**C# Lab Programs**

1.Develop a C# program to simulate simple airthmetic calculator for addition, subtraction, multiplication, division and mod opertaions. Read the operator and operands through console.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace C\_\_Program\_1

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine("enter the number1");

float number1 = Convert.ToSingle(Console.ReadLine());

Console.WriteLine("enter the number2");

float number2 = Convert.ToSingle(Console.ReadLine());

Console.WriteLine("enter the operator");

char operation = char.Parse(Console.ReadLine());

double result = 0;

switch (operation)

{

case '+':

result = number1 + number2;

break;

case '-':

result = number1 - number2;

break;

case '\*':

result = number1 \* number2;

break;

case '/':

if (number2 != 0)

{

result = number1 / number2;

}

else

{

Console.WriteLine("division by zero is not allowed");

Console.ReadLine();

}

break;

case '%':

if (number2 != 0)

{

result = number1 % number2;

}

else

{

Console.WriteLine("Modulus by zero is not allowed");

Console.ReadLine();

}

break;

default:

Console.WriteLine("invalid operator");

Console.ReadLine();

return;

}

Console.WriteLine("Result:" + number1 + " " + operation + " " + number2 + " = " + result);

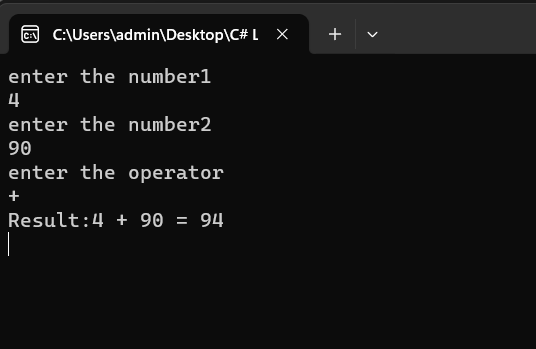
Console.ReadLine();

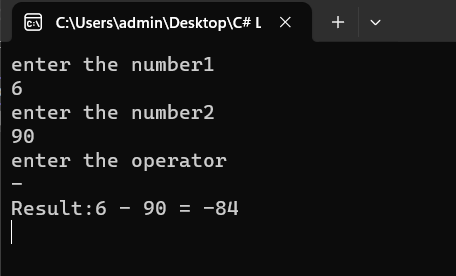
}

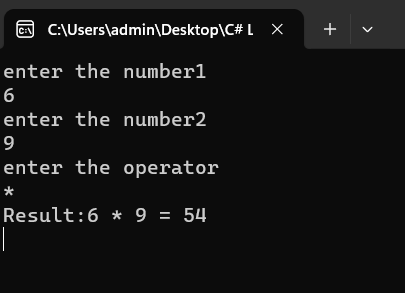
}

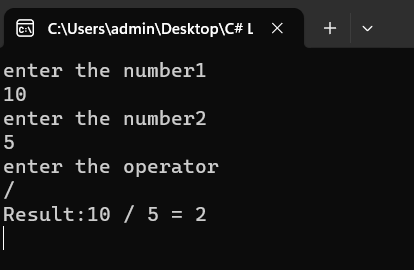
}

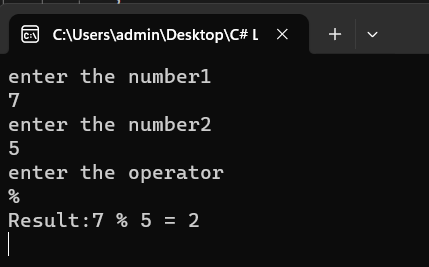
**OUTPUT**

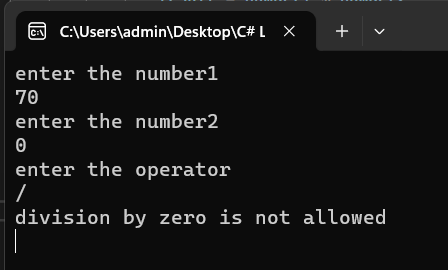


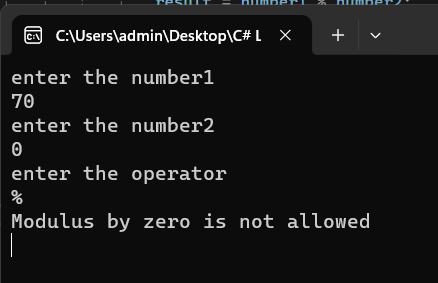












2. Develop a C# program to print Armstrong number between 1 to 1000.

using ArmstrongNumber;

using System;

namespace ArmstrongNumber

{

class Program

{

public void Check\_ArmstrongNumber(int number)

{

int originalNumber = number;

int n = CountDigit(number);

int sum = 0;

while (number > 0)

{

int digit = number % 10;

sum = sum + (int)Math.Pow(digit, n);

number = number / 10;

}

if (sum == originalNumber)

{

Console.WriteLine(originalNumber);

}

}

public int CountDigit(int number)

{

int count = 0;

while (number > 0)

{

count++;

number = number / 10;

}

return count;

}

}

class MainClass

{

static void Main(String[] args)

{

Program obj = new Program();

Console.WriteLine("Armstrong number between 1 to 1000 are");

for (int i = 1; i <= 1000; i++)

{

obj.Check\_ArmstrongNumber(i);

}

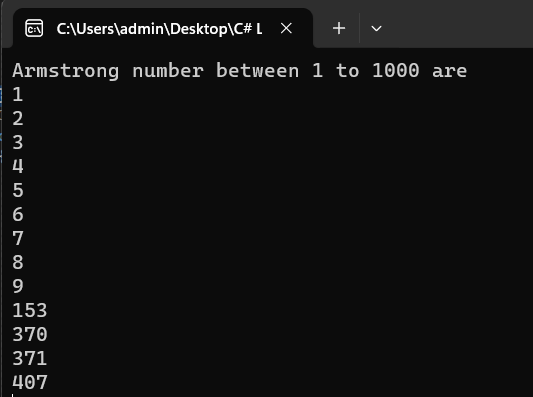
Console.ReadLine();

}

}

}

OUTPUT



3. Develop a c# program to list all substring in a given string[Hint:use of Substring() method].

using System;

namespace Substrings

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine("enter the string");

string str=Console.ReadLine();

Console.WriteLine("the possible substrings in the given string are");

Getsubstring(str);

}

static void Getsubstring(string str)

{

for(int i=0; i<str.Length; i++)

{

for(int j=i+1; j<=str.Length; j++)

{

string substring=str.Substring(i,j-i);

Console.WriteLine(substring);

}

}

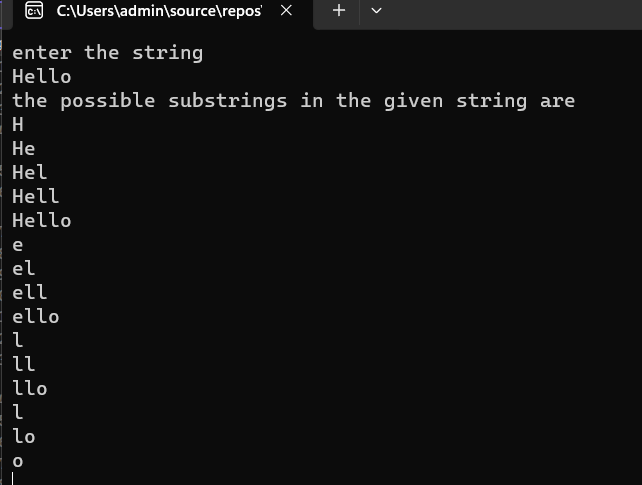
Console.ReadLine();

}

}

}

OUTPUT



4. Develop a C# program to demonstrate Diision by Zero and Index Out of Range exception.

using System;

namespace ExceptionHandling

{

class Program

{

public static void Main(string[] args)

{

Console.WriteLine("Enter numerator:");

float numerator = int.Parse(Console.ReadLine());

Console.WriteLine("Enter denominator:");

float denominator = int.Parse(Console.ReadLine());

try

{

if (denominator == 0)

{

throw new DivideByZeroException("Cannot divide by zero.");

}

float result = numerator / denominator;

Console.WriteLine("Result of division:"+result);

}

catch (DivideByZeroException ex)

{

Console.WriteLine("Divide by zero exception:"+ex.Message);

}

Console.WriteLine("Enter the array size:");

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

Console.WriteLine("Enter the array elements:");

try

{

int[] array = new int[arraySize]; //creating an array

for (int i = 0; i < arraySize; i++)

{

array[i] = Convert.ToInt32(Console.ReadLine());

}

Console.WriteLine("enter the array position to access value");

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

Console.WriteLine("value in the givin position is:"+array[index]);

}

catch (IndexOutOfRangeException ex)

{

Console.WriteLine("Index out of range exception:" +ex.Message);

}

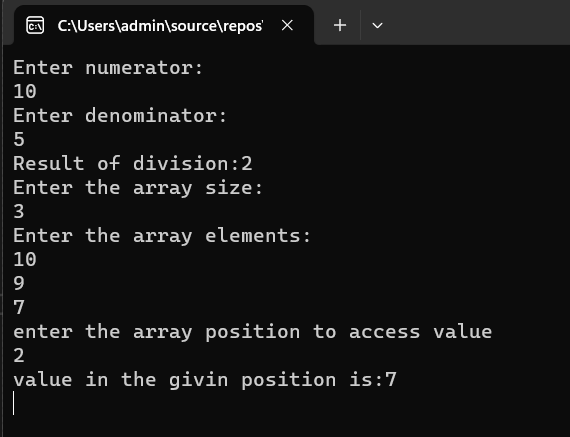
Console.ReadLine();

