|  |  |
| --- | --- |
| **Ex. No. 04** | **OPERATOR OVERLOADING** |
| **28.08.2023** |

**Aim**

To develop C# console application using operator overloading concept.

**Description**

**Operator Overloading:**

* Gives the ability to use the same operator to do various operations, provides additional capabilities to C# operators when they are applied to user defined class.
* It is achieved by defining like a function
* Syntax:

<access\_specifier> static <return\_type> operator <operator>(<parameters>){…}

* It should be static and it does not reference to a class, All the basic operations can be overloaded through it.

**Source Code**

**1.**

using System;

namespace E4{

internal class Matrix{

public int[ , ] mat=new int[2,2];

public Matrix(int a, int b, int c, int d) {

this.mat[0, 0] = a;

this.mat[0, 1] = b;

this.mat[1, 0] = c;

this.mat[1, 1] = d;

}

public void Matrix\_display(){

Console.WriteLine("Displaying Matrix");

Console.WriteLine(this.mat[0, 0] + " "+this.mat[0, 1]);

Console.WriteLine(this.mat[1, 0] + " " + this.mat[1, 1]);

}

public static Matrix operator +(Matrix A, Matrix B){

Matrix C=new Matrix(A.mat[0, 0] + B.mat[0,0], A.mat[0, 1] + B.mat[0, 1], A.mat[1, 0] + B.mat[1, 0], A.mat[1,1] + B.mat[1,1]);

return C;

}

public static Matrix operator \*(Matrix A, Matrix B){

Matrix C = new Matrix(A.mat[0, 0] \* B.mat[0, 0] + A.mat[0, 1] \* B.mat[1,0], A.mat[0, 0] \* B.mat[0, 1] + A.mat[0, 1] \* B.mat[1,1], A.mat[1, 0] \* B.mat[0, 0] + A.mat[1, 1] \* B.mat[1, 0], A.mat[1, 0] \* B.mat[0, 1] + A.mat[1, 1] \* B.mat[1,1]);

return C;

}

static void Main(string[] args){

int n1, n2, n3, n4;

Console.WriteLine("Enter Matrix M1 Elements: ");

Console.Write("[0,0] = ");

n1 = Convert.ToInt32(Console.ReadLine());

Console.Write("[0,1] = ");

n2 = Convert.ToInt32(Console.ReadLine());

Console.Write("[1,0] = ");

n3 = Convert.ToInt32(Console.ReadLine());

Console.Write("[1,1] = ");

n4 = Convert.ToInt32(Console.ReadLine());

Matrix M1 = new Matrix(n1, n2, n3, n4);

int o1, o2, o3, o4;

Console.WriteLine("\nEnter Matrix M2 Elements: ");

Console.Write("[0,0] = ");

o1 = Convert.ToInt32(Console.ReadLine());

Console.Write("[0,1] = ");

o2 = Convert.ToInt32(Console.ReadLine());

Console.Write("[1,0] = ");

o3 = Convert.ToInt32(Console.ReadLine());

Console.Write("[1,1] = ");

o4 = Convert.ToInt32(Console.ReadLine());

Matrix M2 = new Matrix(o1,o2,o3,o4);

Console.Write("M1 ");

M1.Matrix\_display();

Console.Write("M2 ");

M2.Matrix\_display();

Matrix M3 = M1 + M2;

Console.Write("M3 Matrix(M1+M2) ");

M3.Matrix\_display();

Matrix M4 = M1 \* M2;

Console.Write("M4 Matrix(M1\*2) ");

M4.Matrix\_display();

Console.ReadKey();

}

}

}

**2.**

using System;

namespace E4{

internal class Rectangle{

int l, b;

public Rectangle(int length, int breadth){

this.l = length;

this.b = breadth;

}

public void display(){Console.WriteLine("Length= " + this.l + " Breadth= " + this.b);}

public static Rectangle operator +(Rectangle first, Rectangle second){

return new Rectangle(first.l + second.l,second.b+ second.b);

}

public static Boolean operator ==(Rectangle first, Rectangle second){

if (first.l == second.l && first.b == second.b) return true;

else return false;

}

public static Boolean operator !=(Rectangle first, Rectangle second){

if (first.l != second.l && first.b != second.b) return true;

else return false;

}

public static Boolean operator >(Rectangle first, Rectangle second){

if (first.l > second.l && first.b > second.b) return true;

else return false;

}

public static Boolean operator <(Rectangle first, Rectangle second){

if (first.l < second.l && first.b < second.b) return true;

else return false;

}

public static Boolean operator >=(Rectangle first, Rectangle second){

if (first.l >= second.l && first.b >= second.b) return true;

else return false;

}

public static Boolean operator <=(Rectangle first, Rectangle second){

if (first.l <= second.l && first.b <= second.b) return true;

else return false;

}

public static void Main(string[] args){

Console.WriteLine("Enter the Dimensions for R1");

Console.Write("Length: ");

int r1\_l=Convert.ToInt32(Console.ReadLine());

Console.Write("Breadth: ");

int r1\_b = Convert.ToInt32(Console.ReadLine());

Console.WriteLine();

Rectangle R1 = new Rectangle(r1\_l, r1\_b);

Console.WriteLine("Enter the Dimensions for R2");

Console.Write("Length: ");

int r2\_l = Convert.ToInt32(Console.ReadLine());

Console.Write("Breadth: ");

int r2\_b = Convert.ToInt32(Console.ReadLine());

Rectangle R2 = new Rectangle(r2\_l, r2\_b);

Console.WriteLine();

Console.WriteLine("R1 Dimensions");

R1.display();

Console.WriteLine("\nR2 Dimensions");

R2.display();

Rectangle R3 = R1 + R2;

Console.WriteLine("\nR3 (R1+R2) Dimensions");

R3.display();

Console.WriteLine("\nDisplaying Results");

Console.Write("R1==R2: ");Console.WriteLine(R1==R2);

Console.Write("R1<R2: "); Console.WriteLine(R1 < R2);

Console.Write("R1>=R2: "); Console.WriteLine(R1 >= R2);

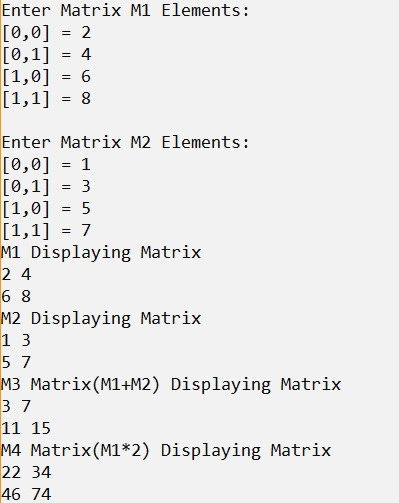
}

}

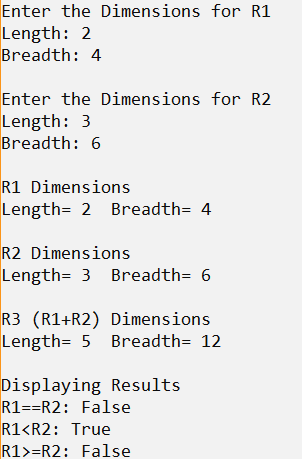
}

**Output**

**1.**



**2.**

1

**Result**

The C# console application using operator overloading has been executed successfully and the desired output is displayed on the screen.