**8. Operators and Expressions**

* Operator perform operations on operands(Variables)

|  |  |  |  |
| --- | --- | --- | --- |
| **Category** | **Operator** | **Example** | **Description** |
| Arithmetic | + | a+b | This is binary operators. The operation done between two operand and result assign to another variable. |
| - | a-b |
| \* | a\*b |
| / | a/b |
| % | a%b |
| Relational | == | a==b | This is a binary operator. Compares two value and return true or false accordingly. |
| != | a!=b |
| < | a<b |
| <= | a<=b |
| > | a>b |
| >+ | a>=b |
| Assignment | = | X=1 | Perform the math operation then assign the value to X.  X=4  X+=1 (X=X+1)  X=4+1=5 |
| += | X+=1 |
| -= | X-=1 |
| \*= | X\*=1 |
| /= | X/=1 |
| %= | X%=1 |
| Logical | && | (a==5 && b==8) | If both condition are true than true otherwise false. |
| || | (a==5 || b==5) | If both condition are false than false otherwise true. |
| ! | !(a==5) | It will inverse the result. |
| Ternary | ? : | (a>b)?a:b | If condition is true? then value a : otherwise value b. |

|  |  |  |  |
| --- | --- | --- | --- |
| Increment/ Decrement | ++ | a++,  ++a | Increase the value by one according to pre-postfix |
| -- | a--,  --a | Decrease the value by one according to pre-postfix. |
| Size Of | sizeof() | sizeof(int) | Return the size of a datatype. |

**Expressions**

Combination of operands and operators are known as Expressions.

1. **Assignment Expression**

It uses the = operator to assign the result of another expression to a variable

Example: A = B+5;

1. **Void Expression**

This type of expression has no value.

Example: Console.WriteLine(1);

**9. Loop Iteration**

* Looping used to execute the statements multiple times.
* Loops are divided into two categories **Entry controlled loops** and **Exit controlled loops**.
* The loop in which condition is tested at the beginning of the loop body is known as Entry controlled loop.
* The loop in which condition is tested at the end of the loop body is known as Exit controlled loop.

1. **For Loop**

* It is Entry controlled loop.
* For loop is used when we know the number of times loop statements are going to execute.
* Syntax:

for (Initialization; Condition; Increment/Decrement)

{

//Statements

}

1. **While Loop**

* It is also entry controlled loop
* In this loop statements are executed till the given condition is true, when it becomes false loop will terminate.
* Syntax:

while ( Condition )

{

//statements

}

1. **do-while loop**

* It is Exit controlled loop.
* This loop will check the condition after executing the statements, in this loop, loop body will be execute at-least one time.
* Syntax:

Do

{

//statements

} while (condition);

1. **Foreach Loop**

* Foreach loop work with the collection object such as array, list etc. to execute the block of statements for each element in the array or collection.
* It is useful to loop through each item in an array or collection object to execute the block of statement repeatedly.
* Syntax:

foreach (Datatype Var-name **in** collection object)

{

//Statements

}

1. **Continue statement**

* Continue statement is used to pass control to the next iteration of loops from the specified position by skipping the remaining code.
* Syntax:

**continue;**

1. **Break statement**

* This statement is used to terminate the execution of loops and in switch statements the control is passed to immediate next statement.
* Syntax:

**break;**

**10. Understanding Arrays**

* An array is a collection of elements of same datatype in continues memory allocation.

**10.1 Define and usage of array**

1. **Single dimension array:** In thiselements stored in a single row in allocated memory.

* Syntax:

Datatype [] arr-name = new datatype [size];

Int [] numbers = new int [5];

1. **Multi-dimensional array:** It allows you to store values of same datatype in two or more dimension.

* Syntax:

Datatype [ , ] arr-name = new datatype [ , ];

string [ , ] names = new string [3,3];

Datatype [][] arr-name = new datatype [] [];

String [][] items = new string [3][];

* In array this arrangement of storing values help in efficient storage of data, easy sorting of data and easy tracking of data.
* The index of array is start from zero.

**11. Defining and Calling Methods**

**11.1 Define method and use**

* A method is a group of statements that perform specific task.
* It provides better readability of code.
* Method may return value or may not.
* We can also reuse the method.
* You can call the method by making the instance of the class.
* Syntax

<Access-modifier><return-type><method-name> (Parameters)

Public int add (int a, int b)

{

//statements

}

**11.2 Different types of parameters in method**

**(a) Value type**

* In value type parameter the actual value gets passed to the function. It passes the copy of the data not actual data.
* If any changes made in the value type parameter it will not affect in the original data.

**(b) Reference type**

* A reference type parameter does not contain data directly, it refers the memory location of a variable.
* When we pass parameter by reference, a new storage location is not created like value type, reference type parameter represents the same memory location which the actual parameter that supplied parameter have.
* We can declare reference parameters using ref keyword
* Ex. Public int square (ref int a)
* We can also use the out keyword. The main difference between ref and out is the ref needs that the variable must be initialized before it passed to the method. But out parameter doesn’t require the variables to be initialized before it passed to the method. But the variable must be initialized in the called method.

**(c) Optional Parameter**

* Optional parameter contains a default value
* If we do not pass any parameter to the optional argument, then it takes its default value.

