Операции классов

Язык C# предоставляет возможности перегрузки некоторых следующих операций:

Операция C#	**Возможность перегрузки**
+, -, !, ++, --, true, false	Этот набор унарных операций может быть перегружен
+, -, *, /, %, &, |, ^, <<, >>	Эти бинарные операции могут быть перегружены
==, !=, <, >, <=, >=	Эти операции сравнения могут быть перегружены. C# требует совместной перегрузки "подобных" операций (т.е. < и >, <= и >=, == и !=)

Перегрузка операторов тесно связана с перегрузкой методов.
Для перегрузки оператора служит ключевое слово **operator**, определяющее операторный метод, который, в свою очередь, определяет действие оператора относительно своего класса.

Существуют две формы операторных методов (operator):

- для унарных операторов,
- для бинарных операторов.

```
// Общая форма перегрузки унарного оператора.
public static возвращаемый_тип operator op(тип_параметра операнд)
{
        // операции
}
// Общая форма перегрузки бинарного оператора.
public static возвращаемый_тип operator op(тип_параметра1 операнд1, тип_параметра2 операнд2)
{
        // операции
}
```

Здесь:

- вместо op подставляется перегружаемый оператор, например + или /,
- возвращаемый_тип обозначает конкретный тип значения, возвращаемого указанной операцией. Это значение может быть любого типа, но обычно оно указывается такого же типа, как и у класса, для которого перегружается оператор.

Для унарных операторов операнд обозначает передаваемый операнд, а для бинарных операторов то же самое обозначают операнд1 и операнд2.

Операторные методы должны иметь оба спецификатора типа - public и static.

Пример перегрузки операторов сравнения в классе Counter (счетчик со значением Value):

```
class Counter
{
   public int Value { get; set; }

   public static Counter operator +(Counter c1, Counter c2)
   {
      return new Counter { Value = c1.Value + c2.Value };
   }
   public static bool operator >(Counter c1, Counter c2)
   {
      return c1.Value > c2.Value;
   }
   public static bool operator <(Counter c1, Counter c2)
   {
      return c1.Value < c2.Value;
   }
}
```

Пример вызова перегруженных операторов:

```
static void Main(string[] args)
{
   Counter c1 = new Counter { Value = 23 };
   Counter c2 = new Counter { Value = 45 };
   bool result = c1 > c2;
   Console.WriteLine(result); // false

   Counter c3 = c1 + c2;
   Console.WriteLine(c3.Value);  // 23 + 45 = 68

   Console.ReadKey();
}
```

Пример перегрузки ++:

Оператор не должен менять значения своих параметров, поэтому для изменения значения объекта в параметрах нужно создавать новый объект и возвращать его с измененным значением:

```
public static Counter operator ++(Counter c1)
{
   return new Counter { Value = c1.Value + 10 };
}
```

Пример вызова ++:

```
Counter counter = new Counter() { Value = 10 };
Console.WriteLine($"{counter.Value}");      // 10
Console.WriteLine($"{(++counter).Value}");  // 20
Console.WriteLine($"{counter.Value}");      // 20
```

Пример перегрузки true-false в классе Counter:

```
class Counter
{
   public int Value { get; set; }

   public static bool operator true(Counter c1)
   {
      return c1.Value != 0;
   }
   public static bool operator false(Counter c1)
   {
      return c1.Value == 0;
   }

   // остальное содержимое класса
}
```

true-false перегружаются, когда мы хотим использовать объект типа в качестве условия. Например:

```
Counter counter = new Counter() { Value = 0 };
if (counter)
{
   Console.WriteLine(true);
}
else
{
   Console.WriteLine(false);
}
```