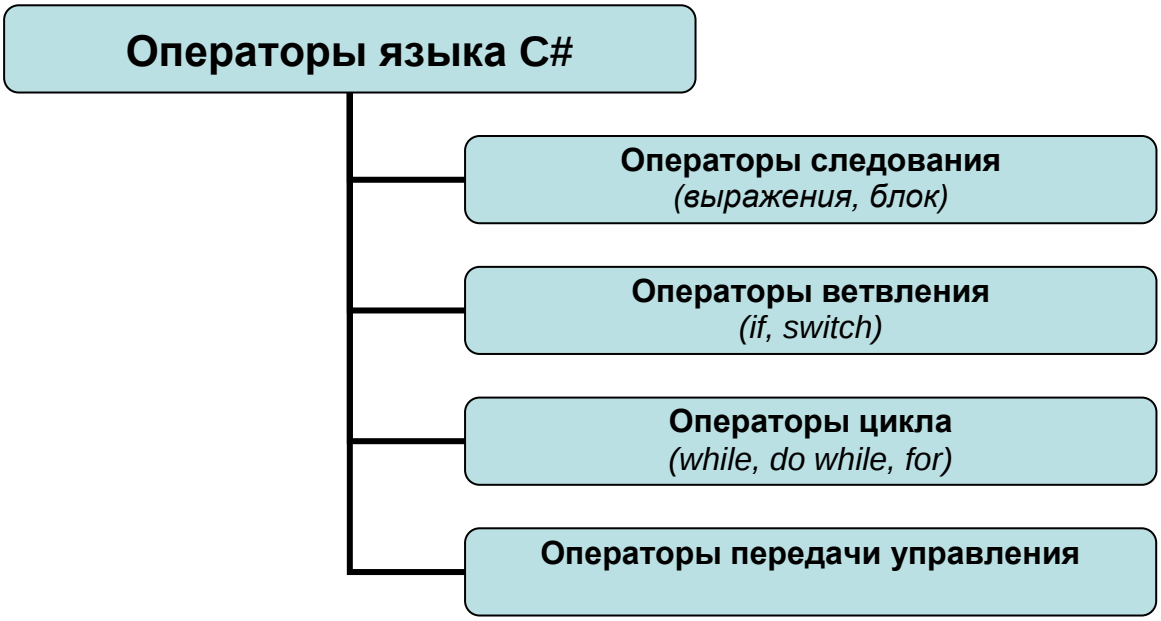
ОПЕРАТОРЫ ЯЗЫКА C#

Программа на языке C# состоит из последовательности операторов, каждый из которых определяет законченное описание некоторого действия и заканчивается точкой с запятой. Все операторы можно разделить на 4 группы: операторы следования, операторы ветвления, операторы цикла и операторы передачи управления.



Операторы следования

Операторы следования выполняются в естественном порядке: начиная с первого до последнего. К операторам следования относятся: выражение и составной оператор.

Любое *выражение*, завершающееся точкой с запятой, рассматривается как оператор, выполнение которого заключается вычислением значения выражения или выполнением законченного действия, например, вызовом метода. Например:

```
++i;                    //оператор инкремента
x+=y;                   //оператор сложения с присваиванием
Console.WriteLine(x);   //вызов метода
x=Math.Pow(a,b)+a*b;    //вычисление сложного выражения
```

Составной оператор или *блок* представляет собой последовательность операторов, заключенных в фигурные скобки {}. Блок обладает собственной *областью видимости*: объявленные внутри блока имена доступны только внутри данного блока или блоков, вложенных в него. Составные операторы применяются в случае, когда правила языка предусматривают наличие только одного оператора, а логика программы требует нескольких операторов. Например, тело цикла *while* должно состоять только из одного оператора. Если заключить несколько операторов в фигурные скобки, то получится блок, который будет рассматриваться компилятором как единый оператор.

Операторы ветвления

Операторы ветвления позволяют изменить порядок выполнения операторов в программе. К операторам ветвления относятся условный оператор *if* и оператор выбора *switch*.

Условный оператор if

Условный оператор *if* используется для разветвления процесса обработки данных на два направления. Он может иметь одну из форм: *сокращенную* или *полную*.

Форма *сокращенного оператора if:*

```
if (B)  S;
```

где *B* – логическое выражение, истинность которого проверяется; *S* – оператор: простой или составной.

