Перегрузка операторов

- Язык C# позволяет определить значение оператора относительно создаваемого класса. Этот процесс называется перегрузкой операторов. Перегружая оператор, вы расширяете его использование для класса.
- При перегрузке оператора ни одно из его исходных значений не теряется. Перегрузку оператора можно расценивать как введение новой операции для класса. Следовательно, перегрузка оператора “+”, например, для обработки связного списка (в качестве оператора сложения) не изменяет его значение применительно к целым числам. Главное достоинство перегрузки операторов состоит в том, что она позволяет интегрировать новый тип класса со средой программирования.

Основы перегрузки операторов

Перегрузка операторов тесно связана с перегрузкой методов. Для перегрузки операторов используется ключевое слово operator, позволяющее создать операторный метод, который определяет действие оператора, связанное с его классом. Существует две формы методов operator: одна используется для унарных операторов, а другая — для бинарных. Общий же формат (для обоих случаев) таков:

```
// Общий формат перегрузки для унарного оператора.
public static тип_возврата operator op(тип_параметра операнд)
{  // операции }
// Общий формат перегрузки для бинарного оператора.
public static тип_возврата operator op(
 тип_параметра1 операнд1, тип_параметра2 операнд2)
{  // операции }
```

Здесь элемент op - это оператор (например “+” или “/”), который перегружается. Элемент тип_возврата — это тип значения, возвращаемого при выполнении заданной операции. Несмотря на то что можно выбрать любой тип, тип возвращаемого значения чаще всего будет совпадать с типом класса, для которого этот оператор перегружается

Перегрузка бинарных операторов

```
using System;
class ThreeD {
 int x, y, z; // 3-х-мерные координаты.
 public ThreeD() {  x = y = z = 0;  }
  public ThreeD(int i, int j, int k) {  x = i;  y = j;  z = k;  }
  // Перегрузка бинарного оператора "+".
 public static ThreeD operator +(ThreeD op1,  ThreeD op2)
 {  ThreeD result = new ThreeD();
result.x = op1.x + op2.x; // Эти операторы выполняют
 result.y = op1.y + op2.y; // целочисленное сложение.
 result.z = op1.z + op2.z;
 return result;  }
  // Перегрузка бинарного оператора "-".
 public static ThreeD operator -(ThreeD op1, ThreeD op2) {
 ThreeD result = new ThreeD();
 /* Обратите внимание на порядок операндов.  op1 - левый операнд, op2 - правый. */
 result.x = op1.x - op2.x; // Эти операторы выполняют
 result.y = op1.y - op2.y; // целочисленное вычитание.
 result.z = op1.z - op2.z;
 return result;  }
public void show() {
 Console.WriteLine(x + ", " + y + ", " + z);  } }
```

```
class ThreeDDemo {
 public static void Main() {
 ThreeD a = new ThreeD(1, 2, 3);
 ThreeD b = new ThreeD(10, 10, 10);
 ThreeD c = new ThreeD();
  Console.Write("Координаты точки a: ");  a.show();
 Console.WriteLine();  Console.Write("Координаты точки b: ");
 b.show();  Console.WriteLine();
  c = a + b; // Складываем a и b.
 Console.Write("Результат сложения a + b: ");  c.show();
 Console.WriteLine();
  c = a + b + c; // Складываем a, b и c.
 Console.Write("Результат сложения a + b + c: ");  c.show();
 Console.WriteLine();
  c = c - a; // Вычитаем a из c.
 Console.Write("Результат вычитания c - a: ");  c.show();
 Console.WriteLine();
  c = c - b; // Вычитаем b из c.
 Console.Write("Результат вычитания c - b: ");  c.show();
 Console.WriteLine();  } }
```

При выполнении эта программа генерирует следующие результаты:

Координаты точки a: 1, 2, 3

Координаты точки b: 10, 10, 10

Результат сложения a + b: 11, 12, 13

Результат сложения a + b + c: 22, 24, 26

Результат вычитания c - a: 21, 22, 23

Результат вычитания c - b: 11, 12, 13

Перегрузка унарных операторов

Унарные операторы перегружаются точно так же, как и бинарные. Главное отличие состоит в том, что в этом случае существует только один операнд. Рассмотрим, например, метод, который перегружает унарный “минус” для класса ThreeD.

```
// Перегрузка унарного оператора "-".
public static ThreeD operator -(ThreeD op) {
 ThreeD result = new ThreeD();
 result.x = -op.x;
 result.y = -op.y;
 result.z = -op.z;
  return result;
}
```

