Кафедра «Автоматизированные станочные системы»

Dept. of Automated Manufacturing Systems

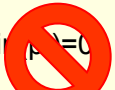
Логические значения. Ветвление

Лекция 14

Троицкий Л. 1

Операции отношения

- Действие операций отношения (сравнения) над числами соответствует их математическому пониманию. Результатом этих операций является значение «да» или «нет» (**True, False**).
- При использовании операций отношения для строковых значений сравнение выполняется ***посимвольно слева направо согласно значениям кодов символов***.
- В большинстве случаев **нельзя напрямую сравнивать вычисленное вещественное значение с константой**:



Вещественные вычисления всегда выполняются с ограниченной точностью и скорее всего sin(pi) будет равен 0.000000000001, а не точно 0.

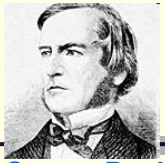
Операции отношения

Операция	Действие	Выражение	Результат
=	Равно	A = B	True, если A = B
<>	Не равно	A <> B	True, если A < B или A > B
<	Меньше	A < B	True, если A < B
>	Больше	A > B	True, если A > B
<=	Меньше или равно	A <= B	True, если A < B или A = B
>=	Больше или равно	A >= B	True, если A > B или A = B

Примеры операций отношений

Выражение	*Результат*
123 = 132	False
123 <> 132	True
17 <= 19	True
17 > 19	False
7 >= 7	True
False <> True	True
'ABC' < 'ABD'	True

Логические (булевские) операции



George Boole
1815-1864

- Всего существуют 32 булевские операции: конъюнкция, дизъюнкция, штрих Шефера, стрелка Пирса и т.д.
- Базовыми являются четыре: логическое сложение ИЛИ (OR), логическое умножение И (AND), исключающее ИЛИ (XOR) и инверсия (NOT)

Базовые логические операции

Операция	Действие	Выражение	A	B	Результат
not (унарная)	Логическое отрицание (инверсия)	not A	True		False
			False		True
and	Логическое И	A and B	True	True	True
			True	False	False
			False	True	False
			False	False	False

Базовые логические операции

or	Логическое ИЛИ	A or B	True	True	True
			True	False	True
			False	True	True
			False	False	False
xor	Исключающее ИЛИ	A xor B	True	True	False
			True	False	True
			False	True	True
			False	False	False

Примеры

Выражение	Результат
not (17 > 19)	True
(7 <= 8) or (3 < 2)	True
(7 <= 8) and (3 < 2)	False
(7 <= 8) xor (3 > 2)	False

Оператор ветвления if

- *Оператор ветвления* *if* изменяет естественный порядок выполнения операторов программы. Его общий вид:

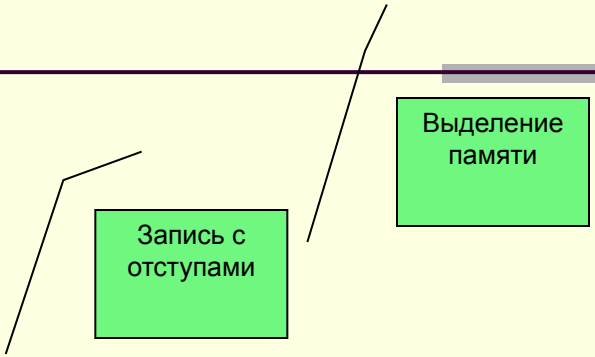
if *<условие>* ***then***
<оператор ***1****>*
else
<оператор ***2****>;*

Здесь ; не ставится!!!

При выполнении условного оператора сначала вычисляется ***условие****, результат которого может принимать только булевский тип, а затем, в зависимости от значения результата (****True, False****), выполняется* ***Оператор1****, стоящий после ключевого слова* ***then*** *(если результат равен* ***True****), или* ***Оператор2****, стоящий после ключевого слова* ***else*** *(если результат равен* ***False****).*

- Пример:

```
var
A, B, C: Integer;
begin
A := 2;
B := 8;
if A > B then
 C := A
else
C := B;
Label1.Caption:='C='+IntToStr(C);
End;
```



- В данном случае значение выражения A > B ложно, следовательно, появится сообщение «C=8».

- У оператора **if** существует и другая форма, в которой **else** отсутствует:

- ***if*** <*условие*> ***then*** <*оператор*>*;*

- Логика работы этого оператора **if** еще проще: выполнить оператор, если условие истинно, и пропустить оператор, если оно ложно.

Составной оператор

- *Составной оператор* представляет собой группу из произвольного числа операторов, отделенных друг от друга точкой с запятой и заключенную в так называемые операторные скобки - **begin** и **end**:

```
begin
<оператор 1>;
<оператор 2>;
<оператор N>
End;
```

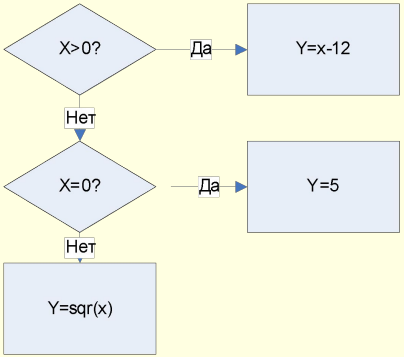
Здесь ; не нужна!

Составной оператор может находиться в любом месте программы, где разрешен простой оператор

Вложенные операторы if

Операторы IF могут быть вложенными. Если часть else используется во вложенных if, то каждое else соответствует тому if, который ему непосредственно предшествует.

```
if x>0
  then
    y := x-12
  else
    if x=0
      then
        y := 5
      else
        y := sqr(x);
```



Оператор ветвления case

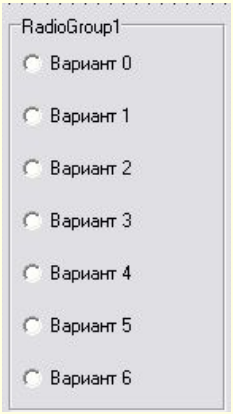
- *Оператор ветвления* **case** применяется, если необходимо сделать выбор из конечного числа заранее известных вариантов. Он состоит из выражения, называемого *переключателем*, и операторов, каждому из которых предшествует свой *список допустимых значений переключателя*:

case *<переключатель>* ***of***
*<список **1** значений переключателя>: <оператор **1**>;*
*<список **2** значений переключателя>: <оператор **2**>;*
……
<список N значений переключателя>: <оператор N>;
else *<оператор N+**1**>*
end*;*

- **Оператор case вычисляет значение переключателя (который может быть задан выражением), затем последовательно просматривает списки его допустимых значений в поисках вычисленного значения и, если это значение найдено, выполняет соответствующий ему оператор. Если переключатель не попадает ни в один из списков, выполняется оператор, стоящий за словом else. Если часть else отсутствует, управление передается следующему за словом end оператору.**

- **Все значения переключателя должны быть уникальными, а диапазоны не должны пересекаться, иначе компилятор сообщит об ошибке. Тип значений должен быть совместим с типом переключателя.**

Пример




```
case RadioGroup1.ItemIndex of
  0: Label1.Caption:= ‘Выбран вариант‘ 0;
  1, 3..5: Label1.Caption:= ‘Выбран вариант 1
    или ‘3..5;
  else Label1.Caption:= ‘Не пойму, что
    выбрано';
end;
```