Overloading Relational Operator

```
using System;
//Overloading relational operator
class A{
       public int a;
     public A(){}

       public A(int a){
          this.a=a;
       }

      public static bool operator ==(A obj1, A obj2){
            return obj1.a==obj2.a;
      }

      public static bool operator !=(A obj1, A obj2){
            return obj1.a!=obj2.a;
      }
}

public class Program
{
      public static void Main()
      {
          A obj1=new A(10);
            A obj2=new A(12);

            bool res=obj1==obj2;
            Console.WriteLine("Equal: {0}",res);

            res=obj1!=obj2;
            Console.WriteLine("Equal: {0}",res);

      }
}
```

Output:

```
Equal: False
Equal: True
```

Likewise, we can overload < and > operator. <= and >= Operator.

Overloading Assignment Operator

We cannot overload assignment operator in C#.

Value Type and Reference Type

Value Type

A Value Type stores its contents in memory allocated on the stack. When you created a Value Type, a single space in memory is allocated to store the value and that variable directly holds a value. If you assign it to another variable, the value is copied directly and both variables work independently. Predefined datatypes, structures, enums are also value types, and work in the same way.

Following are the value types:

All predefined types like:

- int
- float
- double
- char
- boolean etc.

struct
enum

Example of value type:

```
using System;
public class Program
{
      public void Test(int a){
            a=20;
      }

      public static void Main()
      {
            Program obj=new Program();
            int a=10;
            obj.Test(a);
            Console.WriteLine("Value of a is: {0}",a);
      }
}
```

Output:

```
Value of a is: 10
```

Since, int is a value type. After changing value of a in Test function, value of a remains unchanged.

We can forcefully send reference of a variable by using ref keyword. So, that we can change value of a.

```
using System;
public class Program
{
      public void Test(ref int a){
           a=20;
      }
      public static void Main()
      {
            Program obj=new Program();
            int a=10;
            obj.Test(ref a);
            Console.WriteLine("Value of a is: {0}",a);
      }
}
```

Output:

```
Value of a is: 20
```

Reference Type

Reference Types are used by a reference which holds a reference (address) to the object but not the object itself.

Because reference types represent the address of the variable rather than the data itself, assigning a reference variable to another doesn't copy the data. Instead it creates a second copy of the reference, which refers to the same location of the heap as the original value. Reference Type variables are stored in a different area of memory called the heap. This means that when a reference type variable is no longer used, it can be marked for garbage collection.

Examples of reference types are **Classes, Objects, Arrays, Indexers, Interfaces etc.**

Example Program - Array

```
using System;
class A{
      public void Test(int[] arr){
          arr[1]=20;
            arr[2]=30;
      }
}
public class Program
{
      public static void Main()
      {
            int[] arr=new int[5];
            arr[0]=10;

            A obj=new A();
            obj.Test(arr);

            foreach(var item in arr)
                  Console.WriteLine(item);
      }
}
```

Output:

```
10
20
30
0
0
```