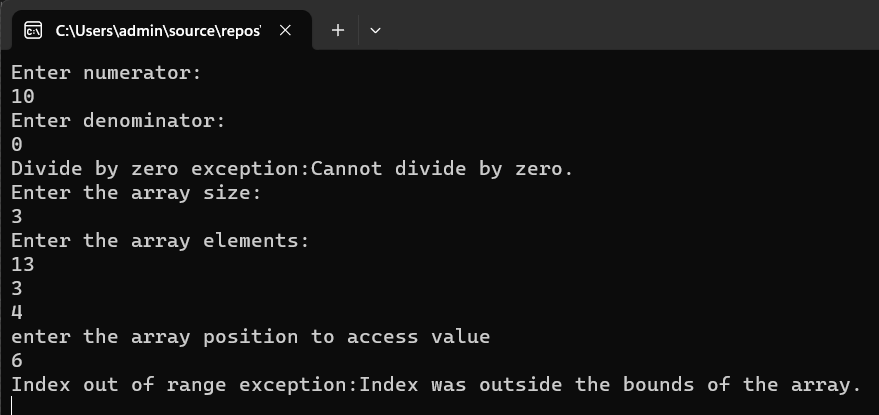
}

}

}

OUTPUT





**5. Develop a C# Program to Generate and Print Pascal Triangle using Two Dimensional Arrays.**

using System;

class Program

{

public static void Main(String[] args)

{

Console.WriteLine("Enter the number of rows");

int Rows = int.Parse(Console.ReadLine());

int[,] pascalTriangle = Generate(Rows);

Print(pascalTriangle);

Console.ReadLine();

}

static int[,] Generate(int Rows)

{

int[,] triangle = new int[Rows, Rows];

for (int i = 0; i < Rows; i++)

{

triangle[i, 0] = 1;

for (int j = 1; j < i; j++)

{

triangle[i, j] = triangle[i - 1, j - 1] + triangle[i - 1, j];

}

triangle[i, i] = 1;

}

return triangle;

}

static void Print(int[,] triangle)

{

Console.WriteLine("Pascal's Triangle:");

for (int i = 0; i < triangle.GetLength(0); i++)

{

// Add leading spaces for formatting

for (int space = 0; space < triangle.GetLength(0) - i - 1; space++)

{

Console.Write(" ");

}

for (int j = 0; j <= i; j++)

{

Console.Write(triangle[i, j] + " ");

}

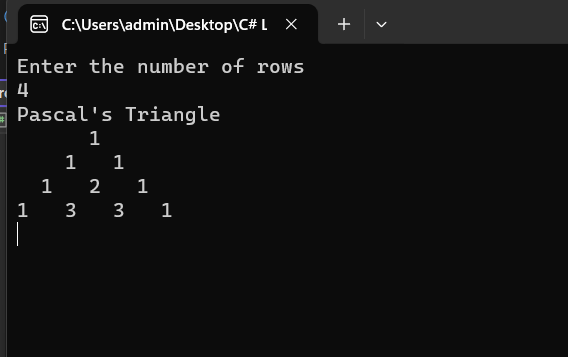
Console.WriteLine();

}

}

}

**OUTPUT**

****

**6. Develop a C# Program to Generate and Print Floyds Triangle using Jagged arrays.**

using System;

class Program

{

public static void Main(String[] args)

{

Console.WriteLine("Enter the number of rows");

int Rows = int.Parse(Console.ReadLine());

Console.WriteLine("Floyd’s Triangle");

Print(Rows);

Console.ReadLine();

}

static void Print(int Rows)

{

int[][] triangle = new int[Rows][];

int value = 1;

for (int i = 0; i < Rows; i++)

{

triangle[i] = new int[i + 1];

for (int j = 0; j <= i; j++)

{

Console.Write(triangle[i][j]+ value++ +" ");

}

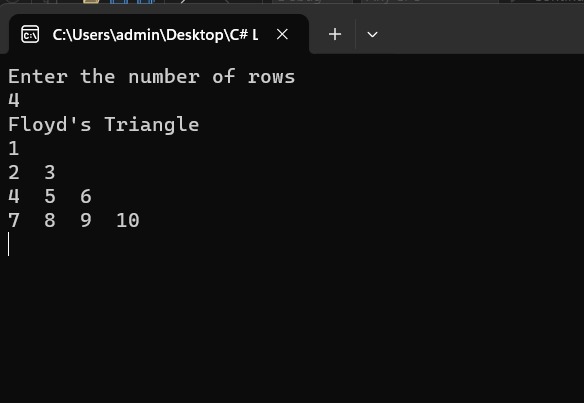
Console.WriteLine();

}

}

}

**OUTPUT**

****

**7. Develop a C# Program to read a text file and Copy the file content s to another text file.**

using System;

using System.IO;

namespace FileHandling

{

class Program

{

static void Main(string[] args)

{

try

{

string sourceFile = @"D:\test.text"; //source file creation

string destinationFile = @"D:\dest.text"; //destination file creation

if (File.Exists(sourceFile))

{

string[] array = new string[] //writing multiple lines to the source file

{

"hi hello",

"today is friday",

"im from mysore"

};

File.WriteAllLines(sourceFile, array);

string[] lines = File.ReadAllLines(sourceFile); //reading the content of the source file

foreach (string line in lines)

{

Console.WriteLine(line); //display the multiple line content of source file

}

File.WriteAllLines(destinationFile, array); //copying the source file content to destination file

Console.WriteLine("-----------------------------------------------------------");

Console.WriteLine("content copied successfully from source to destination file");

}

}

catch (Exception e)

{

Console.WriteLine(e);

}

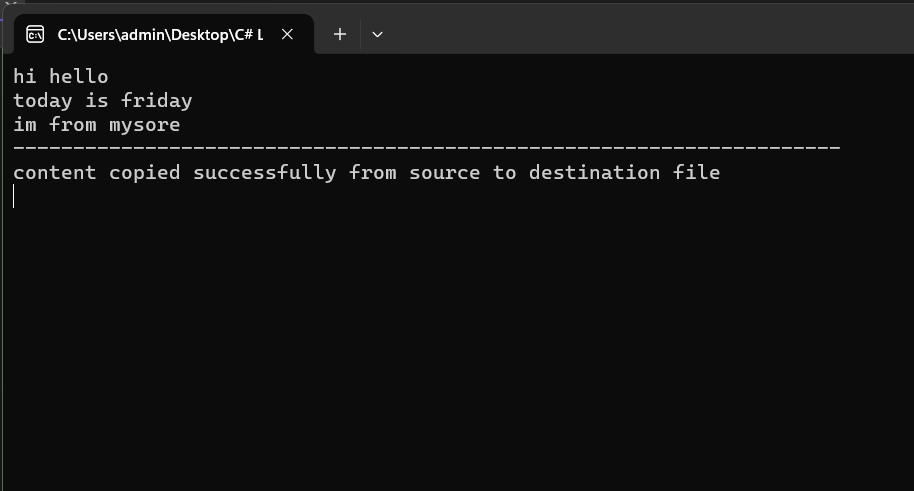
Console.ReadLine();

