Лекция 38

Перегрузка операторов

Перегрузка операторов

В языке C# допускается определять назначение оператора по отношению к создаваемому классу. Этот процесс называется перегрузкой операторов.

Когда оператор перегружается, ни одно из его первоначальных назначений не теряется. Он просто выполняет еще одну, новую операцию относительно конкретного объекта.

Поэтому перегрузка оператора +, например, для обработки связного списка не меняет его назначение по отношению к целым числам, т.е. к их сложению.

Перегрузка операторов

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**): одна — **для унарных операторов**, другая — **для бинарных**.

Перегрузка 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
}
```

op - перегружаемый оператор, например + или /.

Перегрузка операторов

Возвращаемый_тип может быть любым, но зачастую это тип класса, для которого перегружается оператор.

Операторные методы должны иметь оба модификатора, **public** и **static**.

Тип операнда унарных операторов должен быть таким же, как и у класса, для которого перегружается оператор. А в бинарных операторах хотя бы один из операндов должен быть такого же типа, как и у его класса.

В параметрах оператора **нельзя** использовать модификатор **ref** или **out**.

Пример 1

```
using System;
class ThreeD {  // Класс для хранения трехмерных координат
    int x, y, z; // Трехмерные координаты
    public ThreeD() { x = y = z = 0; }
    public ThreeD(int i, int j, int k) { x = i; y = j; z = k; }
    public static ThreeD operator +(ThreeD op1, ThreeD
   op2) {  // Перегрузить бинарный оператор +
        ThreeD result = new ThreeD();
    /* Сложить координаты двух точек и возвратить результат */
        result.x = op1.x + op2.x;
        result.y = op1.y + op2.y;
        result.z = op1.z + op2.z;
        return result;
    }
```

Пример 1

```
// Перегрузить бинарный оператор -
public static ThreeD operator -(ThreeD op1, ThreeD op2) {
    ThreeD result = new ThreeD();
    /* Важен порядок следования операндов:
       op1 — левый операнд, op2 — правый операнд */
    result.x = op1.x - op2.x;
    result.y = op1.y - op2.y;
    result.z = op1.z - op2.z;
    return result;
}
```

Пример 1

```
// Перегрузить унарный оператор -
public static ThreeD operator -(ThreeD op) {
    ThreeD result = new ThreeD();
    result.x = -op.x;          result.y = -op.y;
    result.z = -op.z;          return result;
}
// Перегрузить префиксный унарный оператор ++
public static ThreeD operator ++(ThreeD op) {
    ThreeD result = new ThreeD();
    // Возвратить результат инкрементирования
    result.x = op.x + 1;          result.y = op.y + 1;
    result.z = op.z + 1;          return result;
}
```

Пример 1

```
    public void Show() {    // Вывести координаты X, Y, Z
        Console.WriteLine(x + ", " + y + ", " + z);
    }
}
class ThreeDDemo {
    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.Write("Координаты точки b: ");
        b. Show ();
```

Пример 1

```
c = a + b;     // Сложить координаты точек a и b
Console.Write("Результат сложения a + b: ");
c . Show ();
c=a+b+c;     // Сложить координаты точек a, b и c
Console.Write("Результат сложения a + b + c: ");
c.Show();
c = c - a;      // Вычесть координаты точки a
Console.Write("Результат вычитания c - a: ");
c.Show();
c = c - b;      // Вычесть координаты точки b
Console.Write("Результат вычитания c - b: ");
c.Show();
c = -a;        // Присвоить точке c
               // отрицательные координаты точки a
```

Пример 1

```
Console.Write("Результат присваивания -a: ");
c.Show();
c = a++;  // Присвоить точке с координаты точки a,
          // а затем инкрементировать их
Console.WriteLine("Если c = a++");
Console.Write("то координаты точки c равны ");
c.Show();
Console.Write("а координаты точки a равны ");
a.Show();
a = new ThreeD(1, 2, 3);  // Установить исходные
                          // координаты (1,2,3) точки a
Console.Write("\nУстановка исходных координат точки a: ");
```

Пример 1