**12. Working with strings**

**12.1 String class study**

* A string is represented by class System.String.
* This class is immutable, means once created its can’t changed.
* Whenever it seems that any method or operation is changing a string, it is basically creating a new string object.

**12.2 Use of various string methods**

|  |  |
| --- | --- |
| **Name** | **Description** |
| ToUpper() | Return a copy of uppercase converted string. |
| ToLower() | Return a copy of lowercase converted string. |
| Equals(object) | Check whether the two string having same value or not. |
| Compare(string,string) | Compare two specified string object and returns an integer. |
| Concat(str) | Join the strings |
| IndexOf(str) | Returns the index of the first occurrence of a specified text in a string. |
| lastIndexOf(str) | Return the index of last occurrence of a specified text in a string. |
| search(str) | Searches a string for a specified value and return the position of the match. |
| Replace(str, str) | Replaces a specified value with another value in the string. |
| Substring() | It is used to get a part of string from a specified index. |
| charAt(i) | Return a character at specified index in a string. |

* There is also availability of string builder class
* String and StringBuilder both represent sequences of characters but they are implemented differently.
* String is a immutable. In each operation that appears to modify a string object, it actually creates a new string.
* StringBuilder is a mutable string class , it means we can modify the instance of class by appending, removing, replacing or inserting characters on a single string.
* StringBuilder is faster than string in making the changes to string.
* If you want to modify your string more than 4-5 time than use StringBuilder and then convert it to string by using ToString() method.

**13. Working with Datetime**

**12.1 Datetime class study**

* Datetime class is predefine class in .net library which we can use whenever we want.
* A date and time format specifier is a special character that enables you to display the date and time values in different formats

|  |  |
| --- | --- |
| **Format Specifier** | **Description** |
| d | Short date pattern |
| D | Long Date pattern |
| f | Full date/Time(Short time) |
| F | Full date/Time (Long time) |
| g | General date/time(short time) |
| G | General date/time(long time) |
| M/m | Month-day pattern |
| t | Short time |
| T | Long time |
| Y /y | Year month pattern |
| ddd | Day of week |
| dddd | Long name day of week |
| MM | Month in a number format |
| MMM | Name of the month |
| ss | Second (0 to 59) |
| tt | AM or PM |
| HH | Hours (00 to 59) |

**Programs of module 2**

1. **Calculator**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Calculator

{

class Program

{

public static void Addition(int a, int b)

{

int result = a + b;

Console.WriteLine("Addition is: {0}", result);

}

public static void Subtraction(int a, int b)

{

int result = a - b;

Console.WriteLine("Subtraction is: {0}", result);

}

public static void Multiplication(int a, int b)

{

int result = a \* b;

Console.WriteLine("Multiplication is: {0}", result);

}

public static void Division(int a, int b)

{

int result = a / b;

Console.WriteLine("Division is: {0}", result);

}

public static void Module(int a, int b)

{

int result = a % b;

Console.WriteLine("Module is: {0}", result);

}

static void Main(string[] args)

{

Console.WriteLine("Enter first number: ");

int num1 = int.Parse(Console.ReadLine());

Console.WriteLine("Enter second number: ");

int num2 = int.Parse(Console.ReadLine());

Console.WriteLine("Enter Your choice: \n (1)Addition \n (2)Subtraction \n (3)Multiplication \n (4)Division \n (5)Module");

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

if (choice == 1)

{

Program.Addition(num1, num2);

}

else if (choice == 2)

{

Program.Subtraction(num1, num2);

}

else if (choice == 3)

{

Program.Multiplication(num1, num2);

}

else if (choice == 4)

{

Program.Division(num1, num2);

}

else if (choice == 5)

{

Program.Module(num1, num2);

}

else

{

Console.WriteLine("Invalid choice");