Созданный таким образом объект и возвращается операторным методом operator-(). Обратите внимание на то, что сам операнд остается немодифицированным. Такое поведение соответствует обычному действию унарного “минуса”. Например, в выражении

```
a = -b
```

```
// Перегрузка унарного оператора "++".
public static ThreeD operator ++(ThreeD op){
 // Оператор "++" модифицирует аргумент.
 op.x++;
 op.y++;
 op.z++;
 return op;
}
```

Выполнение операций над значениями встроенных C#-типов

Для любого заданного класса и оператора любой операторный метод сам может перегружаться. Одна из самых распространенных причин этого — разрешить операции между объектами этого класса и другими (встроенными) типами данных.

```
// Перегружаем "+" для суммирования объекта и int-значения.
public static ThreeD operator +(ThreeD op1, int op2)
{  ThreeD result = new ThreeD();
  result.x = op1.x + op2;
 result.y = op1.y + op2;
 result.z = op1.z + op2;
 return result; }
```

Метод operator+(ThreeD, int) позволяет выполнять инструкции, подобные следующей.

```
ob1 = ob2 + 10;
```

Но он не позволяет выполнять инструкции такого рода:

```
ob1 = 10 + ob2;
```

Чтобы сделать допустимыми две формы инструкций, необходимо перегрузить оператор “+” еще раз. Новая версия должна будет в качестве первого параметра принимать значение типа int, а в качестве второго — объект типа ThreeD.

Перегрузка операторов отношений

```
// Перегрузка операторов "<" и ">".
using System;
// Класс трехмерных координат.
class ThreeD {
 int x, y, z; // 3-х-мерные координаты.
 public ThreeD() {  x = y = z = 0;  }
  public ThreeD(int i, int j, int k) {  x = i;  y = j;  z = k;  }
  // Перегрузка оператора "<".
 public static bool operator <(ThreeD op1, ThreeD op2){
 if((op1.x < op2.x) && (op1.y < op2.y) &&  (op1.z < op2.z))  return true;
else  return false;  }
  // Перегрузка оператора ">".
 public static bool operator >(ThreeD op1, ThreeD op2){
 if((op1.x > op2.x) && (op1.y > op2.y) &&
 (op1.z > op2.z))  return true;  else  return false;  }
public void show()  {  Console.WriteLine(x + ", " + y + ", " + z); }
}
```

```
class ThreeDDemo {
 public static void Main() {
 ThreeD a = new ThreeD(5, 6, 7);
 ThreeD b = new ThreeD(10, 10, 10);
 ThreeD c = new ThreeD(1, 2, 3);
  Console.Write("Координаты точки a: ");  a.show();
 Console.Write("Координаты точки b: ");  b.show();
 Console.Write("Координаты точки c: ");  c.show();
 Console.WriteLine();
  if(a > c) Console.WriteLine("a > c - ИСТИНА");
 if(a < c) Console.WriteLine("a < c - ИСТИНА");
 if(a > b) Console.WriteLine("a > b - ИСТИНА");
 if(a < b) Console.WriteLine("a < b - ИСТИНА");
 }
}
```

При выполнении эта программа генерирует результаты:

Координаты точки a: 5, б, 7

Координаты точки b: 10, 10, 10

Координаты точки c: 1, 2, 3

a > c - ИСТИНА a < b - ИСТИНА

На перегрузку операторов отношений налагается серьезное ограничение: их следует перегружать парами. Например, перегружая оператор “<”, вы также должны перегрузить оператор “>”, и наоборот. Пары операторов отношений:

== !=

< >

<= >=

Перегружая операторы “==” и “!=”, следует перегрузить также методы Object.Equals() и Object.GetHashCode(). Эти методы (а также их перегрузка) рассматриваются далее.