}

}

}

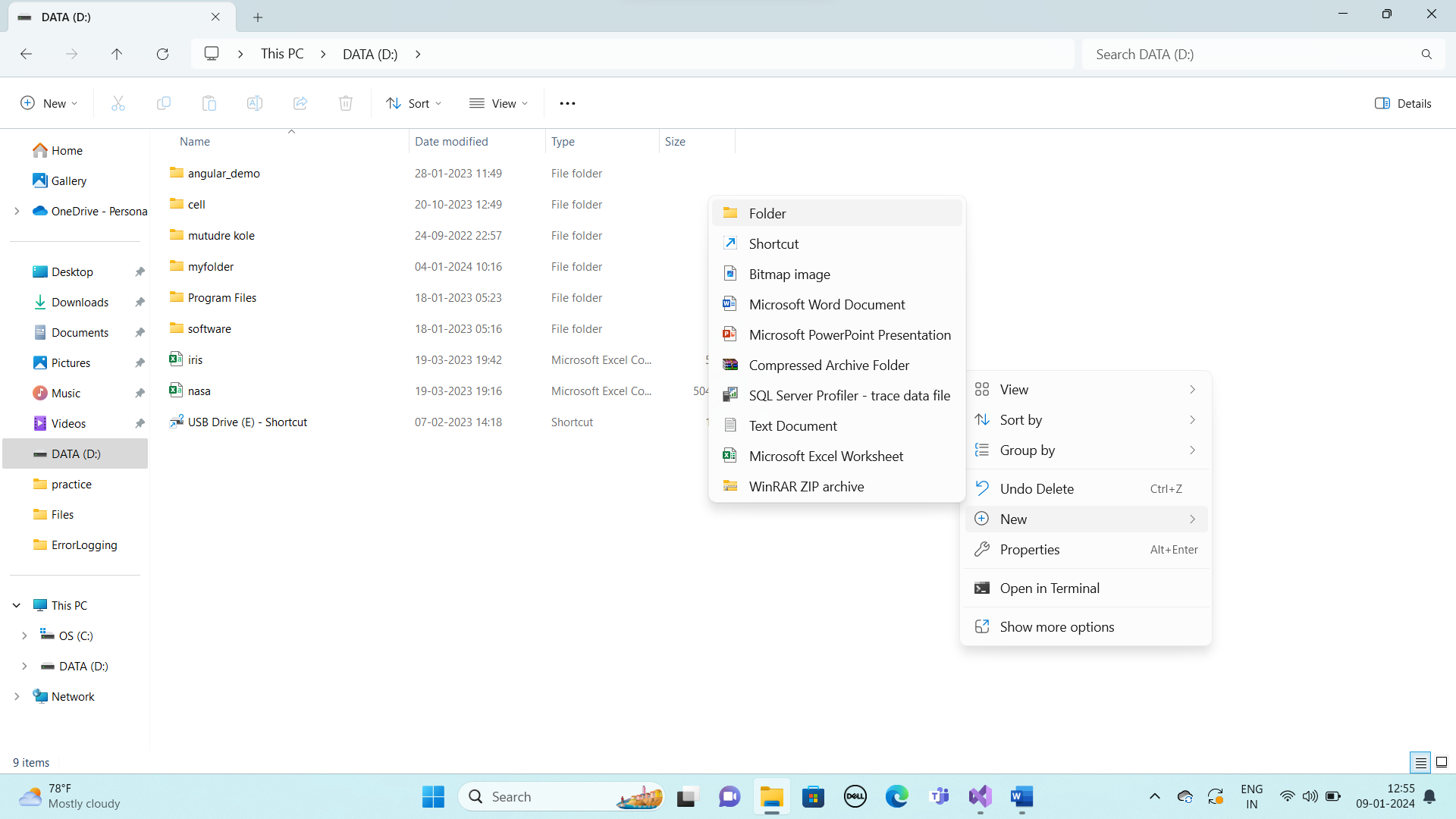
**OUTPUT**

****

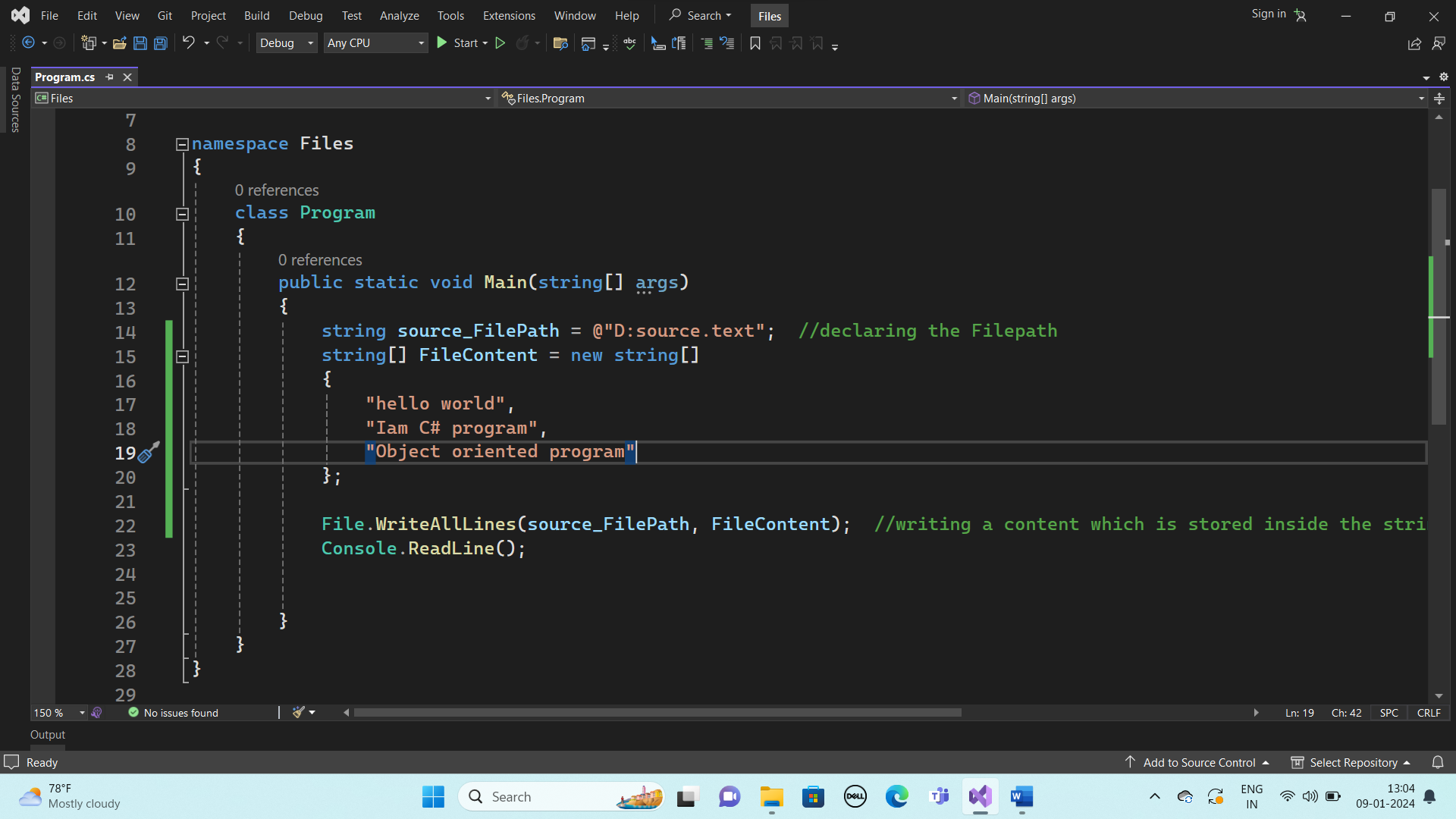
**7th program (File Handling) with steps**

**Step1: create a folder in D drive**

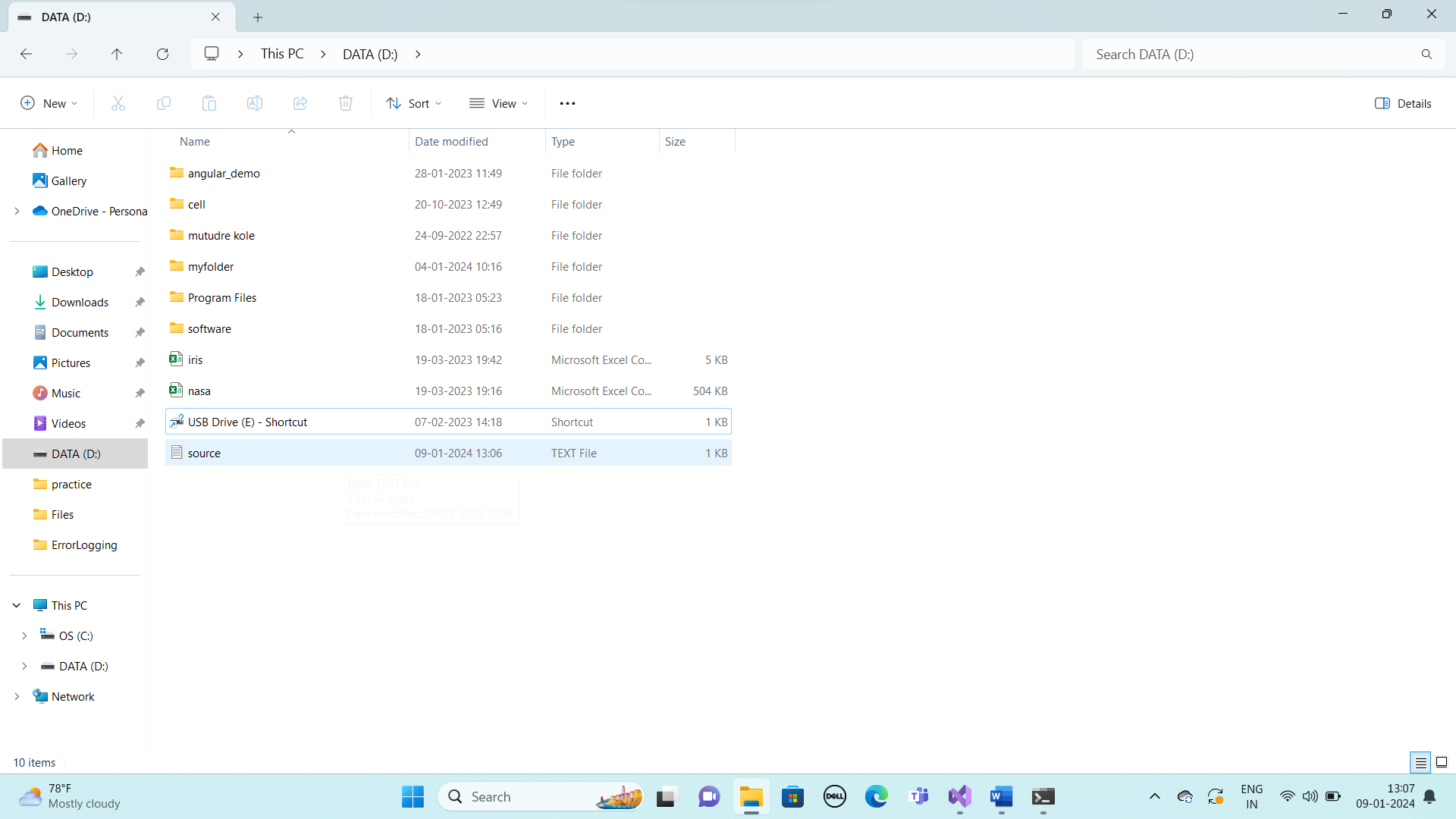
**Right click->new->Folder->Folder Name**