}

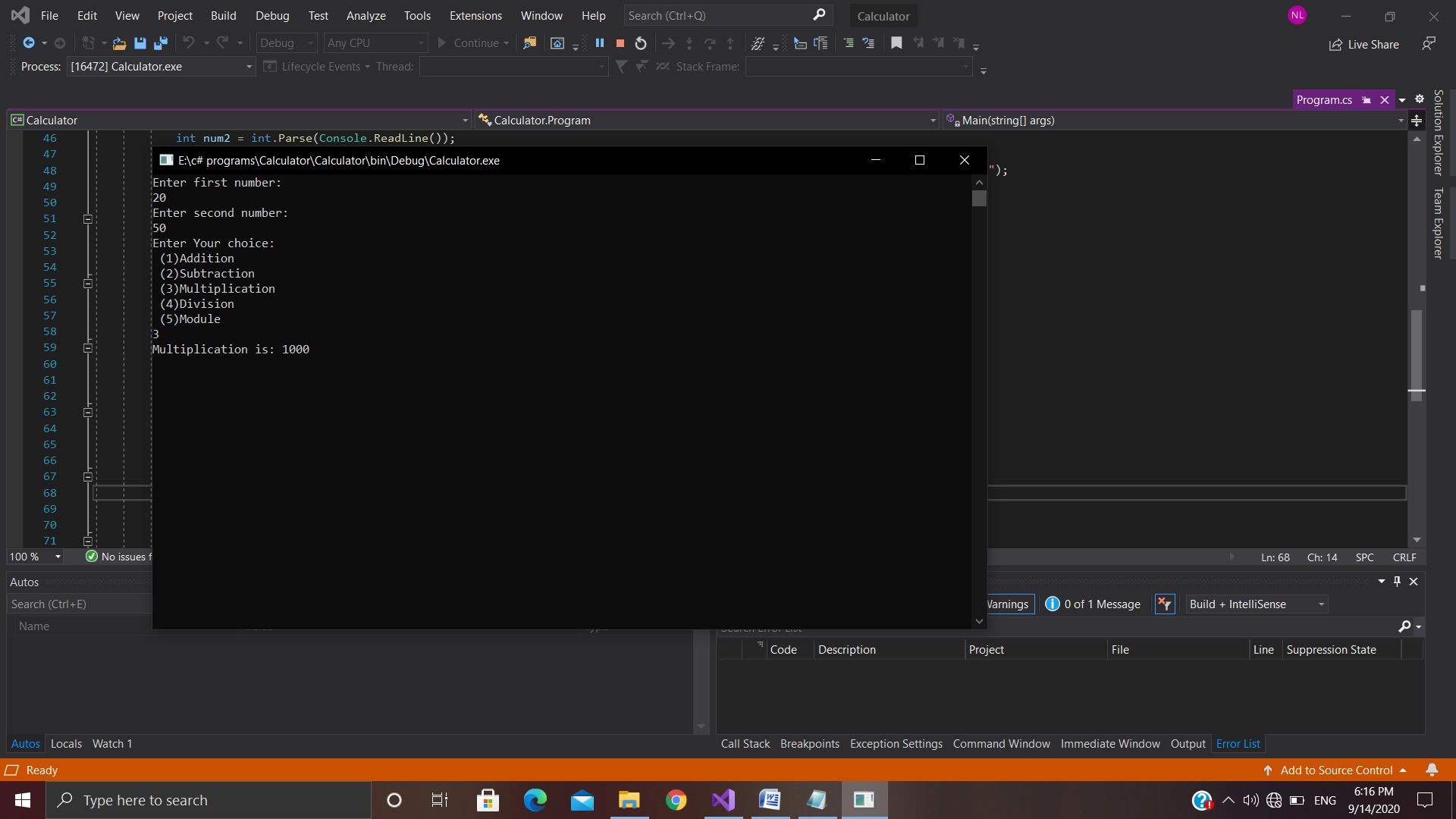
Console.ReadLine();

}

}

}

* **Output**

****

1. **Array and loops**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace one\_dim\_array

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("enter the size of array: ");

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

int[] numbers = new int[size];

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

{

Console.WriteLine("Enter element: ");

numbers[i] = int.Parse(Console.ReadLine());

}

Console.WriteLine("Get element using for each");

foreach (int item in numbers)

{

Console.WriteLine(item);

}

Console.WriteLine("Get element using while loop");

int j = 0;

while ( j < size)

{

Console.WriteLine(numbers[j]);

j++;

}

Console.WriteLine("Get element using do while loop");

int k = 0;

do

{

Console.WriteLine(numbers[k]);

k++;

} while (k<numbers.Length);

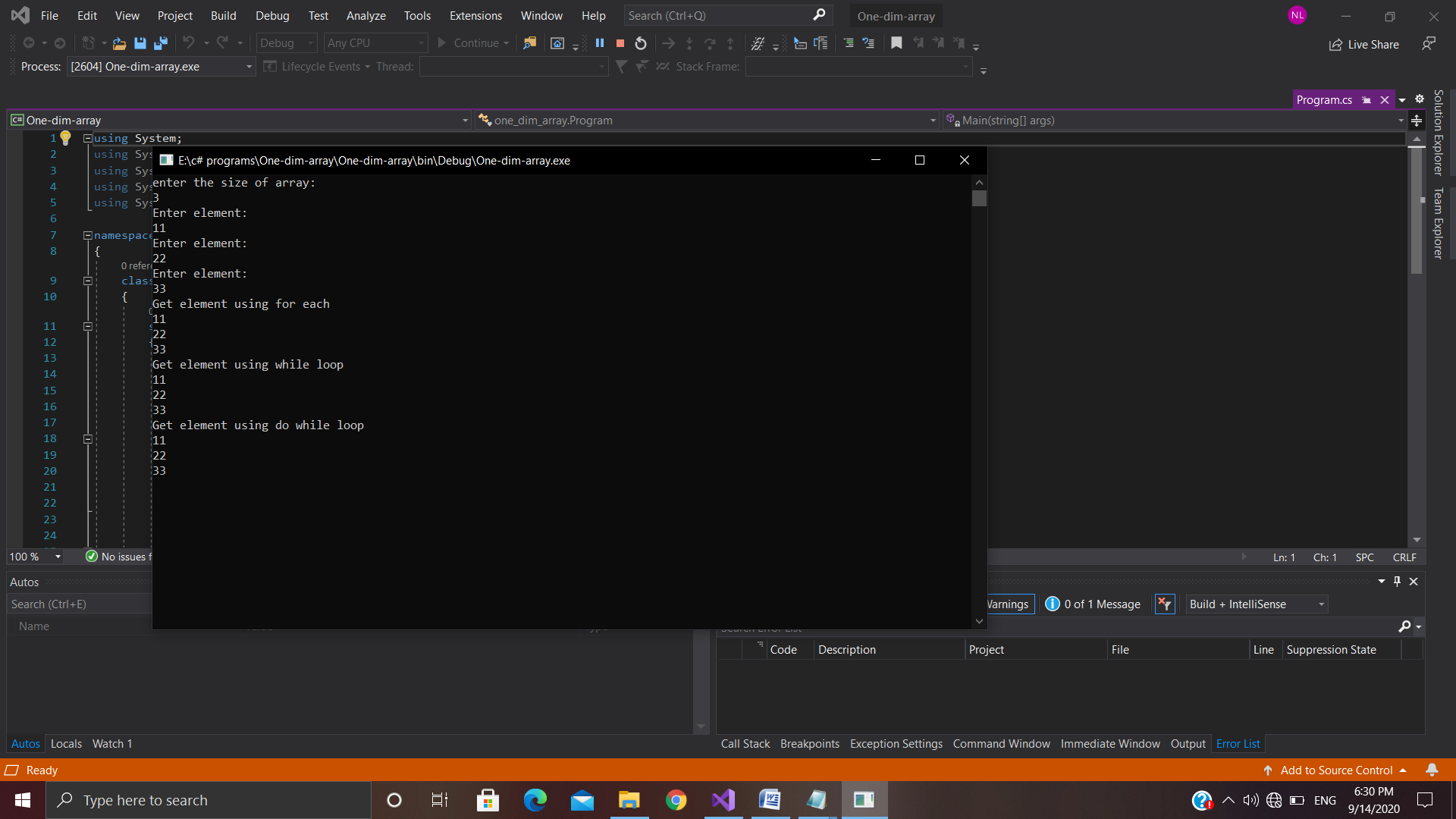
Console.ReadLine();