Перегрузка операторов true и false

Ключевые слова true и false в целях перегрузки также можно использовать в качестве унарных операторов. Перегруженные версии этих операторов обеспечивают специфическое определение понятий ИСТИНА и ЛОЖЬ в отношении создаваемых программистом классов. Если для класса реализовать таким образом ключевые слова true и false, то затем объекты этого класса можно использовать для управления инструкциями if, while, for и do-while, а также в ?-выражении. Их можно даже использовать для реализации специальных типов логики (например, нечеткой логики).

Операторы true и false должны быть перегружены в паре. Оба они выполняют функцию унарных операторов и имеют такой формат:

```
public static bool operator true(тип_параметра op)
{
 // Возврат значения true или false.
}
public static bool operator false(тип_параметра op) {
 // Возврат значения true или false.
}
```

Предполагается, что ThreeD-объект истинен, если по крайней мере одна его координата не равна нулю. Если все три координаты равны нулю, объект считается ложным. В целях демонстрации здесь также реализован оператор декремента.

```
// Перегрузка операторов true и false для класса ThreeD.
using System;
// Класс трехмерных координат.
class ThreeD {
 int x, y, z; // 3-х-мерные координаты.
 public ThreeD() {  x = y = z = 0;  }
 public ThreeD(int i, int j, int k) {  x = i;  y = j;  z = k;  }
 // Перегружаем оператор true.
 public static bool operator true(ThreeD op) {
 if((op.x !=0) || (op.y != 0) || (op.z != 0))
 return true; // Хотя бы одна координата не равна 0.
 else  return false;  }
```

```
 // Перегружаем оператор false.
 public static bool operator false(ThreeD op) {
 if((op.x == 0) && (op.y == 0) && (op.z == 0))
 return  true; // Все координаты равны нулю.
 else  return false;  }
 // Перегружаем унарный оператор "--".
 public static ThreeD operator --(ThreeD op) {  op.x--;
    op.y--; op.z--;
 return op;  }
 // Отображаем координаты X, Y, Z.
public void show() {
 Console.WriteLine(x + ", " + y + ", " + z);  }
}
```

```
class TrueFalseDemo {
 public static void Main() {
 ThreeD a = new ThreeD(5, 6, 7);
 ThreeD b = new ThreeD(10, 10, 10);
 ThreeD c = new ThreeD(0, 0, 0);
 Console.Write("Координаты точки a: ");  a.show();
 Console.Write("Координаты точки b: ");  b.show();
 Console.Write("Координаты точки c: ");  c.show();
 Console.WriteLine();
  if(a) Console.WriteLine("a - это ИСТИНА.");
  else Console.WriteLine("a - это ЛОЖЬ.");
 if(b) Console.WriteLine("b - это ИСТИНА.");
else Console.WriteLine("b - это ЛОЖЬ.");
 if(c) Console.WriteLine("c - это ИСТИНА.");
 else Console.WriteLine("c - это ЛОЖЬ.");
 Console.WriteLine();
 Console.WriteLine( "Управляем циклом, используя объект класса
     ThreeD.");
 do {  b.show();  b--;  } while(b);  } }
```

Вот какие результаты генерирует эта программа:
Координаты точки a: 5, 6, 7
Координаты точки b: 10, 10, 10
Координаты точки c: 0, 0, 0
a - это ИСТИНА.
b - это ИСТИНА.
c - ЭТО ЛОЖЬ.
Управляем циклом, используя объект класса ThreeD.
10, 10, 10
9, 9, 9
8, 8, 8
7, 7, 7
6, 6, 6
5, 5, 5
4, 4, 4
3, 3, 3
2, 2, 2
1, 1, 1

Перегрузка логических операторов

В C# определены следующие логические операторы: &, |, !, && и ||. Перегруженными могут быть только &, |, !. Однако при соблюдении определенных правил можно использовать и операторы && и ||, действующие по сокращенной схеме.

