tutorialspoint
SIMPLYEASYLEARNING

C# - Operator Overloading

Advertisements

Previous Page Next Page

You can redefine or overload most of the built-in operators available in C#. Thus a programmer can use operators with user-defined types as well. Overloaded operators are functions with special names the keyword **operator** followed by the symbol for the operator being defined. similar to any other function, an overloaded operator has a return type and a parameter list.

For example, go through the following function –

```
public static Box operator+ (Box b, Box c) {
   Box box = new Box();
   box.length = b.length + c.length;
   box.breadth = b.breadth + c.breadth;
   box.height = b.height + c.height;
   return box;
}
```

The above function implements the addition operator (+) for a user-defined class Box. It adds the attributes of two Box objects and returns the resultant Box object.

Implementing the Operator Overloading

The following program shows the complete implementation –

Live Demo

```
using System;

namespace OperatorOvlApplication {
   class Box {
      private double length;   // Length of a box
      private double breadth;  // Breadth of a box
      private double height;   // Height of a box

      public double getVolume() {
         return length * breadth * height;
      }
      public void setLength( double len ) {
         length = len;
      }
      public void setBreadth( double bre ) {
         breadth = bre;
      }
      public void setHeight( double hei ) {
         height = hei;
      }

      // Overload + operator to add two Box objects.
      public static Box operator+ (Box b, Box c) {
         Box box = new Box();
         box.length = b.length + c.length;
         box.breadth = b.breadth + c.breadth;
         box.height = b.height + c.height;
         return box;
      }
   }
   class Tester {
      static void Main(string[] args) {
         Box Box1 = new Box();   // Declare Box1 of type Box
         Box Box2 = new Box();   // Declare Box2 of type Box
         Box Box3 = new Box();   // Declare Box3 of type Box
         double volume = 0.0;    // Store the volume of a box here

         // box 1 specification
         Box1.setLength(6.0);
         Box1.setBreadth(7.0);
         Box1.setHeight(5.0);

         // box 2 specification
         Box2.setLength(12.0);
         Box2.setBreadth(13.0);
         Box2.setHeight(10.0);

         // volume of box 1
         volume = Box1.getVolume();
         Console.WriteLine("Volume of Box1 : {0}", volume);

         // volume of box 2
         volume = Box2.getVolume();
         Console.WriteLine("Volume of Box2 : {0}", volume);

         // Add two object as follows:
         Box3 = Box1 + Box2;

         // volume of box 3
         volume = Box3.getVolume();
         Console.WriteLine("Volume of Box3 : {0}", volume);
         Console.ReadKey();
      }
   }
}
```

When the above code is compiled and executed, it produces the following result −

```
Volume of Box1 : 210
Volume of Box2 : 1560
Volume of Box3 : 5400
```

Overloadable and Non-Overloadable Operators

The following table describes the overload ability of the operators in C# −

Sr.No.	Operators & Description
1	**+, -, !, ~, ++, --** These unary operators take one operand and can be overloaded.
2	**+, -, *, /, %** These binary operators take one operand and can be overloaded.

3	**==, !=, <, >, <=, >=** The comparison operators can be overloaded.
4	**&&, ||** The conditional logical operators cannot be overloaded directly.
5	**+=, -=, *=, /=, %=** The assignment operators cannot be overloaded.
6	**=, ., ?:, ->, new, is, sizeof, typeof** These operators cannot be overloaded.

Example

In the light of the above discussions, let us extend the preceding example, and overload few more operators –