}

}

}

* **Output**



1. **Two-dimensional array**

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace two\_dim\_array

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Rectangular array");

int[,] items = new int[3, 3]

{

{ 1,2,3},

{ 4,5,6},

{ 7,8,9}

};

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

{

for (int j = 0; j < items.GetLength(1); j++)

{

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

}

Console.WriteLine();

}

Console.WriteLine("");

Console.WriteLine("By using for each");

foreach (int item in items)

{

Console.Write(" " + item);

}

Console.WriteLine("\n");

Console.WriteLine("Jagged array");

string[][] names = new string[2][];

{

names[0] = new string[] { "milk", "coffe", "tea" };

names[1] = new string[] { "bread", "Butter", "chease", "jam" };

};

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

{

for (int j = 0; j < names[i].Length; j++)

{

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

}

Console.WriteLine();

}

Console.WriteLine("");

Console.WriteLine("By using for each");

foreach (string[] item in names)

{

foreach (string s in item)

{

Console.WriteLine(" " + s);

}

}

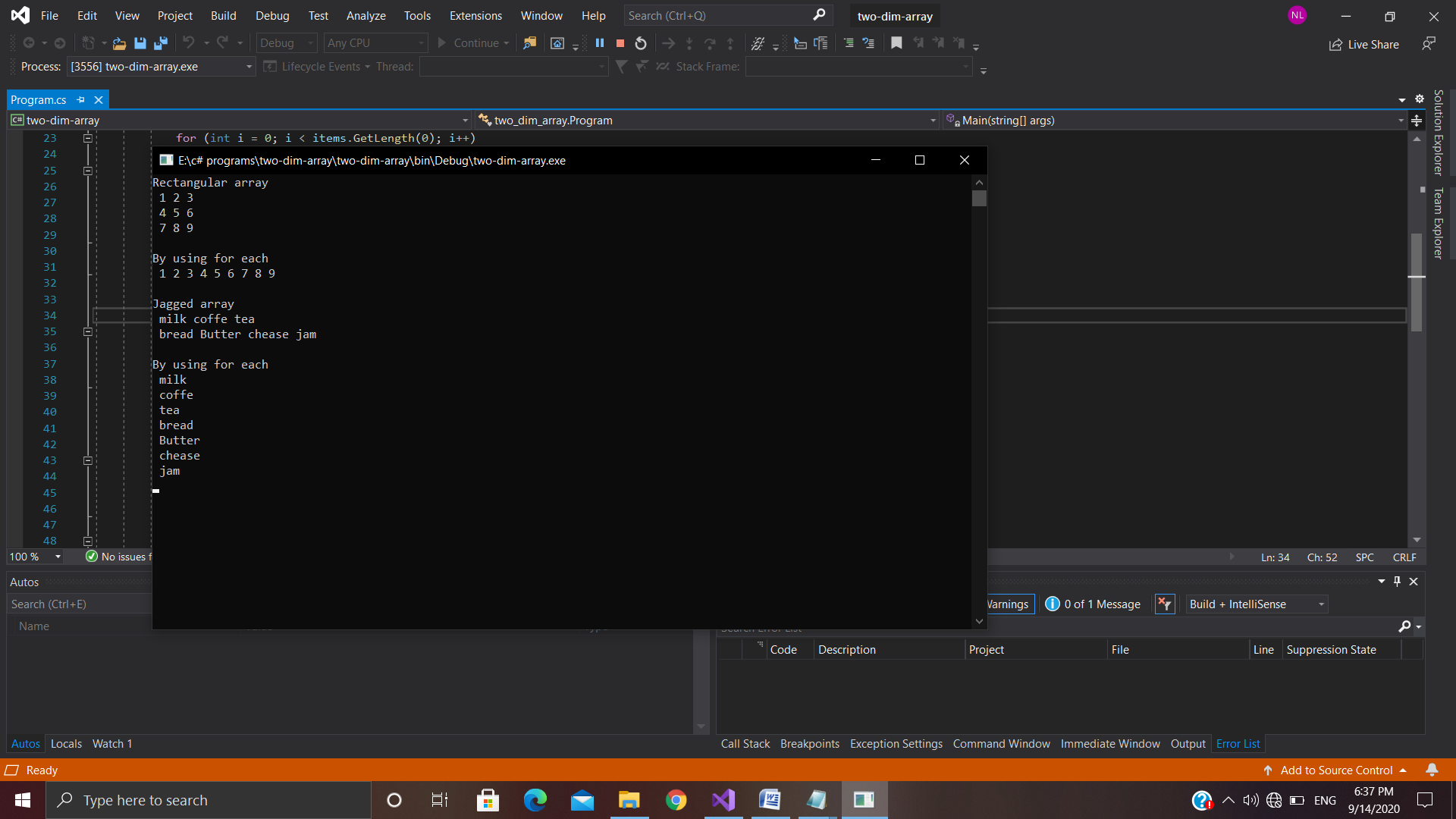
Console.ReadLine();

