Условный оператор

До сих пор все команды в наших программах выполнялись **последовательно**, сверху вниз. Но иногда какой-либо оператор нужно выполнять не всегда, а только при определенных значениях. Для того, чтобы поставить **условия** выполнения того или иного блока операторов, нужно воспользоваться условным оператором **if**.

Оператор **if** вычисляет условное выражение указанное в **круглых скобках**, результатом этого выражения является **логическое значение**. Если полученный результат является истинным (**true**), то инструкция выполняется. Если выражение возвращает ложное значение (**false**), то инструкция не выполняется. Все инструкции, которые выполняются при **истинности** условия заключаются в **фигурных скобках**.

  В общем виде **оператор** **if** записывается так:

if (выражение)  
{  
   инструкция;

   инструкция;

   инструкция;

}

Пример:

int x = Convert.ToInt32(Console.ReadLine());

if (x > 10)

{

   Console.WriteLine(x);

}

Операторы **if** могут идти последовательно, проверяя несколько условий подряд:

int x = Convert.ToInt32(Console.ReadLine());

int x = Convert.ToInt32(Console.ReadLine());

if (x > 10)  
{

Console.WriteLine(x);

}

if (y > 10)  
{

Console.WriteLine(y);

}

Операторы **if** могут быть вложенными в другие операторы **if**:

int x = Convert.ToInt32(Console.ReadLine());

int y = Convert.ToInt32(Console.ReadLine());

if (x > 10)  
{

   if (y < 8)

   {

      Console.WriteLine(x + y);

   }

}

Если же условие оказывается **ложно**, то никаких действий не выполняется. Однако часто бывает необходимо выполнить одни инструкции, если определенное условие **истинно**, и другие инструкции, если условие **ложно**. Именно для таких случаев используется ветвление **if - else**. Оно состоит из оператора **if**, за которым следует блок инструкций и ключевого слова **else**, за которым следует другой блок инструкций.  Синтаксис оператора **if else**:

if (выражение)

{  
   инструкция;

   инструкция;

   инструкция;

}

else

{

   инструкция;

   инструкция;

}

Оператор else **не является обязательным**. Блок инструкций расположенный после **else** выполняется, когда условное выражение в **if** возвращает значение **false**.

Оператор else **не может быть использован** отдельно от оператора **if**. Блок **else** должен располагаться только после оператора **if**.

Рассмотрим пример кода, который определяет является ли число положительным:

int x = Convert.ToInt32(Console.ReadLine());

if (x > 0)  
{

   Console.WriteLine("Положительное");  
}

else  
{  
   Console.WriteLine("Отрицательное");  
}

Можно использовать комбинацию **if** и **else**, если необходимо проверить несколько условий подряд, но не проверять после первого истинного:

int x = Convert.ToInt32(Console.ReadLine());

if (x > 0)  
{

   Console.WriteLine("Положительное");  
}

else if (x < 0)  
{  
   Console.WriteLine("Отрицательное");  
}

Операторы сравнения

В условных выражениях нужно пользоваться операторами сравнения:

* **>, <** - больше, меньше
* **>=, <=** - больше либо равно, меньше либо равно
* **==, !=** - равно, не равно

Стоит отметить, что "**=**" - оператор **присвоения**, а два раза равно "**==**" - это оператор **сравнения**, которое сравнивает на равенство два объекта. Для лучшего понимания рассмотрим пример:

static void Main(string[] args)  
{  
  int b = 2;

int c = 3;

int d = 2;  
  Console.WriteLine(b > c); // false  
  Console.WriteLine(b < c); // true  
  Console.WriteLine(b >= c); // false  
  Console.WriteLine(b >= d); // true  
  Console.WriteLine(b == c); // false  
  Console.WriteLine(b == d); // true  
  Console.WriteLine(b != c); // true  
}

Давайте разберем программу, которая выводит **"YES"**, если введенные два числа равны, в противном случае - **"NO"**:

int a = Convert.ToInt32(Console.ReadLine());

int b = Convert.ToInt32(Console.ReadLine());

if (a == b)

{

    Console.WriteLine("YES");

}

else

{

    Console.WriteLine("NO");

}

Логические операторы

**Логические операторы** выполняются над **логическими типами** данных. Существуют следующие логические операторы - **&&**, **||**, **!**. Как и операторы сравнения, логические операторы возвращают одно **логическое** значение - **true** либо **false**, в зависимости от значений находящихся по обе стороны логических значений.