Live Demo

```
using System;

namespace OperatorOvlApplication {
   class Box {
      private double length;    // Length of a box
      private double breadth;   // Breadth of a box
      private double height;    // Height of a box

      public double getVolume() {
         return length * breadth * height;
      }
      public void setLength( double len ) {
         length = len;
      }
      public void setBreadth( double bre ) {
         breadth = bre;
      }
      public void setHeight( double hei ) {
         height = hei;
      }

      // Overload + operator to add two Box objects.
      public static Box operator+ (Box b, Box c) {
         Box box = new Box();
         box.length = b.length + c.length;
         box.breadth = b.breadth + c.breadth;
         box.height = b.height + c.height;
         return box;
      }
      public static bool operator == (Box lhs, Box rhs) {
         bool status = false;
         if (lhs.length == rhs.length && lhs.height == rhs.height
            && lhs.breadth == rhs.breadth) {

            status = true;
         }
         return status;
      }
      public static bool operator !=(Box lhs, Box rhs) {
         bool status = false;

         if (lhs.length != rhs.length || lhs.height != rhs.height ||
            lhs.breadth != rhs.breadth) {

            status = true;
         }
         return status;
      }
      public static bool operator <(Box lhs, Box rhs) {
         bool status = false;

         if (lhs.length < rhs.length && lhs.height < rhs.height
            && lhs.breadth < rhs.breadth) {

            status = true;
         }
         return status;
      }
      public static bool operator >(Box lhs, Box rhs) {
         bool status = false;

         if (lhs.length > rhs.length && lhs.height >
            rhs.height && lhs.breadth > rhs.breadth) {
```

```
            status = true;
         }
         return status;
      }
      public static bool operator <=(Box lhs, Box rhs) {
         bool status = false;
         
         if (lhs.length <= rhs.length && lhs.height
            <= rhs.height && lhs.breadth <= rhs.breadth) {
            
            status = true;
         }
         return status;
      }
      public static bool operator >=(Box lhs, Box rhs) {
         bool status = false;
         
         if (lhs.length >= rhs.length && lhs.height
            >= rhs.height && lhs.breadth >= rhs.breadth) {
            
            status = true;
         }
         return status;
      }
      public override string ToString() {
         return String.Format("({0}, {1}, {2})", length, breadth, height);
      }
   }
   class Tester {
      static void Main(string[] args) {
         Box Box1 = new Box();   // Declare Box1 of type Box
         Box Box2 = new Box();   // Declare Box2 of type Box
         Box Box3 = new Box();   // Declare Box3 of type Box
         Box Box4 = new Box();
         double volume = 0.0;    // Store the volume of a box here
         
         // box 1 specification
         Box1.setLength(6.0);
         Box1.setBreadth(7.0);
         Box1.setHeight(5.0);
         
         // box 2 specification
         Box2.setLength(12.0);
         Box2.setBreadth(13.0);
         Box2.setHeight(10.0);
         //displaying the Boxes using the overloaded ToString():
         Console.WriteLine("Box 1: {0}", Box1.ToString());
         Console.WriteLine("Box 2: {0}", Box2.ToString());

         // volume of box 1
         volume = Box1.getVolume();
         Console.WriteLine("Volume of Box1 : {0}", volume);

         // volume of box 2
         volume = Box2.getVolume();
         Console.WriteLine("Volume of Box2 : {0}", volume);

         // Add two object as follows:
         Box3 = Box1 + Box2;
         Console.WriteLine("Box 3: {0}", Box3.ToString());

         // volume of box 3
         volume = Box3.getVolume();
         Console.WriteLine("Volume of Box3 : {0}", volume);

         //comparing the boxes
         if (Box1 > Box2)
            Console.WriteLine("Box1 is greater than Box2");
         else
            Console.WriteLine("Box1 is not greater than Box2");

         if (Box1 < Box2)
            Console.WriteLine("Box1 is less than Box2");
         else
            Console.WriteLine("Box1 is not less than Box2");

         if (Box1 >= Box2)
            Console.WriteLine("Box1 is greater or equal to Box2");
         else
            Console.WriteLine("Box1 is not greater or equal to Box2");

         if (Box1 <= Box2)
            Console.WriteLine("Box1 is less or equal to Box2");
         else
            Console.WriteLine("Box1 is not less or equal to Box2");

         if (Box1 != Box2)
            Console.WriteLine("Box1 is not equal to Box2");
         else
            Console.WriteLine("Box1 is not greater or equal to Box2");
         Box4 = Box3;
         if (Box3 == Box4)
            Console.WriteLine("Box3 is equal to Box4");
         else
            Console.WriteLine("Box3 is not equal to Box4");

         Console.ReadKey();
      }
   }
}
```

When the above code is compiled and executed, it produces the following result −

```
Box 1: (6, 7, 5)
Box 2: (12, 13, 10)
Volume of Box1 : 210
Volume of Box2 : 1560
Box 3: (18, 20, 15)
Volume of Box3 : 5400
Box1 is not greater than Box2
Box1 is less than Box2
Box1 is not greater or equal to Box2
Box1 is less or equal to Box2
Box1 is not equal to Box2
Box3 is equal to Box4
```

Previous Page Next Page