}

}

}

* **Output**



1. **Defining and calling method (Value type)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ValueTypeMethod

{

class Program

{

public static void add(int a)

{

a += 5;

Console.WriteLine("Square is: "+a);

}

static void Main(string[] args)

{

int num = 5;

Console.WriteLine("value before calling method: " + num);

Program.add(num);

Console.WriteLine("value after calling method: "+ num);

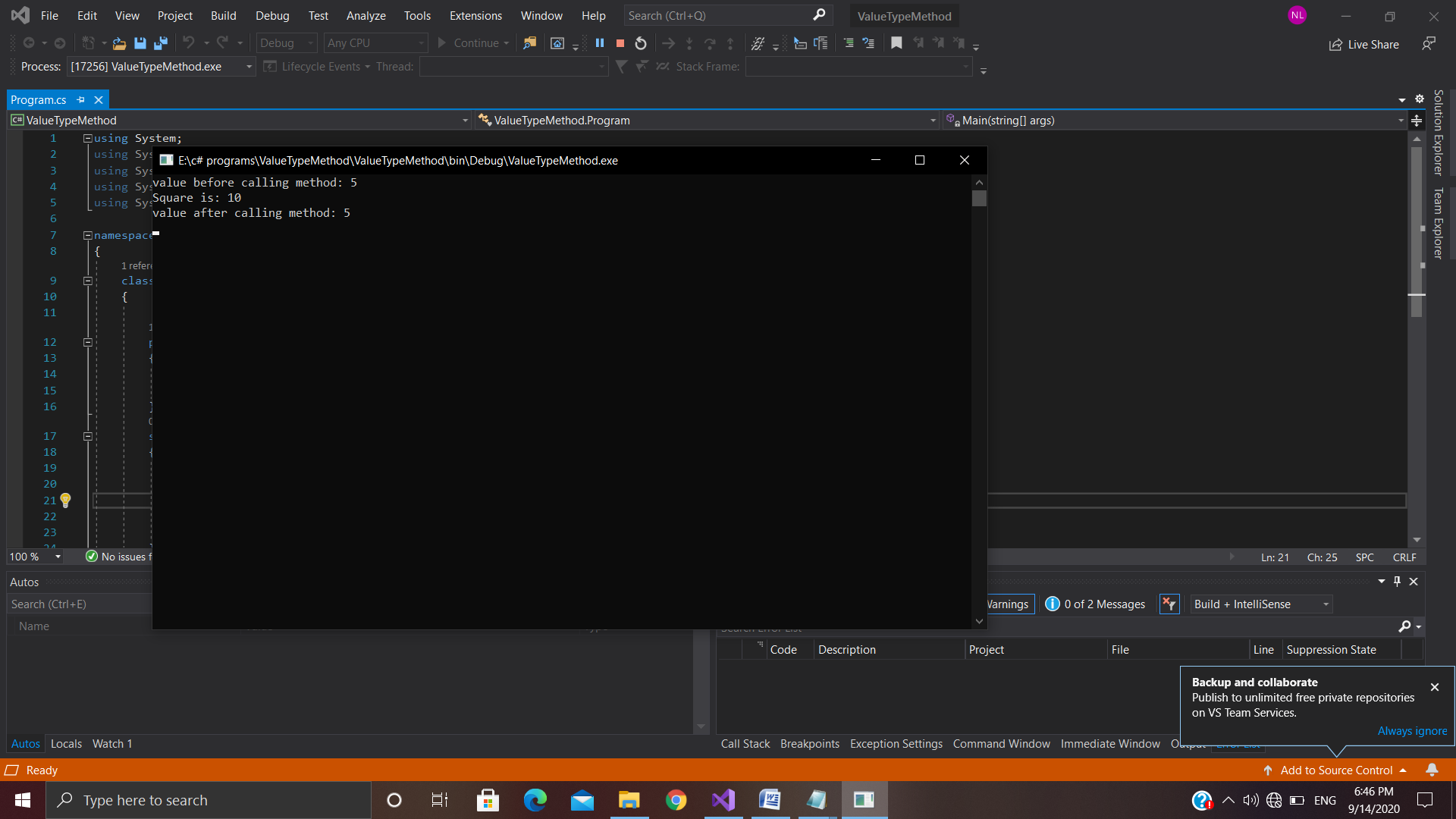
Console.ReadLine();