```
        a.Show();
        c = ++a;    // Инкрементировать координаты
                    // точки a, а затем присвоить их точке c
        Console.WriteLine("\nЕсли c = ++a");
        Console.Write("то координаты точки c равны ");
        c.Show();
       Console.Write("а координаты точки a равны ");
       a.Show();
    }
}
```

Пример 1

Результат выполнения программы:

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
Результат присваивания -a: -1, -2, -3
Если c = a++
то координаты точки c равны 1, 2, 3
а координаты точки a равны 2, 3, 4
Установка исходных координат точки a: 1, 2, 3
Если c = ++a
то координаты точки c равны 2, 3, 4
а координаты точки a равны 2, 3, 4

Пример 2

```
/* Перегрузить бинарный оператор + трижды: один раз — для
   сложения объектов класса ThreeD, второй раз — для сложения
   объекта типа ThreeD и целого значения типа int, а третий раз
   — для сложения целого значения типа int и объекта типа
   ThreeD */
using System;
class ThreeD {// Класс для хранения трехмерных координат
    int x, y, z; // Трехмерные координаты
    public ThreeD() { x = y = z = 0; }
    public ThreeD(int i, int j, int k) {
        x = i; y = j; z = k;
    }
```

Пример 2

```
// Перегрузить бинарный оператор + для
// сложения объектов класса ThreeD
public static ThreeD operator +(ThreeD op1,
ThreeD op2) {
    ThreeD result = new ThreeD();
    /* Сложить координаты двух точек и
возвратить результат */
    result.x = op1.x + op2.x;
    result.y = op1.y + op2.y;
    result.z = op1.z + op2.z;
    return result;
}
```

Пример 2

```
// Перегрузить бинарный оператор + для сложения
// объекта типа ThreeD и целого значения типа int
   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;
    }
```

Пример 2

```
// Перегрузить бинарный оператор + для сложения
// целого значения типа int и объекта типа ThreeD
    public static ThreeD operator +(int op1, ThreeD op2) {
        ThreeD result = new ThreeD();
        result.x = op2.x + op1;
        result.y = op2.y + op1;
        result.z = op2.z + op1;
        return result;
    }
    public void Show() { // Вывести координаты X, Y, Z
            Console.WriteLine(x + ", " + y + ", " + z) ;
     }
}
```

Пример 2

```
class ThreeDDemo {
    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.Write("Координаты точки b: ");
          b. Show ();
          c = a + b; // Сложить объекты класса ThreeD
          Console.Write("Результат сложения a + b: ");
          c.Show();
```

Пример 2

```
        c=b+10;   // Сложить объект типа ThreeD
                  // и целое значение типа int
        Console.Write("Результат сложения b + 10: ");
       c.Show();
        Console.WriteLine();
        c=15+b;  // Сложить целое значение типа
                  // int и объект типа ThreeD
        Console.Write("Результат сложения 15 + b: ");
        c.Show();
    }
}
```

Пример 2

Выполнение этого кода дает следующий результат:

Координаты точки a: 1, 2, 3

Координаты точки b: 10, 10, 10

Результат сложения a + b: 11, 12, 13

Результат сложения b + 10: 20, 20, 20

Результат сложения 15 + b: 25, 25, 25

Перегрузка операторов

На перегрузку операторов отношения накладывается следующее важное **ограничение**: они должны перегружаться попарно.

Так, если перегружается оператор <, то следует перегрузить и оператор >, и наоборот. Ниже приведены составленные в пары перегружаемые операторы отношения.

==	!=
<	>
<=	>=

Пример 3

```
using System;
class ThreeD {
    int x, y, z;    // Трехмерные координаты
    public ThreeD() { x = y = z = 0; }
    public ThreeD(int i, int j, int k) {
        x = i; y = j; z = k;
    }
    public static bool operator <(ThreeD op1,
   ThreeD op2) {   // Перегрузить оператор <
        if (Math.Sqrt(op1.x* op1.x + op1.y * op1.y +
   op1.z * op1.z) < Math.Sqrt(op2.x * op2.x + op2.y
   * op2.y + op2.z * op2.z)) return true;
        else return false;
}
```

// сравнивается расстояние от начала координат до точки с координатами x, y, z

Пример 3

