Логические выражение и ветвления в Паскале

Логические выражения и логические переменные

Логическими константами Паскаля являются *true* (истина) и *false* (ложь). Для записи логических выражений используются операции сравнения

>	больше
<	меньше
=	равно
>=	больше или равно
<=	меньше или равно
<>	не равно

Значение логического выражения может быть присвоено переменной логического типа Boolean. Например,

```
Program Example;
Var f : Boolean;
    x : Integer;
Begin
   Write('x=');
   Read(x);
   f := x > 3;
   WriteLn(f)
End.
```

В результате работы этой программы на экран будет выведено значение TRUE, если введённое значение x больше 3, или FALSE в противном случае.

Для построения сложных логических выражений используются логические операции *not*, *and*, *or*, *xor*.

Not — это логическое отрицание (инверсия). Изменяет значение логического выражения на противоположное: not true равносильно false, not false равносильно true.

And — логическое умножение (логическое и, конъюнкция). Пусть A и B — логические выражения. Тогда выражение A *and* B истинно тогда и только тогда, когда истинны *оба* операнда: и A, и B. Например, выражение (x>3) and (x<10) истинно при x=5, ложно при x=2 и x=11.

Or — логическое сложение (логическое или, дизъюнкция). A or B истинно в случае, когда является истинным *хотя бы один* из операндов. Например, выражение (x>3) or (x=1) истинно при x=5, x=1 и ложно при x=0.

Xor — исключающее или. A *xor* B является истинным, если *только один* из операндов: или только A, или только B — является истинным. Например, выражение (x>3) xor (x<10) истинно при x=20, x=0 и ложно при x=5.

Наивысшим приоритетом среди логических операций обладает *not*, после неё следует *and*, наименьший приоритет у *or* и *xor*. Следует отметить, что в Паскале приоритет операций сравнения ниже приоритета логических операций, поэтому при построении логических выражений с несколькими сравнениями используют скобки. Например,

(x>=5) *and* (x<10)

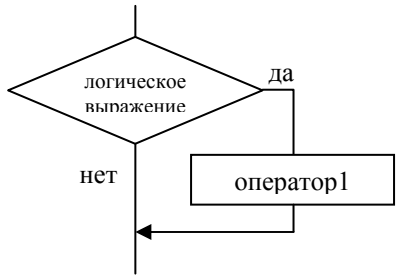
Смотрите также примеры выше.

В Delphi и FreePascal есть типы данных ByteBool, WordBool и LongBool, переменные которых могут принимать как целочисленные, так и логические значения. Однако предпочтительным для логических переменных является использование типа Boolean.

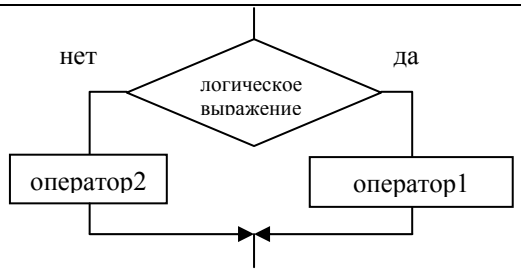
Условный оператор

Условный оператор применяется для разветвления процесса вычислений на два направления. Различают ветвление в полной и неполной форме.

Ветвление в неполной форме

Блок-схема	Паскаль
	**If** логическое выражение **Then** оператор1

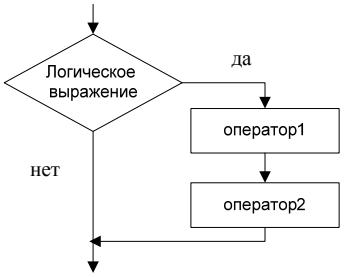
Ветвление в полной форме

Блок-схема	Паскаль
	**If** логическое выражение **Then** оператор1 **Else** оператор2

Внимание! Перед *else* точка с запятой не ставится. Она является разделителем между двумя операторами. Однако *else* — это часть условного оператора.

Во всех приведённых случаях под «оператор1» и «оператор2» понимается единственный оператор. Если же в какой-либо ветви алгоритма требуется использование нескольких операторов, то применяют конструкцию под названием составной оператор.

Составной оператор Паскаля представляет собой несколько операторов, заключённых в операторные скобки: открывающую *begin* и закрывающую *end*. По сути, тело программы тоже представляет собой один составной оператор. Вот пример ветвления с использованием составного оператора.

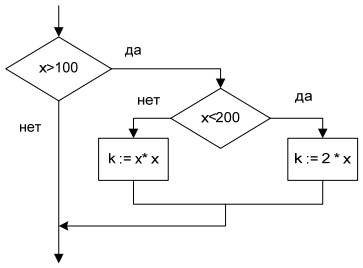


Запись этого фрагмента на языке Паскаль:

```
If логическое выражение Then
Begin
    оператор1;
    оператор2
End
```

Вместо логического выражения можно использовать логическую переменную. Обратите также внимание, что перед *end* точка с запятой не ставится. Её можно было и поставить, но трактовалась бы она как пустой (т.е. ничего не выполняющий) оператор.