}

}

}

* **Output**



**(5)Defining and calling method (Ref type)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RefTypeMethod

{

class Program

{

public static void add(ref int a)

{

a += 5;

Console.WriteLine("addition is: "+a);

}

static void Main(string[] args)

{

int num = 8;

Console.WriteLine("value before calling method: "+num);

Program.add(ref num);

Console.WriteLine("value after calling method: "+num);

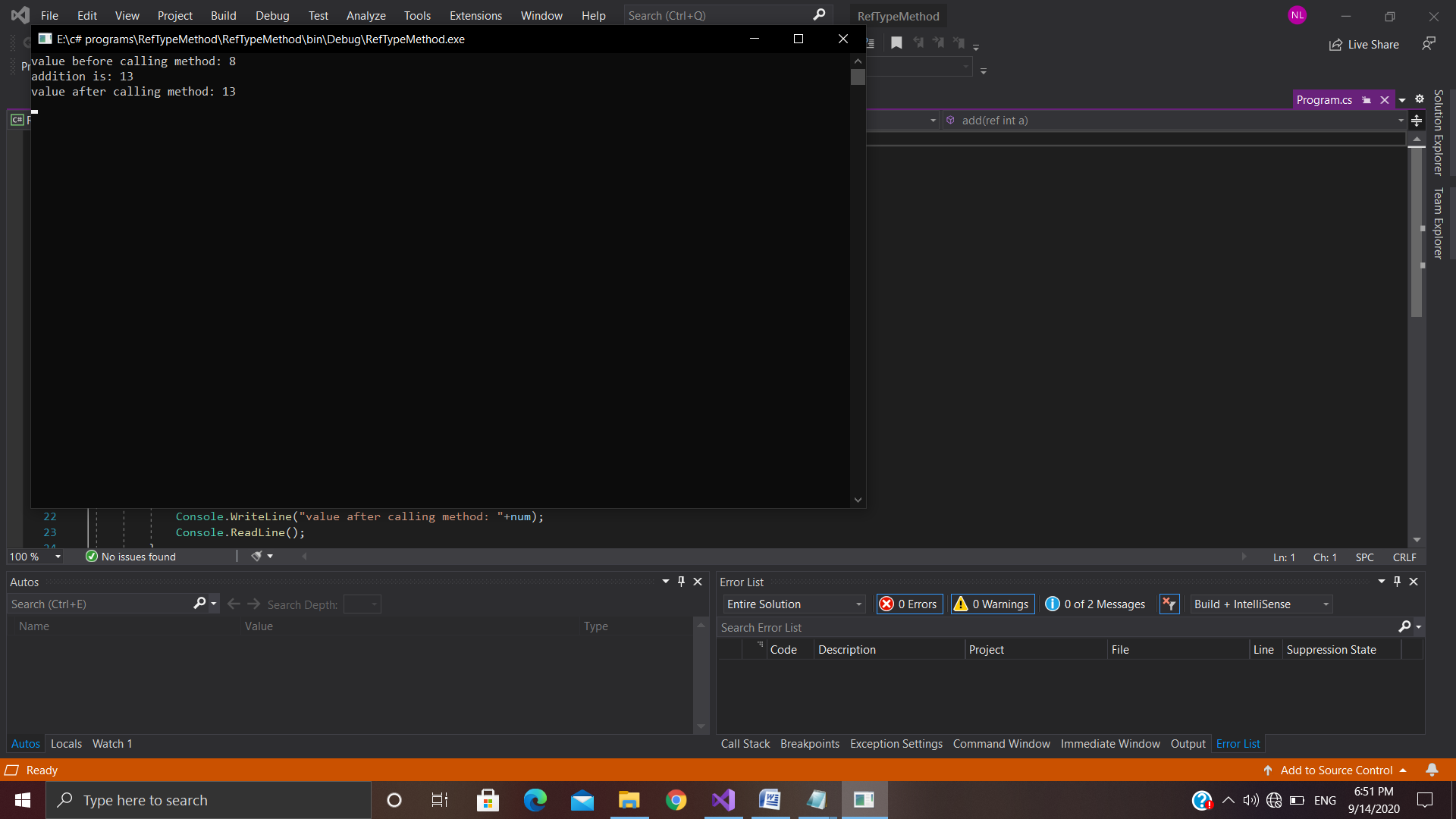
Console.ReadLine();

}

}

}

* **Output**



**(6)Defining and calling method (Optional type)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace OptionalParameter

{

class Program

{

public static void number(int a = 10, int b = 20)

{

Console.WriteLine("First number is : "+a);

Console.WriteLine("Second number is : "+b);