```
  public static bool operator >(ThreeD op1,
ThreeD op2) {   // Перегрузить оператор >
      if (Math.Sqrt(op1.x * op1.x + op1.y * op1.y +
op1.z * op1.z) > Math.Sqrt(op2.x * op2.x + op2.y
* op2.y + op2.z * op2.z)) return true;
     else return false;
   }
   public void Show() { // Вывести координаты X, Y, Z
       Console.WriteLine(x + ", " + y + ", " + z);
   }
}
```

Пример 3

```
class ThreeDDemo {
    static void Main() {
        ThreeD a = new ThreeD(5, 6, 7);
        ThreeD b = new ThreeD(10, 10, 10);
        ThreeD c = new ThreeD(1, 2, 3);
        ThreeD d = new ThreeD(6, 7, 5);
         Console.Write("Координаты точки a: ")
         a.Show();
          Console.Write("Координаты точки b: ")
         b.Show();
         Console.Write("Координаты точки c: ")
          c.Show();
```

Пример 3

```
        Console.Write("Координаты точки d: ")
        d. Show ();
        Console.WriteLine();
      if (a > c) Console.WriteLine("a > c истинно");
      if (a < c) Console.WriteLine("a < c истинно");
      if (a > b) Console.WriteLine("a > b истинно");
      if (a < b) Console.WriteLine("a < b истинно");
      if(a > d) Console.WriteLine("a > d истинно");
      else if (a < d) Console.WriteLine("a < d истинно");
            else Console.WriteLine("Точки a и d находятся
на одном расстоянии " + "от начала отсчета");
    }
}
```

Пример 3

Результат выполнения кода:

Координаты точки a: 5, 6, 7

Координаты точки b: 10, 10, 10

Координаты точки c: 1, 2, 3

Координаты точки d: б, 7, 5

a > c истинно

a < b истинно

Точки a и d находятся на одном расстоянии от начала отсчета

Перегрузка операторов

Ключевые слова **true** и **false** можно также использовать в качестве унарных операторов для целей перегрузки, для того, чтобы объекты этого класса могли использоваться операторами **if**, **while**, **for** и **do-while** или же в условном выражении **?:**.

Операторы **true** и **false** должны перегружаться попарно.

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

В любом другом случае возвращается результат типа **bool**.

Пример 4

```
using System;
class ThreeD {
      int x, y, z;        // Трехмерные координаты
       public ThreeD() { x = y = z = 0; }
       public ThreeD(int i, int j, int k) {
            x = i; y = j; z = k;
      }
       public static bool operator true(ThreeD op) {
          // Перегрузить оператор true
          if ((op.x != 0) || (op.y != 0) || (op.z != 0))
              return true; // Хотя бы одна координата не равна нулю
```

Пример 4

```
    else
        return false;
}
public static bool operator false(ThreeD op) {
  // Перегрузить оператор false
  if ((op.x == 0) && (op.y == 0) && (op.z == 0))
        return true; // Все координаты равны нулю
    else
         return false;
}
```

Пример 4

```
public static ThreeD operator --(ThreeD op){
   // Перегрузить унарный оператор --
     ThreeD result = new ThreeD();
     result.x = op.x - 1;
     result.y = op.y - 1;
     result.z = op.z - 1;
     return result;
}
public void Show() { // Вывести координаты X, Y, Z
      Console.WriteLine(x + ", " + y + ", " + z);
}
}
```

Пример 4

```
class TrueFalseDemo {
    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();
```

Пример 4

```
        if (a) Console.WriteLine("Точка a истинна.");
        else Console.WriteLine("Точка a ложна.");
        if (b) Console.WriteLine("Точка b истинна.");
        else Console.WriteLine("Точка b ложна.");
        if (c) Console.WriteLine("Точка c истинна.");
        else Console.WriteLine("Точка c ложна.");
        Console.WriteLine("Управление циклом с
помощью объекта класса ThreeD.");
        do {
            b. Show ();
            b--;
        } while(b);
    }
}
```

Пример 4

Координаты точки a: 5, 6, 7 // Результат
Координаты точки b: 10, 10, 10
Координаты точки c: 0, 0, 0
Точка a истинна
Точка b истинна
Точка c ложна
Управление циклом с помощью объекта класса ThreeD.
10, 10, 10
9, 9, 9
8, 8, 8
1, 1, 1
6, 6, 6
5, 5, 5
4, 4, 4
3, 3, 3
2, 2, 2
1, 1, 1

Перегрузка операторов

В **C#** предусмотрены следующие **логические операторы**: **&**, |, **!**, **&&** и ||. Из них **перегрузке**, безусловно, **подлежат только** операторы **&**, | и **!**. Тем не менее, соблюдая определенные правила, можно извлечь также пользу из укороченных логических операторов **&&** и ||.

Перегрузку операторов **&** и | можно выполнять совершенно естественным путем, получая в каждом случае результат типа **bool**. Аналогичный результат, как правило, дает и перегружаемый оператор **!**.

Пример 5