- Если вы не планируете использовать логические операторы, работающие по сокращенной схеме, то можете перегружать операторы & и | по своему усмотрению, причем каждый вариант должен возвращать результат типа bool. Перегруженный оператор ! также, как правило, возвращает результат типа bool.

```
// Простой способ перегрузки операторов !, | и &  для
    класса ThreeD.
using System;
// Класс трехмерных координат.
class ThreeD {
 int x, y, z; // 3-х-мерные координаты.
 public ThreeD() {  x = y = z = 0;  }
 public ThreeD(int i, int j, int k) {  x = i;  y = j;  z = k;  }
 // Перегрузка оператора "|".
 public static bool operator |(ThreeD op1, ThreeD op2){
 if( ((op1.x != 0) || (op1.y != 0) || (op1.z != 0)) |
 ((op2.x != 0) || (op2.y != 0) || (op2.z != 0)) )
 return true;  else  return false;  }
```

```
 // Перегрузка оператора "&".
 public static bool operator &(ThreeD op1, ThreeD op2)
{  if( ((op1.x != 0) && (op1.y != 0) && (op1.z != 0)) &
 ((op2.x != 0) && (op2.y != 0) && (op2.z != 0)) )  return true;
 else  return false;  }
// Перегрузка оператора "!".
 public static bool operator !(ThreeD op)
 {  if((op.x != 0) || (op.y != 0) || (op.z != 0))
 return false;
 else return true;  }
 // Отображаем координаты X, Y, Z.
 public void show()
 {  Console.WriteLine(x + ", " + y + ", " + z); }
}
```

```
class TrueFalseDemo {
 public static void Main() {
 ThreeD a = new ThreeD(5, 6, 7);
 ThreeD b = new ThreeD(10, 10, 10);
 ThreeD c = new ThreeD(0, 0, 0);
 Console.Write("Координаты точки a: ");  a.show();
 Console.Write("Координаты точки b: ");  b.show();
 Console.Write("Координаты точки c: ");  c.show();
 Console.WriteLine();
 if(!a) Console.WriteLine("a - ЛОЖЬ.");
 if(!b) Console.WriteLine("b - ЛОЖЬ.");
 if(!c) Console.WriteLine("c - ложь.");
 Console.WriteLine();
 if(a & b) Console.WriteLine("a & b - ИСТИНА.");  else Console.WriteLine("a & b - ЛОЖЬ.");
 if(a & c) Console.WriteLine("a & c - ИСТИНА.");  else Console.WriteLine("a & c - ЛОЖЬ.");
 if(a | b) Console.WriteLine("a | b - ИСТИНА.");  else Console.WriteLine("a | b - ЛОЖЬ.");
 if(a | c) Console.WriteLine("a | c - ИСТИНА.");  else Console.WriteLine("a | c - ЛОЖЬ.");
 }
}
```

При выполнении эта программа генерирует следующие результаты:

Координаты точки a: 5, 6, 7

Координаты точки b: 10, 10, 10

Координаты точки c: 0, 0, 0

c - ЛОЖЬ.

a & b - ИСТИНА.

a & c - ЛОЖЬ.

a | b - ИСТИНА.

a | c - ИСТИНА.