}

static void Main(string[] args)

{

Program.number(b:30);

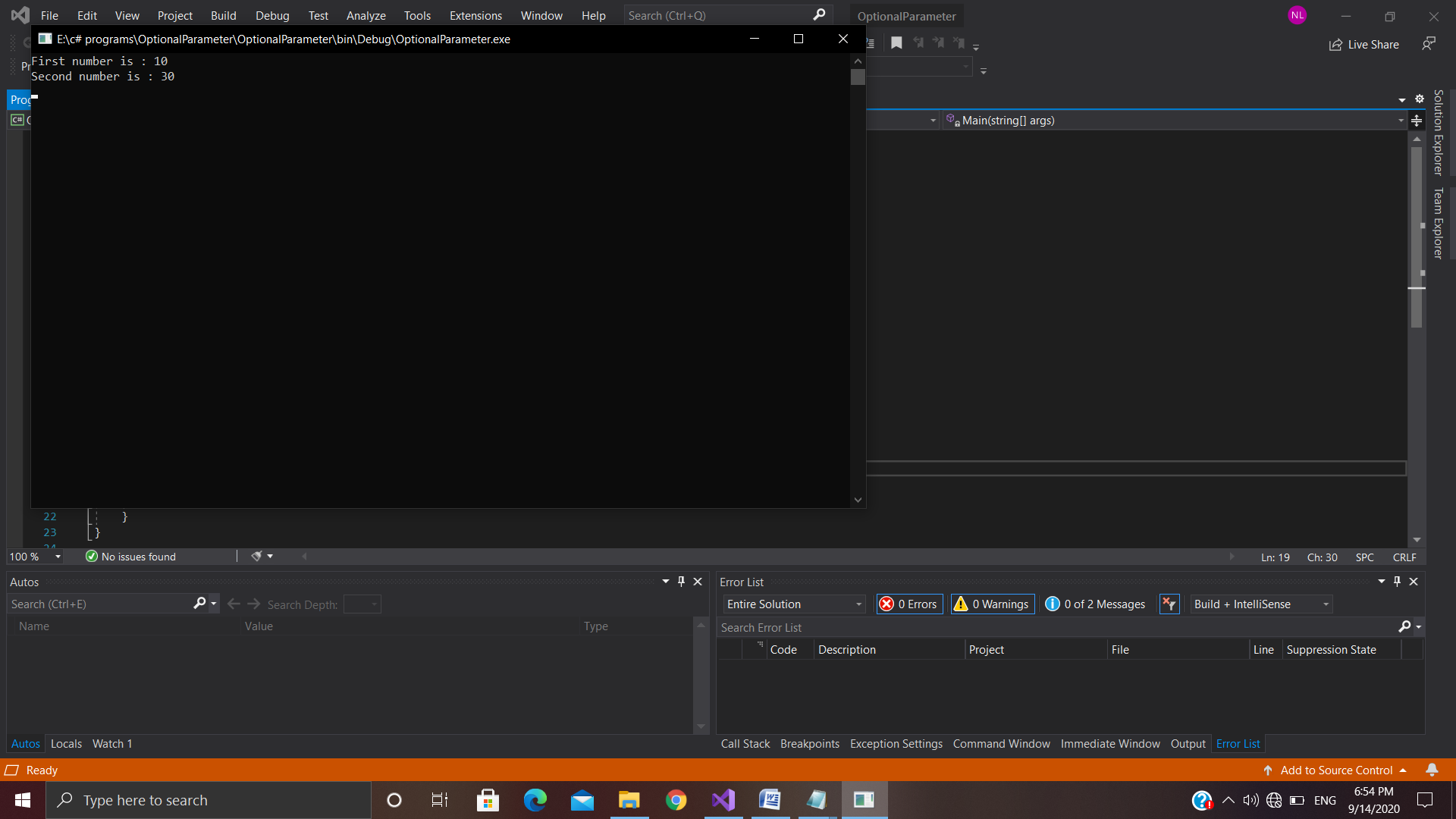
Console.ReadLine();

}

}

}

* **Output**



**(7)String Class**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StringClass

{

class Program

{

static void Main(string[] args)

{

string s = "Hello ";

Console.WriteLine("string is : "+s);

Console.WriteLine("Length of the string is : " + s.Length);

Console.WriteLine("use of clone : " + s.Clone());

Console.WriteLine("use of toupper : " + s.ToUpper());

Console.WriteLine("use of tolower : " + s.ToLower());

Console.WriteLine("use of indexof : " + s.IndexOf('l'));

Console.WriteLine("use of lastindexof : " + s.LastIndexOf('l'));

string b = s.Replace("lo", "looo..!");

Console.WriteLine("Use of replace: " + b);

string a = "world";

Console.WriteLine("Another string is : "+ a);

string c = string.Concat(s, a);

Console.WriteLine("After concatination : " + c);

Console.WriteLine("compare using compare : " + string.Compare(a, s));

Console.WriteLine("compare using Equal : " + s.Equals(a));

Console.WriteLine("use of substring : " + c.Substring(5));