```
using System;
class ThreeD {
    int x, y, z; // Трехмерные координаты
    public ThreeD() { x = y = z = 0; }
    public ThreeD(int i, int j, int k) {
        x = i; y = j; z = k;
    }
    public static bool operator |(ThreeD op1, ThreeD op2)
    { // Перегрузить логический оператор |
        if ( ((op1.x != 0) || (op1.y != 0) || (op1.z != 0)) |
   ((op2.x != 0) || (op2.y != 0) || (op2.z != 0) ) )
            return true;
        else return false;
    }
```

Пример 5

```
public static bool operator &(ThreeD op1, ThreeD op2)
{     // Перегрузить логический оператор &
  if ( ((op1.x != 0) && (op1.y != 0) && (op1.z != 0))
&((op2.x != 0) && (op2.y != 0) && (op2.z != 0)) )
        return true;
  else return false;
}
public static bool operator !(ThreeD op) {// Оператор !
   if ((op.x != 0) || (op.y != 0) || (op.z != 0))   return false;
   else return true;
}
public void Show() { // Вывести координаты X, Y, Z
    Console.WriteLine(x + ", " + y + ", " + z);  }
}
```

Пример 5

```
class TrueFalseDemo {
    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();
        if (!a) Console.WriteLine("Точка a ложна.");
        if (!b) Console.WriteLine("Точка b ложна.");
```

Пример 5

```
        if (!c) Console.WriteLine("Точка с ложна.");
        if (a & b) Console.WriteLine("a & b истинно.");
        else Console.WriteLine("a & b ложно.");
        if (a & c) Console.WriteLine("a & c истинно.");
        else Console.WriteLine("a & c ложно.");
        if (a | b) Console.WriteLine("a | b истинно.");
        else Console.WriteLine("a | b ложно.");
        if (a | c) Console.WriteLine("a | c истинно.");
        else Console.WriteLine("a | c ложно.");
    }
}
```

Пример 5

Результат выполнения программы:

Координаты точки a: 5, 6, 7

Координаты точки b: 10, 10, 10

Координаты точки c: 0, 0, 0

Точка c ложна.

a & b истинно.

a & c ложно.

a | b истинно.

a | c истинно.

Перегрузка операторов

Для того чтобы применение укороченных логических операторов **&&** и || стало возможным, необходимо соблюсти следующие **четыре правила**.

Во-первых, в классе должна быть произведена перегрузка логических операторов **&** и |.

Во-вторых, перегружаемые методы операторов **&** и | должны возвращать значение того же типа, что и у класса, для которого эти операторы перегружаются.

В-третьих, каждый параметр должен содержать ссылку на объект того класса, для которого перегружается логический оператор.

И **в-четвертых**, для класса должны быть перегружены операторы **true** и **false**.

Пример 6

```
using System;
class ThreeD {
    int x, y, z; // Трехмерные координаты
    public ThreeD() { x = y = z = 0; }
    public ThreeD(int i, int j, int k) { x = i; y = j; z = k; }
// Перегрузить логический оператор | для укороченного вычисления
    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);
    }
```

Пример 6

```
// Перегрузить логический оператор &
// для укороченного вычисления
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);
}
public static bool operator !(ThreeD op) {
   // Перегрузить логический оператор !
      if (op) return false;
      else return true;
}
```

Пример 6

```
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;
}
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;
}
public void Show() {   // Ввести координаты X, Y, Z
      Console.WriteLine(x + ", " + y + ", " + z) ;
}
}
```

Пример 6

