**----------- lambda expression--------------------------**

static void Main(string[] args)

{

Func<int, int> func1 = x => x + 1;

Console.WriteLine("Input a Number : ");

int i = Int32.Parse(Console.ReadLine());

// Use lambda expression with statement body.

//

Func<int, int> func2 = x => { return x + 1; };

//

// Use formal parameters with expression body.

//

Func<int, int> func3 = (int x) => x + 1;

//

// Use parameters with a statement body.

//

Func<int, int> func4 = (int x) => { return x + 1; };

//

// Use multiple parameters.

//

Func<int, int, int> func5 = (x, y) => x - y;

//

// Use no parameters in a lambda expression.

//

Action func6 = () => Console.WriteLine();

//

// Use delegate method expression.

//

Func<int, int> func7 = delegate (int x) { return x + 1; };

//

// Use delegate expression with no parameter list.

//

Func<int> func8 = delegate { return 1 + 1; };

//

// Invoke each of the lambda expressions and delegates we created.

// ... The methods above are executed.

//

Console.WriteLine(func1.Invoke(i));

Console.WriteLine(func1.Invoke(1));

Console.WriteLine(func2.Invoke(1));

Console.WriteLine(func3.Invoke(1));

Console.WriteLine(func4.Invoke(1));

Console.WriteLine(func5.Invoke(2, 2));

func6.Invoke();

Console.WriteLine(func7.Invoke(1));

Console.WriteLine(func8.Invoke());

Console.ReadKey();

}

**--------------Ienumerable-----------------------**

public static System.Collections.IEnumerable save() { yield return 3; yield return 6; yield return 5; yield return 7; yield return 9; yield return 11; } static void Main(string[] args)

{

foreach (int number in save()) { Console.Write(number); } Console.ReadKey();

}

**--------------Delegate-----------------------**

public delegate void MyDelegate(string message); public static void MyMethod(string msg) { Console.WriteLine(msg); } static void MyMethodWithCallback(int param1, int param2, MyDelegate Callback) { Callback("number is :" + (param1 + param2).ToString()); }

MyDelegate handle = MyMethod; handle("Hello"); MyMethodWithCallback(2, 4, handle);