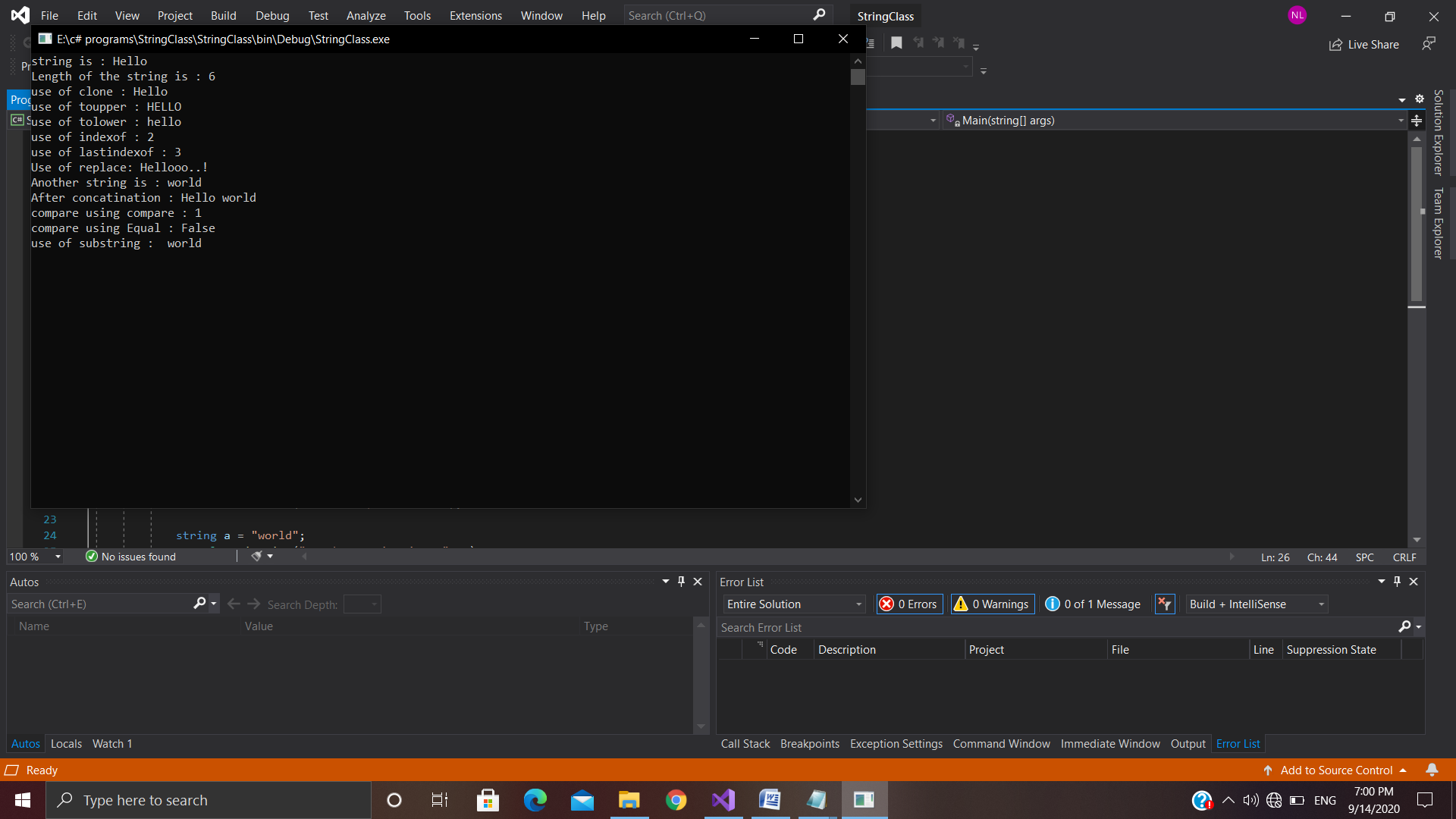
Console.ReadLine();

}

}

}

* **Output**



**(8)Datetime Class**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Datetime

{

class Program

{

static void Main(string[] args)

{

DateTime dt = DateTime.Now;

Console.WriteLine("Datetime using (d) : {0:d}",dt);

Console.WriteLine("Datetime using (D) : {0:D}", dt);

Console.WriteLine("Datetime using (f) : {0:f}", dt);

Console.WriteLine("Datetime using (F) : {0:F}", dt);

Console.WriteLine("Datetime using (g) : {0:g}", dt);

Console.WriteLine("Datetime using (G) : {0:G}", dt);

Console.WriteLine("Datetime using (m/M) : {0:m}", dt);

Console.WriteLine("Datetime using (MM) : {0:MM}", dt);

Console.WriteLine("Datetime using (MMM) : {0:MMM}", dt);

Console.WriteLine("Datetime using (y/Y) : {0:y}", dt);

Console.WriteLine("Datetime using (t) : {0:t}", dt);

Console.WriteLine("Datetime using (tt) : {0:tt}", dt);

Console.WriteLine("Datetime using (T) : {0:T}", dt);

Console.WriteLine("Datetime using (ddd) : {0:ddd}", dt);

Console.WriteLine("Datetime using (dddd) : {0:dddd}", dt);

Console.WriteLine("Datetime using (ss) : {0:ss}", dt);

Console.WriteLine("Datetime using (HH) : {0:HH}", dt);

Console.ReadLine();

}

}

}

* **Output**