```
class TrueFalseDemo {
    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();
       if (a) Console.WriteLine("Точка a истинна.");
       if (b) Console.WriteLine("Точка b истинна.");
```

Пример 6

```
    if (c) Console.WriteLine("Точка c истинна.");
    if (!a) Console.WriteLine("Точка a ложна.");
    if (!b) Console.WriteLine("Точка b ложна.");
    if (!c) Console.WriteLine("Точка c ложна.");
    Console.WriteLine("Применение логических
операторов & и |");
    if (a & b) Console.WriteLine("a & b истинно.");
    else Console.WriteLine("a & b ложно.");
    if (a & c) Console.WriteLine("a & c истинно.");
    else Console.WriteLine("a & c ложно.");
    if (a | b) Console.WriteLine("a | b истинно.");
    else Console.WriteLine("a | b ложно.");
    if (a | c) Console.WriteLine("a | c истинно.");
    else Console.WriteLine("a | c ложно.");
```

Пример 6

```
        Console.WriteLine("Применение укороченных
    логических операторов && и ||");
        if (a && b) Console.WriteLine("a && b истинно.");
        else Console.WriteLine("a && b ложно.");
        if (a && c) Console.WriteLine("a && c истинно.");
        else Console.WriteLine("a && c ложно.");
        if (a || b) Console.WriteLine ("a || b истинно.");
        else Console.WriteLine("a || b ложно.");
        if (a || c) Console.WriteLine("a || c истинно.");
        else Console.WriteLine("a || c ложно.");
    }
}
```

Перегрузка операторов

Оператор преобразования преобразует объект исходного класса в другой тип.

public static explicit operator целевой_тип (исходный_тип v) {**return** значение;}

public static implicit operator целевой_тип (исходный_тип v) {**return** значение;}

где **целевой_тип** обозначает тот тип, в который выполняется преобразование;

исходный_тип — тот тип, который преобразуется;

значение — конкретное значение, приобретаемое классом после преобразования.

Перегрузка операторов

Если оператор преобразования указан **в неявной форме** (**implicit**), то преобразование вызывается автоматически, например, в том случае, когда объект используется в выражении вместе со значением целевого типа.

Если же оператор преобразования указан **в явной форме** (**explicit**), то преобразование вызывается в том случае, когда выполняется приведение типов.

Для одних и тех же исходных и целевых типов данных **нельзя указывать оператор преобразования одновременно в явной и неявной форме**.

Пример 7

```
using System;
class ThreeD {
    int x, y, z;     // Трехмерные координаты
    public ThreeD() { x = y = z = 0; }
    public ThreeD(int i, int j, int k) { x = i; y = j; z = k; }
    public static ThreeD operator +(ThreeD op1, ThreeD
   op2) {   // Перегрузить бинарный оператор +
        ThreeD result = new ThreeD();
        result.x = op1.x + op2.x;
        result.y = op1.y + op2.y;
        result.z = op1.z + op2.z;
        return result;
    }
```

Пример 7

```
// Неявное преобразование объекта типа ThreeD к типу int
    public static implicit operator int(ThreeD op1) {
        return op1.x * opl.y * op1.z;
    }
    public void Show() {  // Вывести координаты X, Y, Z
        Console.WriteLine(x + ", " + y + ", " + z) ;
    }
}
class ThreeDDemo {
    static void Main() {
        ThreeD a = new ThreeD(1, 2, 3);
        ThreeD b = new ThreeD(10, 10, 10);
        ThreeD c = new ThreeD();
        int i;
```

Пример 7

```
        Console.Write("Координаты точки a: ");
        a.Show();
        Console.WriteLine();
        Console.Write("Координаты точки b: ");
        b.Show();
        c = a + b;        // Сложить координаты точек a и b
        Console.Write("Результат сложения a + b: ");
        c.Show();
        i = a;            // Преобразовать в тип int
        Console.WriteLine("Результат присваивания i = a: " + i);
        Console.WriteLine();
         i = a * 2 - b;    // Преобразовать в тип int
        Console.WriteLine("Результат вычисления выражения a * 2 -
    b: " + i);
     }
}
```

Пример 8