****

**Step2:in Visiual studio create file inside the created folder and write the content and execute**

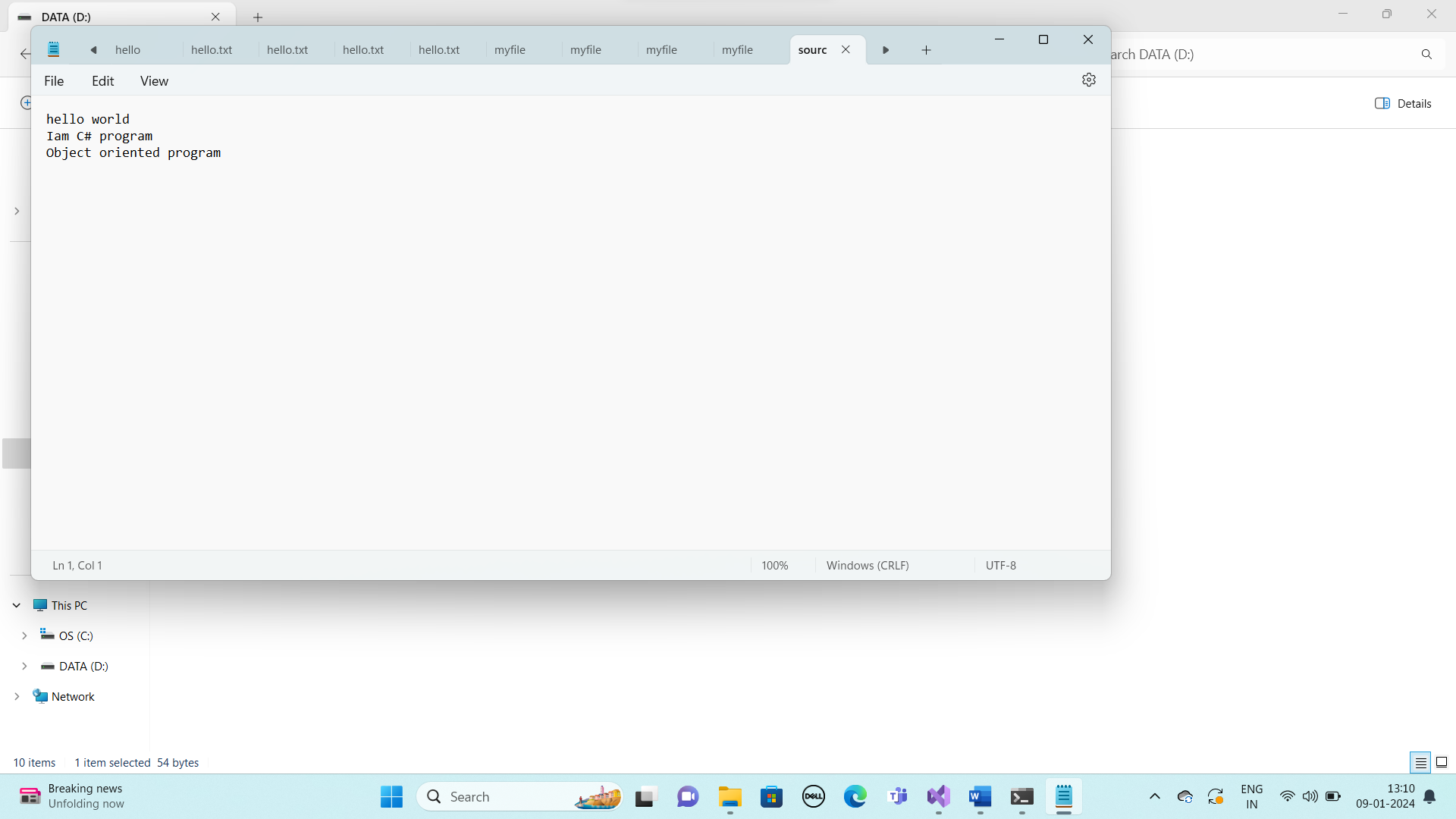
****

**Check whether given file is created and content is stored inside the file in Ddrive->folder->file**

****



**File is created**

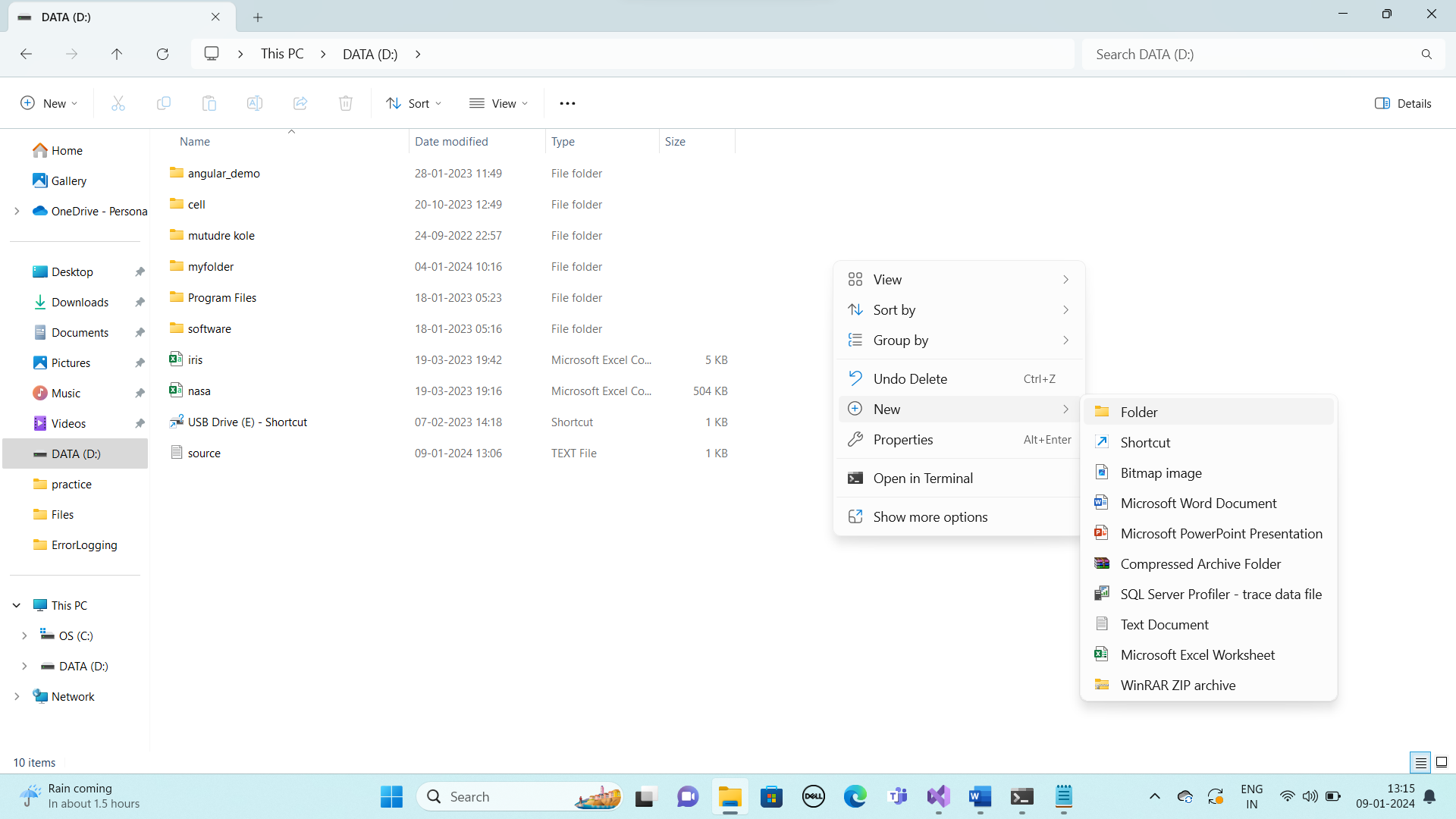
**

**Content stored successfully**

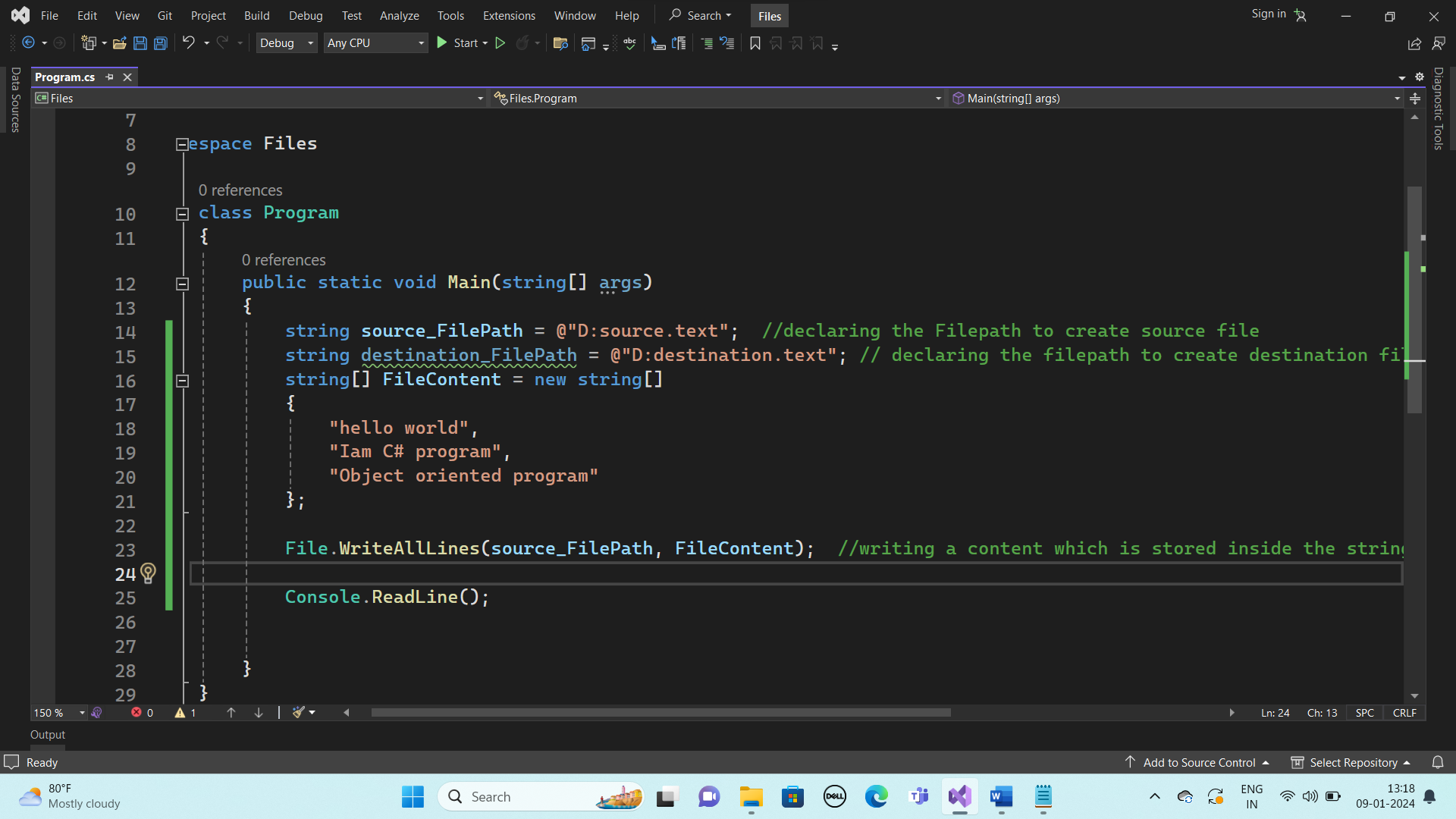
**Step3: display the content of the file using foreach Loop**

