**Операторы выбора if и case**

**Условный оператор If (оператор ветвления)**

**Полная формы условного оператора** называется **If-then-else**, имеет вид:

*If условие*

*Then [оператор1]*

*Else [оператор2];*

Если в качестве оператора должна выполняться серия операторов, то они объединяются в операторные скобки Begin-End.

*If then*

*Begin*

*[операторы1]*

*End*

*Else*

*begin*

*[операторы2]*

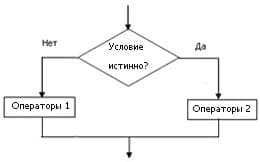
*End;*

Слова if (если) и Then (то) – зарезервированные.

Замечание. Перед ключевым словом else (иначе) точка с запятой не ставится, она ставиться после всего оператора If.

Условие представляют собой логические выражение. Простые условия  записываются в виде равенств или неравенств. В сложных условиях несколько простых условий объединяются с помощью логических операций or, and. В этом случае простые условия заключаются в круглые скобки.

На рисунке показана блок-схема полной формы оператора.



Алгоритм выполнения полного условного оператора:

Проверяется значение логического выражения, записанного в условии после слова If

Если условие истинно (значение выражения условие равно True), то выполняется оператор 1, указанный после слова Then

Если условие ложно, то выполняется оператор 2, записанный после слова  Else.

Неполная форма условного оператора называется if-then, имеет вид:

*If условие*

*Then [оператор];*

*Если в качестве оператора должна выполняться серия операторов, то они объединяются в операторные скобки Begin-End.*

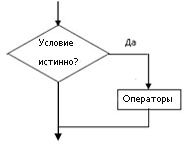
*If условие Then*

*Begin*

*[операторы]*

*End;*

В неполном ветвлении ветвь Else отсутствует.  
На рисунке показана блок схема оператора if-then.



*Алгоритм выполнения неполного условного оператора*

1. Проверяется значение логического выражения, записанного в условии после слова If
2. Если условие истинно (значение выражения условие равно True), то выполняется операторы, указанный после слова Then
3. Если условие ложно, то ничего делать не надо. Происходит переход к следующему оператору.

Операторы ветвления могут быть вложенными, т.е. один оператор if можно расположить внутри другого. В этом случае он позволяет производить выбор более чем из двух вариантов. При написании исходного кода с вложенными операторами для удобочитаемости  кода вложенные операторы выделяются отступами.

*If then begin*

*[операторы 1;]*

*end*

*Else if eckjdbt 2 then begin*

*[операторы2;]*

*End*

*.*

*.*

*.*

*else if условиеN*

*[операторыN;]*

*end*

*else begin*

*;*

end;

**Оператор выбора Case**

Кроме оператора if в языке программирования Паскаль предусмотрен так называемый переключатель case. Его можно трактовать как некий вопрос, имеющий большое число ответов (а не только два, как это имеет место в операторе if-else). Однако в отличие от if, case имеет ряд принципиальных ограничений. Его формат следующий:

**case** селектор **of**

значение1: оператор1;

значение2: оператор2;

значение3: оператор3;

...

**else** операторN

**end**;

В заголовке оператора case вместо логического выражения фигурирует переменная, которую называют селектором. До этого в программе ей присваивается какое-либо значение. Эта переменная может иметь только перечисляемый тип (например, она не может быть вещественного типа). По ходу выполнения оператора case, значение переменной-селектора сравнивается с различными, описанными в нем альтернативами (метками-значениями). Как только совпадение будет найдено, то выполняется блок кода при данной метке и происходит выход в основную ветку программы. Значения-метки являются константами, которые может принимать селектор. Их тип и тип селектора должны быть совместимы по присваиванию.

Если совпадений не будет, то выполняется блок else. Если блок else отсутствует (он является не обязательным), то никакой блок кода в операторе case не выполняется.

**var** n: **integer**;

**begin**

write ('Введите класс школы: ');

readln (n);

**case** n **of**

1..4: writeln ('Младшие классы.');

5..8: writeln ('Средняя школа.');

9,11: writeln ('Старшие классы. Выпускной.');

10: writeln ('Старшие классы.');

**else** writeln ('Error')

**end**;

readln

**end**.

На использование оператора выбора накладываются следующие ограничения:

* селектор должен иметь какой-либо порядковый тип;
* каждая альтернатива должна быть константой, диапазоном, списком диапазонов, но не переменной или выражением.

Логическое выражение сравнивает какие-либо значения между собой. Для сравнения существуют следующие знаки:

<    – меньше;

> ­   – больше;

<=  – меньше или равно;

>=  – больше или равно;

<>  – не равно;

=    – равно.

Далее приведена таблица использования каждого символа и результаты выражений:

|  |  |  |
| --- | --- | --- |
| Знак | Выражение | Результат |
| < | 3<5 | True |
| > | 3>5 | False |
| <= | 3<=3 | True |
| 3<=5 | True |
| 3<=2 | False |
| >= | 3>=2 | True |
| 3>=3 | True |
| 3>=5 | False |
| <> | 3<>3 | False |
| 3<>4 | True |
| = | 3=3 | True |
| 3=3 | True |

Слева или справа от знака может находиться переменная, константа, арифметическое выражение или строка. Например:

(2+3)<=(x\*4)

Обратите внимание, что арифметические выражения слева и справа обрамлены круглыми скобками. Это потому, что данные символы имеют приоритет при вычислении. Т.е. если бы они не использовались:

2+3 <= x\*4

То во время работы программы вначале начнут сравниваться 3 и х. А затем программа попытается результат этого сравнения помножить на 4. Так как нельзя логическое значение помножать или складывать, то возникнет ошибка.

Литература

<http://dvkuklin.ru/fs_ABC/P_05.html>

<https://gospodaretsva.com/urok-6-uslovnye-operatory.html>