```
using System;
class ThreeD { // Класс для хранения трехмерных
    координат
     int x, y, z; // Трехмерные координаты
     public ThreeD() { x = y = z = 0; }
     public ThreeD(int i, int j , int k) { x = i; y = j; z = k; }
     public static ThreeD operator + (ThreeD op1, ThreeD
    op2) {      // Перегрузить бинарный оператор +
          ThreeD result = new ThreeD();
          result.x = op1.x + op2.x;
          result.y = op1.y + op2.y;
          result.z = op1.z + op2.z;
          return result;
     }
```

Пример 8

```
    public static explicit operator int(ThreeD op1) {
      // Выполнить явное преобразование типов
        return op1.x * op1.y * op1.z;
    }
    public void Show() {  // Вывести координаты X, Y, Z
        Console.WriteLine(x + ", " + y + ", " + z);
    }
}
class ThreeDDemo {
    static void Main() {
        ThreeD a = new ThreeD(l, 2, 3) ;
        ThreeD b = new ThreeD(10, 10, 10);
        ThreeD c = new ThreeD();
        int i;
```

Пример 8

```
        Console.Write("Координаты точки a: ");
        a.Show();
        Console.Write("Координаты точки b: ");
        b.Show();
        c = a + b;     // Сложить координаты точек a и b
         Console.Write("Результат сложения a + b: ");
        c.Show();
         i = (int) a;   // Преобразовать в тип int явно,
                        // поскольку указано приведение типов
        Console.WriteLine("Результат присваивания i = a: " + i);
        i = (int)a * 2 - (int)b; // явно требуется приведение типов
        Console.WriteLine("Результат вычисления выражения a
    * 2 - b: " + i);
     }
}
```

Перегрузка операторов

Ограничения на **операторы преобразования**:

- Исходный или целевой тип преобразования должен относиться к классу, для которого объявлено данное преобразование. В частности, нельзя переопределить преобразование в тип **int**, если оно первоначально указано как преобразование в тип **double**.
- Нельзя указывать преобразование в класс **object** или же из этого класса.
- Для одних и тех же исходных и целевых типов данных нельзя указывать одновременно явное и неявное преобразование.
- Нельзя указывать преобразование базового класса в производный класс.
- Нельзя указывать преобразование в интерфейс или же из него.

Перегрузка операторов

К неявным преобразованиям следует прибегать только в тех случаях, когда преобразованию не свойственны ошибки, т. е. удовлетворяются **следующие условия:**

Во-первых, информация не теряется, например, в результате усечения, переполнения или потери знака.

И **во-вторых**, преобразование не приводит к исключительной ситуации.

Если же неявное преобразование не удовлетворяет этим двум условиям, то следует выбрать явное преобразование.

Перегрузка операторов

Ограничения на **перегрузку операторов**:

- **нельзя изменять приоритет любого оператора или количество операндов**, которое требуется для оператора, хотя в операторном методе можно и проигнорировать операнд;
- **перегрузке не подлежит ни один из операторов присваивания**, в том числе и составные, как, например, оператор +=.
- Нельзя перегружать операторы:

&&	**()**	**.**	**?**
??	**[]**	**||**	**=**
=>	**->**	**as**	**checked**
default	**is**	**new**	**sizeof**
typeof	**unchecked**		

Перегрузка операторов

Ограничения на **перегрузку операторов**:

- **нельзя изменять приоритет любого оператора или количество операндов**, которое требуется для оператора, хотя в операторном методе можно и проигнорировать операнд;
- **перегрузке не подлежит ни один из операторов присваивания**, в том числе и составные, как, например, оператор +=.
- Нельзя перегружать операторы:

&&	**()**	**.**	**?**
??	**[]**	**||**	**=**
=>	**->**	**as**	**checked**
default	**is**	**new**	**sizeof**
typeof	**unchecked**		

Перегрузка операторов

Если перегружаются операторы == и **!=**, то для этого обычно требуется также переопределить методы **Object.Equals**() и **Object.GetHashCode**().

Метод **Equals** (object) возвращает логическое значение **true**, если сравниваемые объекты одинаковы, в противном случае — логическое значение **false**. Он может быть также переопределен в создаваемых классах. Например, метод **Equals** (object) можно определить таким образом, чтобы в нем сравнивалось содержимое двух объектов.

Метод **GetHashCode**() возвращает хеш-код, связанный с вызывающим объектом.

Если перегружается оператор ==, то обычно приходится переопределять методы **Equals**(object) и **GetHashCode**(), поскольку чаще всего требуется, чтобы метод **Equals**(object) и оператор == функционировали одинаково.

Перегрузка операторов

Класс **Object** реализует метод **Equals**() со следующей сигнатурой: **public override** bool Equals(object o)

Перегрузив этот метод, программист позволяет своему классу, например, **Fraction** действовать полиморфно с другими объектами. В теле метода **Equals**() потребуется выполнить сравнение с другим объектом **Fraction**.

