Ex. No. 05	DELEGATES & LAMBDA EXPRESSION
04.09.2023	

Aim

To develop C# console application using Delegates and Lambda Expression concept.

Description

Delegates:

Pointer to a method, address of a method is stored it also has the capability to store multiple addresses where all can be run one after the other so the return statement should be void is known as Multicast Delegate.

Syntax: <access_specifier> delegate <return_type> <delegate_name> (<parameters>)

Creating an object to delegate: <delegate_name> <delegate_object> = new <delegate_name>(<function_name>); **calling is done by** <delegate_object>(<parameter>), in the case of multicast delegate add multiple functions using += operator; <delegate_object>+=<class_object>.<function2_name>.

Lambda Expressions:

Short block of code that accepts parameter and returns values, also known as anonymous function.

Syntax: (parameter_list) => <lambda_body>

2 types:

- **Expression lambda:** Lambda body has only one expression.
- **Statement lambda:** Lambda body has more than one statements, enclosed within curly braces.

Source Code

```
using System;
using System.Threading.Tasks;
namespace Ex5{
    internal class Matrix{
        int[,] matrix=new int[2,2];
        public Matrix(int a, int b, int c, int d){
            this.matrix[0, 0] = a;
            this.matrix[0, 1] = b;
            this.matrix[1, 0] = c;
            this.matrix[1, 1] = d;}
        public void Matrix_display(){
            Console.WriteLine("Displaying Matrix");
            Console.WriteLine(this.matrix[0, 0] + " " + this.matrix[0, 1]);
            Console.WriteLine(this.matrix[1, 0] + " " + this.matrix[1, 1]); }
        public void addition(Matrix A, Matrix B){
            Matrix C = new Matrix(A.matrix[0, 0] + B.matrix[0, 0], A.matrix[0, 1] + B.matrix[0,
1], A.matrix[1, 0] + B.matrix[1, 0], A.matrix[1, 1] + B.matrix[1, 1]);
            Console.WriteLine("Addition");
            C.Matrix_display(); }
        public void subtracion(Matrix A, Matrix B){
            Matrix C = new Matrix(A.matrix[0, 0] - B.matrix[0, 0], A.matrix[0, 1] - B.matrix[0, 1],
A.matrix[1, 0] - B.matrix[1, 0], A.matrix[1, 1] - B.matrix[1, 1]);
            Console.WriteLine("Subtraction");
```

```
        C.Matrix_display(); }

    public void multiplication(Matrix A, Matrix B) {

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

        Console.WriteLine("Multiplication");

        C.Matrix_display(); }

    public void division(Matrix A, Matrix B) {

        Matrix C = new Matrix(A.matrix[0, 0] / B.matrix[0, 0], A.matrix[0, 1] / B.matrix[0, 1], A.matrix[1, 0] / B.matrix[1, 0], A.matrix[1, 1] / B.matrix[1, 1]);

        Console.WriteLine("Division");

        C.Matrix_display(); }

    public delegate void Matrix_delegate(Matrix A ,Matrix B);

    static void Main(string[] args){

        var cone_volume_expression = (int radius, int height) => ((1 / 2) * (22 / 7) * (float)Math.Pow(radius, 2) * height);

        Console.WriteLine("Enter Dimensions of the Cone");

        Console.Write("Radius: ");

        int r = Convert.ToInt32(Console.ReadLine());

        Console.Write("Height: ");

        int h = Convert.ToInt32(Console.ReadLine());

        Console.WriteLine("Volume of the Cone = "+cone_volume_expression(r,h));

        var palindrome_expression = (int num) =>{

            int copy = num;
```

```
    int reversed = 0;
    while (copy > 0){
        reversed *= 10;
        int remain = copy % 10;
        reversed += remain;
        copy /= 10; }
    if (num == reversed) { return true; }
    else { return false; } };
Console.Write("\n\nEnter Any Number for Palindrome Checking: ");
int n = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("Palindrome status is " + palindrome_expression(n));
int n1, n2, n3, n4;
Console.WriteLine("\n\nEnter 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_delegate delegate_obj=new Matrix_delegate(M1.addition);
            delegate_obj+=M1.subtracion;
            delegate_obj+=M1.multiplication;
            delegate_obj+=M1.division;
            delegate_obj(M1,M2);
            Console.ReadKey();
        }
    }
}
```

Output

```
Enter Dimensions of the Cone
Radius: 5
Height: 10
Volume of the Cone = 0


Enter Any Number for Palindrome Checking: 1223221
Palindrome status is True


Enter Matrix M1 Elements:
[0,0] = 1
[0,1] = 2
[1,0] = 3
[1,1] = 4

Enter Matrix M2 Elements:
[0,0] = 8
[0,1] = 7
[1,0] = 6
[1,1] = 5
M1 Displaying Matrix
1 2
3 4
M2 Displaying Matrix
8 7
6 5
Addition
Displaying Matrix
9 9
9 9
Subtraction
Displaying Matrix
-7 -5
-3 -1
Multiplication
Displaying Matrix
20 17
48 41
Division
Displaying Matrix
0 0
0 0
```

Result

The C# console application using Delegates and Lambda Expression has been executed successfully and the desired output is displayed on the screen.