При выполнении сокращенной формы оператора *if* сначала вычисляется выражение *B*, затем проводится анализ его результата: если *B* истинно, то выполняется оператор *S*; если *B* ложно, то оператор *S* пропускается. Таким образом, с помощью сокращенной формы оператора *if* можно либо выполнить оператор S, либо пропустить его.

Форма *полного оператора if*:

```
if (B) S1; else S2;
```

где *B* – логическое выражение, истинность которого проверяется; *S1, S2*- оператор: простой или составной.

При выполнении полной формы оператора *if* сначала вычисляется значение выражения *B*, затем анализируется его результат: если *B* истинно, то выполняется оператор *S1*, а оператор *S2* пропускается; если *B* ложно, то выполняется оператор *S2*, а *S1* – пропускается. Таким образом, с помощью полной формы оператора *if* можно выбрать одно из двух альтернативных действий процесса обработки данных.

Рассмотрим несколько примеров записи условного оператора *if*:

```
if (a > 0) x=y;          // Сокращенная форма с простым оператором
if (++i>0) {x=y; y=2*z;}          // Сокращенная форма с составным оператором
if (a > 0 || b<0) x=y; else  x=z;          // Полная форма с простым оператором
if (i!=j-1) { x= 0; y= 1;} else {x=1; y:=0;}   // Полная форма с составными операторами
```

Операторы S1 и S2 могут также являться операторами *if*. Такие операторы называют вложенными. При этом ключевое слово *else* связывается с ближайшим предыдущим словом *if*, которое еще не связано ни с одним *else*. Рассмотрим несколько примеров алгоритмов с использованием вложенных условных операторов:

Пример 1. ***Уровни вложенности***

```
if (A>B)
if (C>D) X=Y;
else X=Z;
else X=R;
```

2 1

Пример 2. ***Уровни вложенности***

```
if (A>B) X=Y;
else if (C>D) X=Z;
     else X=R;
```

2 1

Пример 3. ***Уровни вложенности***

```
if (A < B)
   if (C < D) X =Y;
   else  X = Z;
else
   if (E < F) X= R;
   else  X = Q;
```

2 2 1

Пример 4 ***Уровни вложенности***

```
if (A< B)
   if (C < D)
     if (E < F) X= Q;
     else X = R;
   else X= Z;
else X = Y;
```

3 2 1

Замечание. *Т.к. оператор if допускает наличие только одного оператора действия, то можно записывать его без использования блока {}, например, так:*

```
if (x < 0) y = 0; else y=1;
```

Однако предпочтительнее и в этом случае ставить блок, начиная его с новой строки. Это улучшает читабельность программы и значительно сокращает вероятность ошибки при внесении последующих изменений. В нашем случае оператор if следует записать так:

```
if (x < 0)
{
    y = 0;
}
else
{
    y=1;
}
```

Далее будем придерживаться данного правила не только для оператора if, но и для других операторов.

Рассмотрим несколько примеров использования оператора if.

Пример 1. Найдем наибольшее значение из двух вещественных чисел:

```
static void Main()
    {
        Console.Write("x= ");
        double x = double.Parse(Console.ReadLine());
        Console.Write("y=");
        double y = double.Parse(Console.ReadLine());
        double max;
        if (x > y )
        {
            max=x;
        }
        else
        {
            max=y;
        }
        Console.WriteLine("max= {0}", max);
    }
```

Задания.

1. Объясните, почему в данном примере не требуется инициализация переменной max.
2. Измените программу так, чтобы вычислялось наименьшее значение из двух вещественных чисел.

Замечание. *Вычислить максимум из двух чисел можно с помощью метода Math.Max(x,y).*

Пример 2. Найдем наибольшее значение из трех вещественных чисел:

```
static void Main()
{
   Console.Write("x= ");
   double x = double.Parse(Console.ReadLine());
   Console.Write("y=");
   double y = double.Parse(Console.ReadLine());
   Console.Write("z=");
   double z = double.Parse(Console.ReadLine());
   double max;
   if (x > y  && x>z)
   {
      max=x;
   }
   else
   {
      if (y>z)
      {
         max=y;
      }
      else
      {
         max=z;
      }
   }
      Console.WriteLine("max= {0}", max);
  }
```

Задания.

1. Измените программу так, чтобы вычислялось наименьшее значение из трех вещественных чисел.
2. Решите данную задачу используя метод Math.Min(x,y)