При использовании вложенных ветвлений следует помнить, что *else* относится к ближайшему *if*. Рассмотрим пример.




```
If x > 100 Then
    If x < 200 Then
        k := 2 * x
    Else
        k := x * x
```

Здесь ветвь *Else* относится к сравнению *x* и 200.

Оператор выбора

Оператор выбора предназначен для разветвления процесса вычислений на несколько направлений. Этот оператор организует переход на одну из нескольких ветвей в зависимости от значения заданного выражения (селектора выбора). Формат оператора:

```
Case K Of
    A1: оператор1;
    A2: оператор2;
    …
    AN: операторN
    Else оператор
End
```

Здесь K — выражение-селектор, которое может быть типа перечисления или простого порядкового типа (целого, символьного[1]), исключая логический. A1, …, AN — константы того же типа, что и K. Они выполняют роль меток. Исполнение оператора выбора начинается с вычисления значения K. Полученное значение сравнивается с константами (метками) и выполняется лишь тот оператор, простой или составной, для которого метка совпадает со значением выражения K. После этого управление передаётся оператору, следующему после End. Если значение K не соответствует ни одной из меток, то выполняется оператор, соответствующий ветви else. Эта ветвь в операторе выбора не является обязательной. Если она опущена и K не совпадает ни с одной из меток, то в операторе выбора не выполняются никакие действия.

Одному оператору из состава оператора выбора может соответствовать несколько меток. В этом случае они указываются через запятую или задаётся диапазон в формате

минимальное_значение .. максимальное_значение

Запрещается дублирование меток, т.е. не может быть двух операторов с одинаковой меткой. Пример использования оператора выбора.

```
Program select;
Var x : Integer;
Begin
  WriteLn('Задайте натуральное число');
  Read(x);
  Case x Of
    1..3 : WriteLn('от 1 до 3');
    4,5  : WriteLn('4 или 5');
    10   : WriteLn(x)
    else WriteLn('другое')
  End
End.
```

[1] Типы-перечисления и символьный тип нами пока не обсуждались

Примеры решения задач

Задача 1.

Даны два угла треугольника в градусах. Определить, существует ли такой треугольник. Если да, то будет ли он прямоугольным.

Решение.

Сумма углов треугольника равна 180 градусов. Обозначим заданные углы через A и B, вычислим угол C=180–A–B. Он должен быть положительным. Если это не так, то треугольник не существует. Прямоугольным треугольник будет, если один из его углов равен 90 градусов. Углы прямоугольника будем считать целыми числами.

```
Program Triangle;
Var A,B,C : SmallInt;
Begin
   WriteLn('Введите известные углы треугольника');
   Read(A,B);
   C := 180-A-B;
   If C<0 Then
   WriteLn('Треугольник не существует')
   Else Begin
         Write('Существует. ');
         If (A=90) or (B=90) or (C=90) Then
         WriteLn('Прямоугольный.')
         Else WriteLn('Не прямоугольный.')
       End
End.
```

Задача 2

Из трёх данных вещественных чисел x, y, z выбрать наибольшее.

Решение 1. Используем алгоритм с вложенными полными ветвлениями.

```
Program max3_1;
Var x,y,z,max : Real;
Begin
   WriteLn('Введите 3 числа');
   Read(x,y,z);
   If x>=y
   Then If x>=z Then max:=x Else max:=z
   Else If y>=z Then max:=y Else max:=z;
   WriteLn('Максимальное число ', max)
End.
```

Решение 2. Используем алгоритм с неполными ветвлениями и сложными логическими выражениями.

```
Program Max3_2;
Var x,y,z,max : Real;
Begin
   WriteLn('Введите 3 числа');
   Read(x,y,z);
   If (x>=y) and (x>=z) Then max:=x;
   If (y>=x) and (y>=z) Then max:=y;
   If (z>=x) and (z>=y) Then max:=z;
   WriteLn('Максимальное число ', max)
End.
```

Задачи для самостоятельного решения

1. Напишите программу решения уравнения $ax + b = 0$, где a и b — заданные действительные числа, любое из которых может равняться нулю. Ответ выведите с точностью до тысячных.
2. Напишите программу решения квадратного уравнения $ax^2 + bx + c = 0$, где a, b, c — заданные действительные числа.
3. Год в григорианском календаре считается високосным, если его номер не заканчивается на два нуля и делится нацело на четыре. Если же номер года заканчивается двумя нулями, то високосным он будет только в случае, если число сотен делится на четыре. Напишите программу, которая определяет, будет ли год високосным по григорианскому календарю.