**(given in below program)**

**Step4: create another separate folder in D drive**

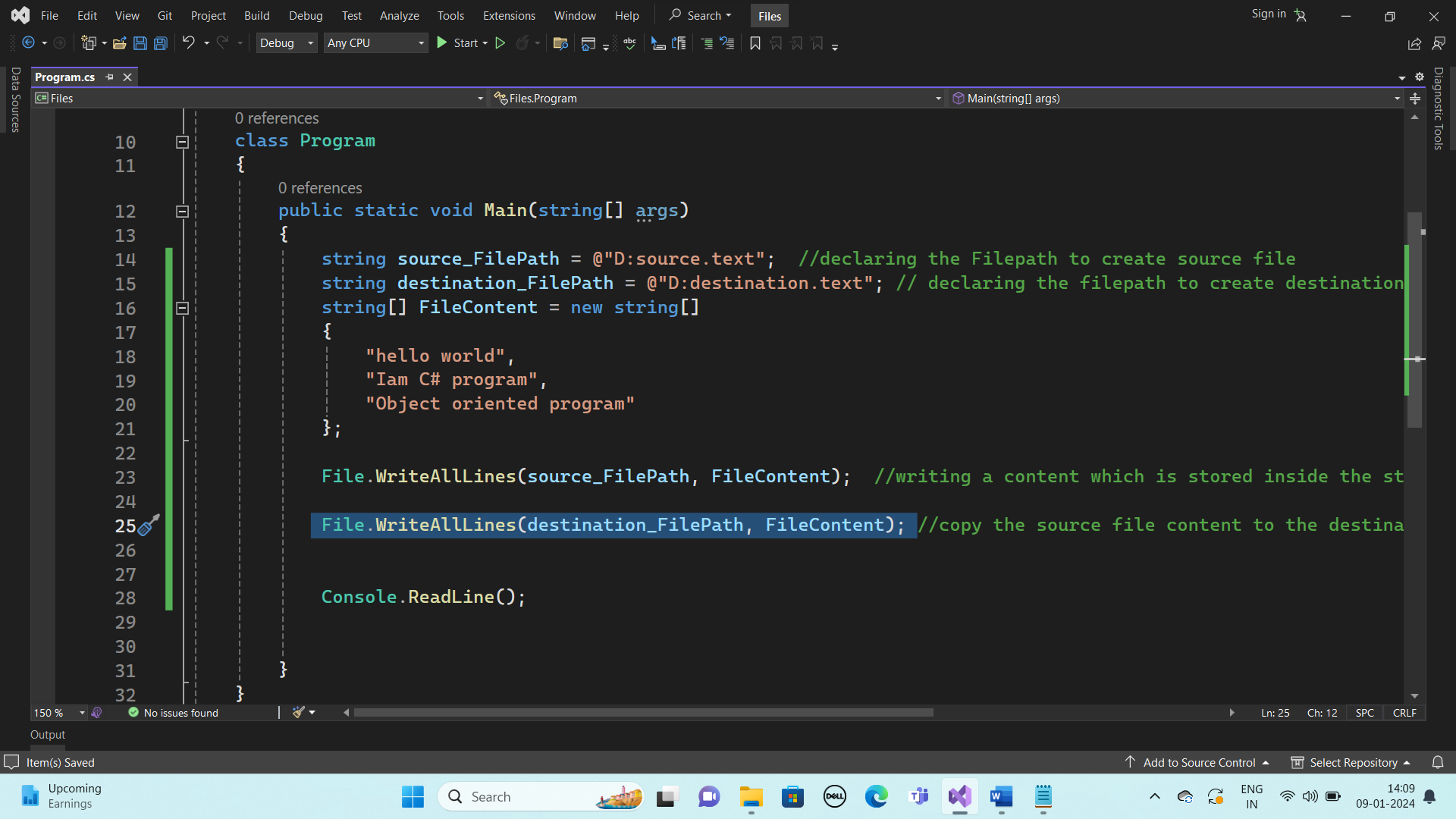
****

**Step 5: in visual studio create destination file path and execute**

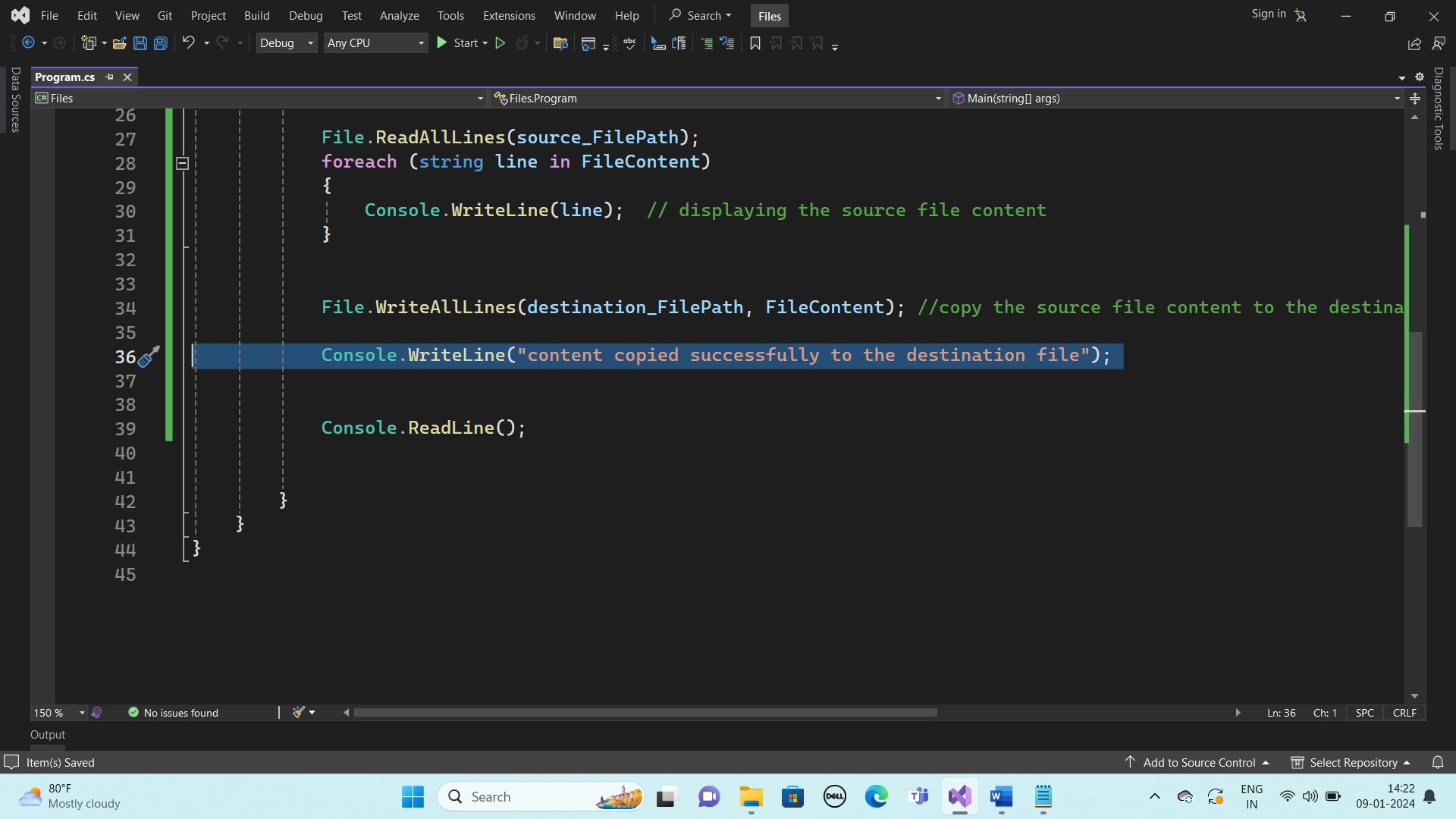
****

**Step6: then copy the source file content to destination file**

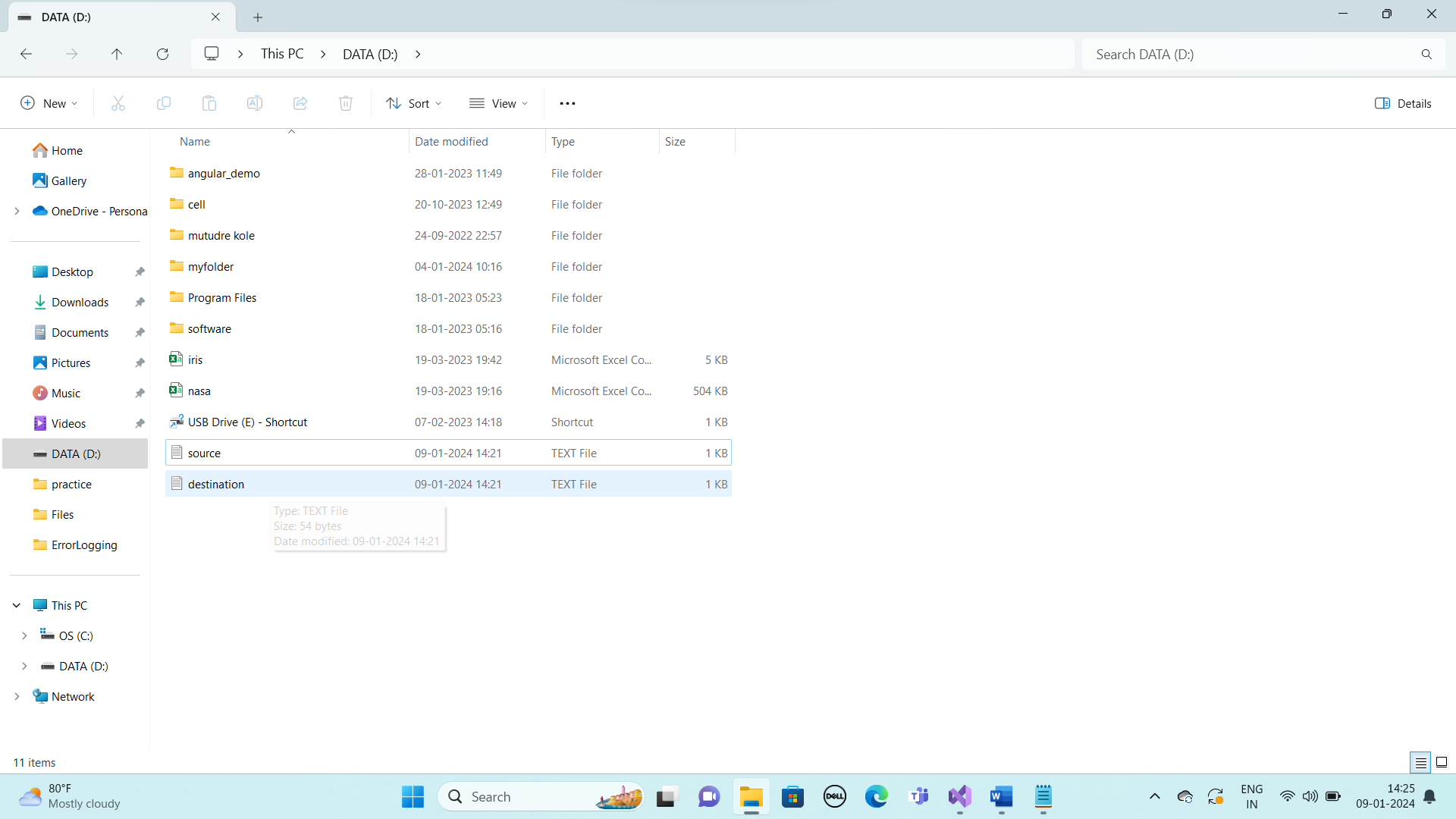
**(code given below)**

****

**Step7: after copy print output statement “content copied successfully” and execute**

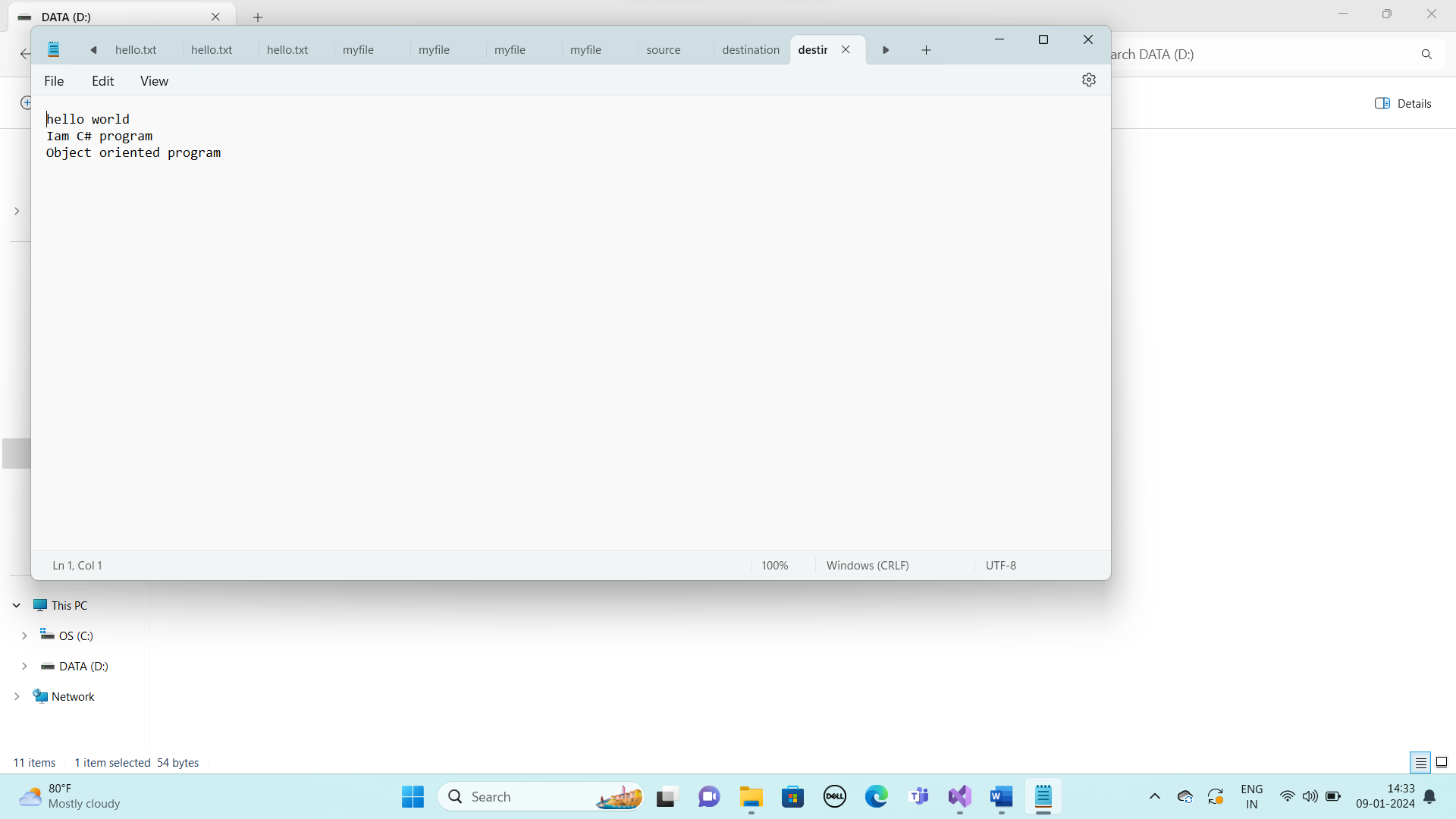
****

**Step 8: check whether the content is copied to the destination file in Ddrive->destinationFolder->destinationFile**

****



**File created**

****

**Content copied successfully from source to destination file**

**7th program code**

using System;

using System.IO;

namespace Files

{

class Program

{

public static void Main(string[] args)

{

string source\_FilePath = @"D:source.text"; //declaring the Filepath to create source file

string destination\_FilePath = @"D:destination.text"; // declaring the filepath to create destination file

string[] FileContent = new string[]

