входящие в **V1** или **V2**, изменяются в теле цикла, это не приведёт к изменению числа повторений цикла.

Пример программы с параметрическим циклом

- Рассмотрим задачу подсчета факториала числа **N**.
- По определению факториал числа **N** равен произведению чисел от **1** до **N** включительно, т.е. **N! = 1*2*3*…*N**.
- Для подсчета факториала числа **N**, надо произвести **N** умножений.


ВВОД n
Fact := 1
i := 1
ДА
i > n
НЕТ
ВЫОД Fact
Fact := Fact * i
STOP
i := I + 1

```
Program FACTORIAL;
    {подсчет факториала числа n}
  Var  i,n, Fact : integer;
  Begin
    {ввод значения n}
    Write('n=');
    Readln(n);
    Fact := 1;
      {цикл для подсчета факториала}
      For i := 1 to n Do
          Fact := Fact * i;
      Writeln('n! = ', Fact)
  End.
```

Другая модификация оператора цикла с параметром

For i:= v1 downto v2 Do A;

Этот оператор отличается от рассмотренного тем, что выполняется по уменьшению значения параметра, т.е. следующее значение параметра есть **Pred(i)**.

Соответственно тело цикла не будет выполнено ни разу, если при входе в цикл **v1 < v2**.