```
public override bool Equals(object o) {
    if (! (o is Fraction) ) { // Если типы совместимы
        return false;
    }
    return this == (Fraction) o;
}
```

Пример 9

```
using System;
public class Fraction {
    public Fraction(int numerator, int denominator){
        Console.WriteLine("Конструктор Fraction (int,
   int)");
        this.numerator=numerator;
        this.denominator=denorcinator;
    }
    public Fraction(int wholeNumber) {
        Console.WriteLine("Конструктор Fraction (int)");
        numerator = wholeNumber;
        denominator = 1;
    }
```

Пример 9

```
public static implicit operator Fraction(int theInt) {
    System.Console.WriteLine("Неявное
преобразование в тип Fraction");
    return new Fraction(theInt);
}
public static explicit operator int(Fraction theFraction)
{
     System.Console.WriteLine("Явное преобразование
в тип int");
    return
        theFraction.numerator / theFraction.denominator;
}
```

Пример 9

```
public static bool operator==(Fraction lhs, Fraction rhs)
{
    Console.WriteLine("Операция ==");
     if (lhs.denominator == rhs.denominator &&
lhs.numerator == rhs.numerator) { return true; }
// здесь должно быть реализовано приведение дробей
     return false;
 }
 public static bool operator !=(Fraction lhs, Fraction
rhs) {
      Console.WriteLine("Операция !=");
      return !(lhs==rhs);
 }
```

Пример 9

```
public override bool Equals(object o) {
    Console.WriteLine("Метод Equals");
    if (! (o is Fraction) ){
        return false;
    }
    return this == (Fraction) o;
}
public static Fraction operator+(Fraction lhs, Fraction
rhs) {
    Console.WriteLine("Операция +");
     // Знаменатели сравниваются друг с другом,
     // и если они равны, числители складываются
```

Пример 9

```
        if (lhs.denominator == rhs.denominator) {
            return new
Fraction(lhs.numerator+rhs.numerator, lhs.denominator);
        }
        // Если знаменатели не равны,
        // выполняется перекрестное умножение
         // 1/2 + 3/4 == (1*4 + 3*2) / (2*4) == 10/8
        int firstProduct = lhs.numerator * rhs.denominator;
        int secondProduct = rhs.numerator *
lhs.denominator;
        return new Fraction(firstProduct + secondProduct,
lhs.denominator * rhs.denominator);
    }
```

Пример 9

```
    public override string ToString() {
        String s = numerator.ToString() + "/" +
    denominator.ToString();
        return s;
    }
    private int numerator;
    private int denominator;
}
public class Tester {
   static void Main() {
     Fraction f1 = new Fraction(3, 4); // Создана простая дробь 3/4
     Console.WriteLine("f1: {0}", f1.ToString());
      Fraction f2 = new Fraction(2, 4); // Создана простая дробь 2/4
```

Пример 9

```
        Console.WriteLine("f2: {0}", f2.ToString());
        Fraction f3 = f1 + f2;
        Console.WriteLine("f1 + f2 = f3: {0}",
    f3.ToString());
        Fraction f4 = f3 + 5;
        Console.WriteLine("f3 + 5 = f4: {0}", f4.ToString());
        Fraction f5 = new Fraction(2,4);
        if (f5 == f2) {
            Console.WriteLine("F5: {0} == F2: {1}",
    f5.ToString(), f2.ToString());
        }
     }
}
```

Контрольные вопросы

1. Какие операции можно перегружать в языке C#?
2. Какие ограничения и особенности следует знать при перегрузке операторов отношения?
3. Какие правила следует соблюдать при перегрузке логических операторов && и ||?