{

"hello world",

"Iam C# program",

"Object oriented program"

};

File.WriteAllLines(source\_FilePath, FileContent); //writing a content which is stored inside the string array to the source file

Console.WriteLine("souce file content:");

File.ReadAllLines(source\_FilePath);

foreach (string line in FileContent)

{

Console.WriteLine(line); // displaying the source file content

}

Console.WriteLine("--------------------------------------------");

File.WriteAllLines(destination\_FilePath, FileContent); //copy the source file content to the destinatination file

Console.WriteLine("content copied successfully to the destination file");

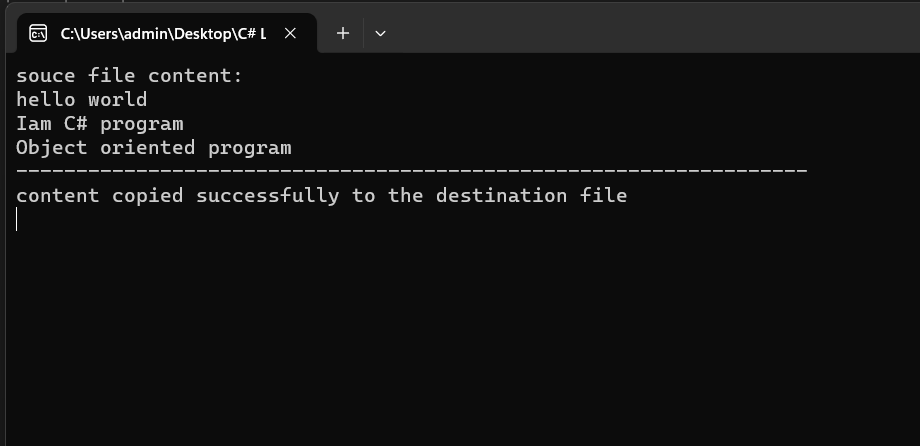
Console.ReadLine();

}

}

}

**Output**

****

**8.Develop a C# program to implement stackwith push pop operations (use class, methods for push and pop and main method).**

using System;

namespace StackDemo2

{

class Stack

{

public int top = -1;

int[] mystack = new int[5];

public void push(int value)

{

if (top < 4)

{

top++;

mystack[top] = value;

}

else

{

Console.WriteLine("overflow condition cannot insert element");

}

}

public void pop()

{

if (top > -1)

{

int popedItem = mystack[top];

top--;

Console.WriteLine("the item poped " + popedItem);

}

else

{

Console.WriteLine("underflow condition cannot delete element");

}

}

public void display()

{

Console.WriteLine("stack contains below elements");

for(int i=0; i<=top; i++)

{

Console.WriteLine(mystack[i]);

}

}

}

class Program

{

static void Main(string[] args)

{

Stack s = new Stack();

while (true)

{

Console.WriteLine("\n1.push\n2.pop\n3.Display");

Console.WriteLine("enter the choice");

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

switch (choice)

{

case 1:

Console.WriteLine("enter element to insert");

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

s.push(value);

break;

case 2:

s.pop();

break;

case 3:

s.display();

break;

default:

Console.WriteLine("wrong choice bye");

break;

}

}

Console.ReadLine();

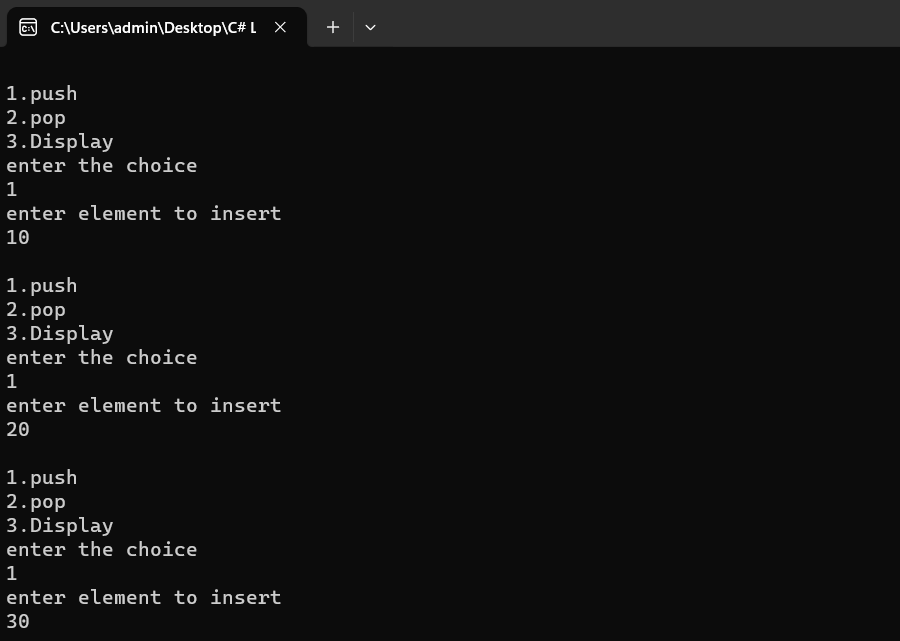
}

}

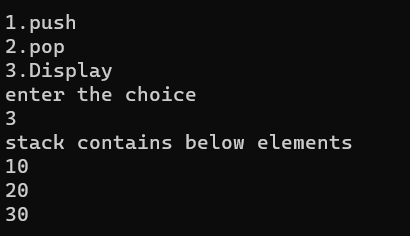
}

**Output**

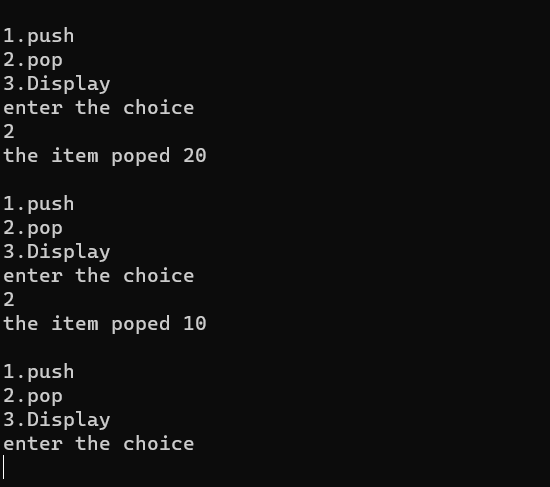
**1.push elements**

****

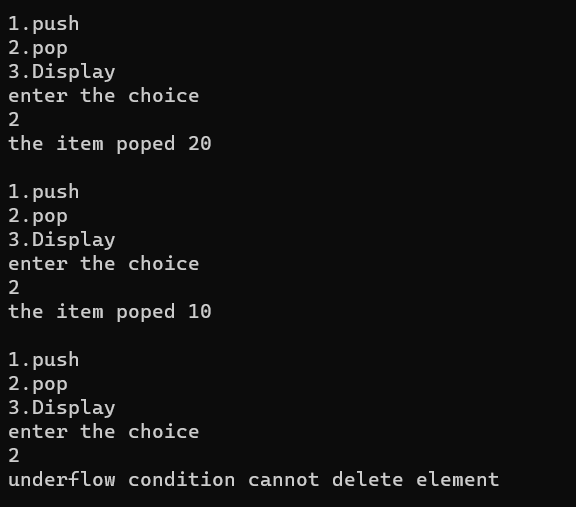
**2.display stack elements**

****

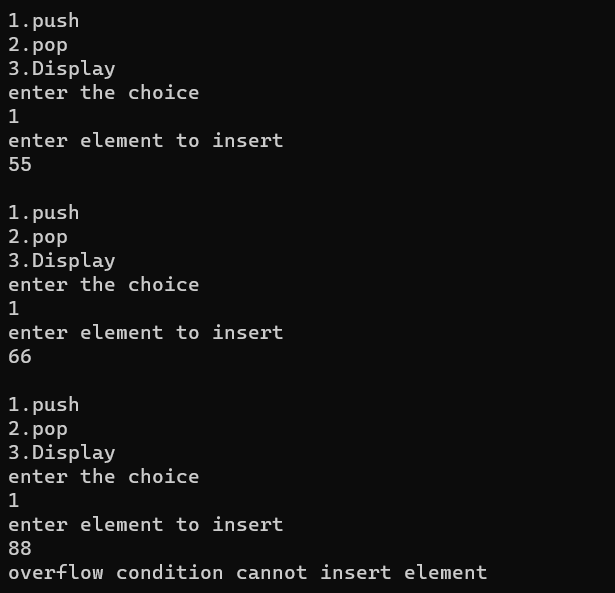
**3.pop elements**

****

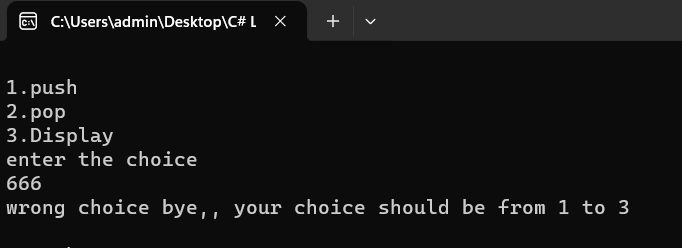
**4.underflow condition**

****

**5.overflow condition**

****

**6.Wrong choice**

****

**9**. **Design a class Complex with Data members, Constructor and method to overloading a binary operator ‘+’ Deelop a C# program to read two complex numbers and print the result of addition.**

using System;

namespace Program\_9

{

class Complex

{

public double real;

public double imaginary;