Чтобы иметь возможность использовать операторы && и ||, действующие по сокращенной схеме вычислений, необходимо соблюдать четыре правила. Во-первых, класс должен перегружать операторы & и |. Во-вторых, &- и | -методы должны возвращать объект класса, для которого перегружаются эти операторы. В-третьих, каждый параметр должен представлять собой ссылку на объект класса, для которого перегружаются эти операторы. В-четвертых, тот же класс должен перегружать операторы true и false. При соблюдении всех этих условий операторы сокращенной схемы действия автоматически становятся доступными для применения.

```
/* Более удачный способ реализации
   операторов !, | и & для класса ThreeD. Эта
   версия автоматически делает
   работоспособными операторы && и ||. */
using System;
// Класс трехмерных координат.
class ThreeD {
 int x, y, z; // 3-х-мерные координаты.
 public ThreeD() {  x = y = z = 0;  }
 public ThreeD(int i, int j, int k) {  x = i;  y = j;  z = k;
 }
```

```
// Перегружаем оператор "|" для вычислений по
 // сокращенной схеме.
 public static ThreeD operator |(ThreeD op1, ThreeD op2){
 if( ((op1.x != 0) || (op1.y != 0) || (op1.z != 0)) |
 ((op2.x != 0) || (op2.y != 0) || (op2.z != 0)) )
 return new ThreeD(1, 1, 1);  else  return new ThreeD(0, 0, 0);  }
// Перегружаем оператор "&" для вычислений по
 // сокращенной схеме.
 public static ThreeD operator &(ThreeD op1, ThreeD op2) {
 if( ((op1.x != 0) && (op1.y != 0) && (op1.z != 0)) &
 ((op2.x != 0) && (op2.y != 0) && (op2.z != 0)) )
 return new ThreeD(1, 1, 1);
 else  return new ThreeD(0, 0, 0);  }

 // Перегружаем оператор "!".
 public static bool operator !(ThreeD op)
 {  if(op) return false;
 else return true;  }
```

```
// Перегружаем оператор true.
 public static bool operator true(ThreeD op) {
 if((op.x != 0) || (op.y != 0) || (op.z != 0))
 return true; // Хотя бы одна координата не  равна нулю.
 else  return false;  }
 // Перегружаем оператор false.
 public static bool operator false(ThreeD op) {
 if((op.x == 0) && (op.y == 0) && (op.z == 0))
 return true; // Все координаты равны нулю.
 else  return false;  }
 // Отображаем координаты X, Y, Z.
 public void show()
 {  Console.WriteLine(x + ", " + y + ", " + z);  }
}
```

```
class TrueFalseDemo {
 public static void Main() {
 ThreeD a = new ThreeD(5, 6, 7);
 ThreeD b = new ThreeD(10, 10, 10);
 ThreeD c = new ThreeD(0, 0, 0);
 Console.Write("Координаты точки a: ");  a.show();
 Console.Write("Координаты точки b: ");  b.show();
 Console.Write("Координаты точки c: ");  c.show();
 Console.WriteLine();
if(a) Console.WriteLine("a - ИСТИНА.");  if(b) Console.WriteLine("b - ИСТИНА.");
 if(c) Console.WriteLine("c - ИСТИНА.");   if(!a) Console.WriteLine("a - ЛОЖЬ.");
 if(!b) Console.WriteLine("b - ЛОЖЬ.");  if(!c) Console.WriteLine("c - ЛОЖЬ.");
 Console.WriteLine();   Console.WriteLine("Используем операторы & и !");
  if(a & b) Console.WriteLine("a & b - ИСТИНА.");  else Console.WriteLine("a & b - ЛОЖЬ.");
 if(a & c) Console.WriteLine("a & c - ИСТИНА.");  else Console.WriteLine("a & c - ЛОЖЬ.");
  if(a | b) Console.WriteLine("a | b - ИСТИНА.");  else Console.WriteLine("a | b - ЛОЖЬ.");
 if(a | c) Console.WriteLine("a | c - ИСТИНА.");  else Console.WriteLine("a | c - ЛОЖЬ.");
 Console.WriteLine();
// Теперь используем операторы &.& и ||, действующие  по сокращенной схеме вычислений.
 Console.WriteLine(  "Используем \"сокращенные\" операторы && и ||");
  if(a && b) Console.WriteLine("a && b - ИСТИНА.");  else Console.WriteLine("a && b - ЛОЖЬ.");
 if(a && c) Console.WriteLine("a && c - ИСТИНА.");  else Console.WriteLine("a && c - ЛОЖЬ.");
  if(a || b) Console.WriteLine("a || b - ИСТИНА.");  else Console.WriteLine("a || b - ЛОЖЬ.");
 if(a || c) Console.WriteLine("a || c - ИСТИНА.");  else Console.WriteLine("a || c - ЛОЖЬ.");  } }
```

Эта программа при выполнении генерирует такие результаты:
Координаты точки a: 5, 6, 7
Координаты точки b: 10, 10, 10
Координаты точки c: 0, 0, 0
a - ИСТИНА.
b - ИСТИНА.
c - ЛОЖЬ.
Используем операторы & и |
a & b - ИСТИНА.
a & c - ЛОЖЬ.
a | b - ИСТИНА.
a | c - ИСТИНА.
Используем "сокращенные" операторы && и ||
a && b - ИСТИНА.
a && c - ЛОЖЬ.
a || b - ИСТИНА.
a || c - ИСТИНА.