Рассмотрим эти операторы в порядке приоритетов операций над ними:

Логическое НЕ

Его также называют **отрицание**, обозначается знаком "**!**". Он является **унарным** оператором, помещаемым перед одиночным операндом. Оператор «**логическое НЕ**» используется для инверсии логического значения своего операнда, то есть значение **true** - заменяет на **false** и наоборот:

bool f = true;

bool d = !f; // false

bool h = !d; // true

Если потребуется инвертировать значение выражения, например **a &&**

**b**, необходимо будет использовать круглые скобки: **!(a && b)**.

Логическое И

Обозначается как **&&**. Он выполняет операцию «**логическое И**» над **двумя** операндами. Он возвращает **true** **тогда и только тогда**, когда **оба** операнда имеют значение **true**. Если один или оба операнда возвращают значение **false**, оператор "**логическое И"** возвращает **false**:

bool a = true;

bool b = true;

bool c = a && b; // true

c = !a && b; // false

c = a && !b; // false

c = !a && !b; // false

Логическое ИЛИ

Обозначается как **||**. Он выполняет операцию «**логическое ИЛИ**» над **двумя** операндами. Если **один или оба** операнда имеют **истинное** значение, он возвращает **true**. Если оба операнда имеют **ложные** значения, он возвращает **false**:

bool a = true;

bool b = true;

bool c = a || b; // true

c = !a || b; // true

c = a || !b; // true

c = !a || !b; // false

Если выражение содержит операторы с разным приоритетом, то порядок их исполнения может влиять на конечный результат выражения. С помощью круглых скобок "**(**" и "**)**" можно **переопределить порядок** исполнения операторов или просто улучшить читаемость выражения.

Операторы с **одинаковым приоритетом** обрабатываются в порядке следования в выражении.

Вложенные условные операторы

В операторе **if** можно использовать не только операторы ввода вывода и присваивания, но и любой другой оператор, в частности, другой оператор **if**. Для лучшего понимания, рассмотрим пример определения знака числа:

int n = Convert.ToInt32(Console.ReadLine());  
if (n > 0)  
{  
   Console.WriteLine("Положительное");  
}  
else  
{  
   if (n < 0)  
   {  
      Console.WriteLine("Отрицательное");  
   }  
   else  
   {  
      Console.WriteLine("Ноль");  
   }  
}

Оператор switch

Часто вместо нескольких расположенных подряд операторов **if** и **else** целесообразно воспользоваться специальной конструкцией **switch-case**. Оператор **switch** сравнивает значение условного выражения с несколькими значениями. Как правило, в качестве выражения используется переменная, в зависимости от значения которой должен быть исполнен тот или иной блок кода. Для сравнения в **switch** используется оператор равенства (**==**).

Представим себе переменную **action**, которая может иметь значения **"Прыгать"**, **"Плавать"** и **"летать"**. Оператор **switch** позволяет легко определить блок кода, который должен исполняться для каждого из этих значений.

  В общем виде конструкция **switch-case** записывается так:

switch (переменная)  
{

case “значение”: инструкция; break;

case “значение”: инструкция; break;

…

case “значение”: инструкция; break;

default: инструкция; break;

}

Если нам нужно указать действие, которе будет выполняться, если **условие не подошло** ни к одному из **case**, будет выполняться блок под оператором **default**. Оператор **default** **не обязателен**.

Чтобы показать разницу между операторами **if** и **switch**, выполним проверку переменной на соответствие нескольким значениям. Следующий пример показывает два различных способа сделать то же самое. Первый способ использует серию операторов **if** и **else if**, а второй - оператор **switch**:

string action = Console.ReadLine();

if (action == "Прыгать")

{

Console.WriteLine("Мне нравится прыгать.");  
}  
else if (action == "Плавать")  
{

Console.WriteLine("Я люблю плавать.");  
}  
else if (action == "Летать")  
{

Console.WriteLine("Хотел бы я научиться летать.");  
}

switch (action)  
{  
  case "Прыгать": Console.WriteLine("Мне нравится прыгать."); break;  
   case "Плавать": Console.WriteLine("Я люблю плавать."); break;  
   case "Летать": Console.WriteLine("Хотел бы я научиться летать."); break;  
}

Оператор **switch** берет значение, и начинает сравнивать его **со всеми значениями**, стоящими рядом с ключевым словом **case**, **в порядке их следования**. Как только совпадение обнаружено, выполняется **только** соответствующий блок кода после двоеточия. Если соответствие **не найдено**, **не** исполняется ни один из блоков, или блок **default**, если мы его укажем.

Использовать оператор **switch** рационально использоватьЕсли у вас слишком много условий, в остальных случаях достаточно ограничиться операторами **if** и **else**.