// Constructor

public Complex(double r, double i)

{

real = r;

imaginary = i;

}

// Method for overloading the + operator

public static Complex operator +(Complex complex1, Complex complex2)

{

double realPart = complex1.real + complex2.real;

double imaginaryPart = complex1.imaginary + complex2.imaginary;

return new Complex(realPart, imaginaryPart);

}

}

class Program

{

public static void Main(string[] args)

{

Console.WriteLine("Enter the real and imaginary parts of the first complex number:");

double real1 = Convert.ToDouble(Console.ReadLine());

double imaginary1 = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Enter the real and imaginary parts of the second complex number:");

double real2 = Convert.ToDouble(Console.ReadLine());

double imaginary2 = Convert.ToDouble(Console.ReadLine());

Complex complex1 = new Complex(real1, imaginary1);

Complex complex2 = new Complex(real2, imaginary2);

Complex sum = complex1 + complex2;

Console.WriteLine("The sum of the two complex numbers is:");

Console.WriteLine(sum.real + " + " + sum.imaginary + "i");

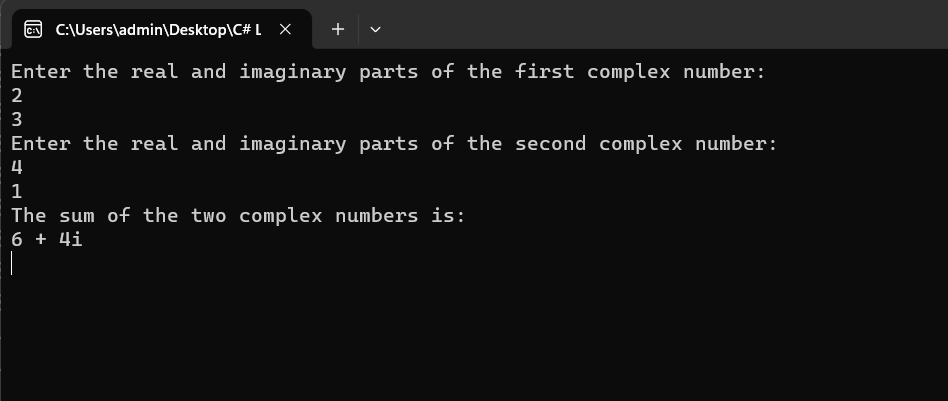
Console.ReadLine();

}

}

}

**OUTPUT**



**10. Develop a C# program to create a class named Shape create three sub classes namely Circle, Triangle, Square Each class has 2 member function named draw() and Erase(). Demonstrate a Polymorphism concept by developing suitable methods, defining member data and main program.**

using System;

namespace \_10\_Program

{

class Shape

{

public virtual void draw()

{

Console.WriteLine("drawing a shape");

}

public virtual void Erase()

{

Console.WriteLine("Erase a Shape");

}

}

class Circle : Shape

{

public override void draw()

{

Console.WriteLine("drawing a circle");

}

public override void Erase()

{

Console.WriteLine("Erase a circle");

}

}

class Triangle : Shape

{

public override void draw()

{

Console.WriteLine("drawing a Traingle");

}

public override void Erase()

{

Console.WriteLine("Erase a Trainagle");

}

}

class Square : Shape

{

public override void draw()

{

Console.WriteLine("drawing a Square");

}

public override void Erase()

{

Console.WriteLine("Erase a Square");

}

}

class MainClass

{

static void Main(string[] args)

{

Shape s= new Shape();

s.draw();

s.Erase();

Circle c = new Circle();

c.draw();

c.Erase();

Triangle t = new Triangle();

t.draw();

t.Erase();

Square sq = new Square();

sq.draw();

sq.Erase();

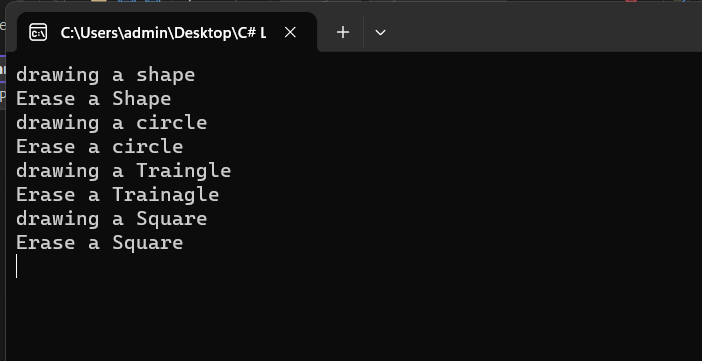
Console.ReadLine();

}

}

}

**OUTPUT**

****

**11. Develop a C# Program to Create an abstract class Shape with abstract method CalculateArea() and CalculatePeremeter() create subclasses Circle and Triangle that extend the Shape Class and implement the respectie methods to Calculate the area and perimeter of each shape.**

using System;

namespace \_11th\_Program

{

abstract class Shape

{

public abstract void CalculateArea();

public abstract void CalculatePerimeter();

}

class Circle : Shape

{

private double radius;

public Circle(double r)

{

radius = r;

}

public override void CalculateArea()

{

double Area = Math.PI \* radius \* radius;

Console.WriteLine("Area: " +Area);

}

public override void CalculatePerimeter()

{

double perimeter = 2 \* Math.PI \* radius;

Console.WriteLine("Perimeter: " +perimeter);

}

}

class Triangle : Shape

{

private float side1;

private float side2;

private float side3;

public Triangle(float s1, float s2, float s3)

{

side1 = s1;

side2 = s2;

side3 = s3;

}

public override void CalculateArea()

{

float s = (side1 + side2 + side3) / 2;

double Area = Math.Sqrt(s \* (s - side1) \* (s-side2) \* (s-side3));

Console.WriteLine("Area: " +Area);

}

public override void CalculatePerimeter()

{

double perimeter = side1 + side2 + side3;

Console.WriteLine("Perimeter: " + perimeter);

}

}

class PrograM

{

static void Main(string[] args)

{

Circle c = new Circle(5);

Console.WriteLine("Circle:");

c.CalculateArea();

c.CalculatePerimeter();

Triangle t = new Triangle(3, 4, 5);

Console.WriteLine("Triangle:");

t.CalculateArea();

t.CalculatePerimeter();

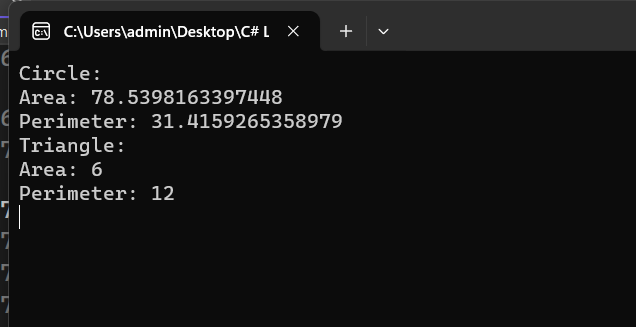
Console.ReadLine();

}

}

}

**OUTPUT**



**12.Develop a C# Program to create an interface Resiazble with methods resizeWidth(int width) and resizeHeigh(int Height) that allow an object to be resized. Create a class Rectabgle that implements the Resizable interface and implement the resize methods.**

using System;

namespace \_12th\_program

{

public interface Iresizeable

{

void Resizewidth(int width);

void Resizeheight(int height);

}

public class Rectangle : Iresizeable

{

public int width;

public int height;

public Rectangle(int Rwidth, int Rheight)

{

width = Rwidth;

height = Rheight;

Console.WriteLine("Current Width value is" + " " + Rwidth + " " + "Current Height value is" + " " + Rheight);

}

public void Resizewidth(int newWidth)

{

if (newWidth > 0)

{

Console.WriteLine("Width is Resized to default width to newWidth " + newWidth);

}

else

{

Console.WriteLine("Width should be Greater than zero");

}

}

public void Resizeheight(int newheight)

{

if (newheight > 0)

{

Console.WriteLine("Height is Resized to default height to newheight " + newheight);

}

else

{

Console.WriteLine("should be Greater than zero");

}

}

}

class Program

{

public static void Main(string[] args)

{

Rectangle r = new Rectangle (100, 50);

Console.WriteLine("Enter the Resize Width");

int newWidth=int.Parse(Console.ReadLine());

Console.WriteLine("Enter the Resize height");

int newheight = int.Parse(Console.ReadLine());

r.Resizewidth(newWidth);

r.Resizeheight(newheight);

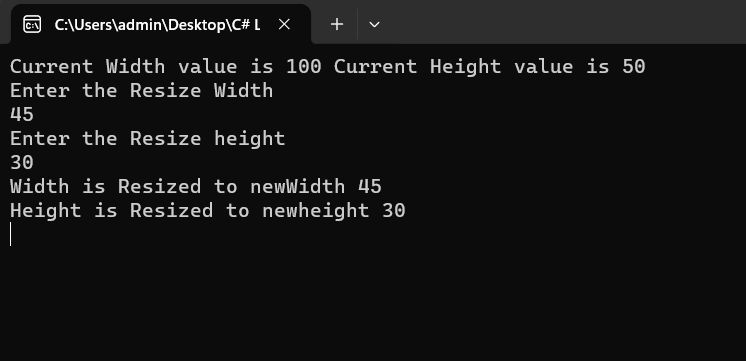
Console.ReadLine();

}

}

}

**OUTPUT**

****