SUMMER 20

Visual Programming

Name : Muhammad Abdullah Minhas

I.D : 13864

Teacher : Sir Ayub

Date : 19th August 19, 2020

Question No 1

How to write Hello program in C# and explain in detail?

Ans) **CODE**

```
 using System;


namespace HelloWorldApplication {
   class HelloWorld {
      static void Main(string[] args) {
         /* my first program in C# */
         Console.WriteLine("Hello World");
         Console.ReadKey();
      }
   }
}
```

Output

```
Hello World
```

Explanation

•The first line of the program using System; - the **using** keyword is used to include the System namespace in the program. A program generally has multiple using statements.

•The next line has the **namespace declaration**. A namespace is a collection of classes. The HelloWorldApplication namespace contains the class HelloWorld.

•The next line has a **class declaration,** the class HelloWorld contains the data and method definitions that your program uses. Classes generally contain multiple methods. Methods define the behavior of the class. However, the HelloWorld class has only one method Main.

•The next line defines the Main method, which is the entry point for all C# programs. The Main method states what the class does when executed.

•The next line **/*...*/** is ignored by the compiler and it is put to add comments in the program.

•The Main method specifies its behavior with the statement **Console.WriteLine**("Hello World");

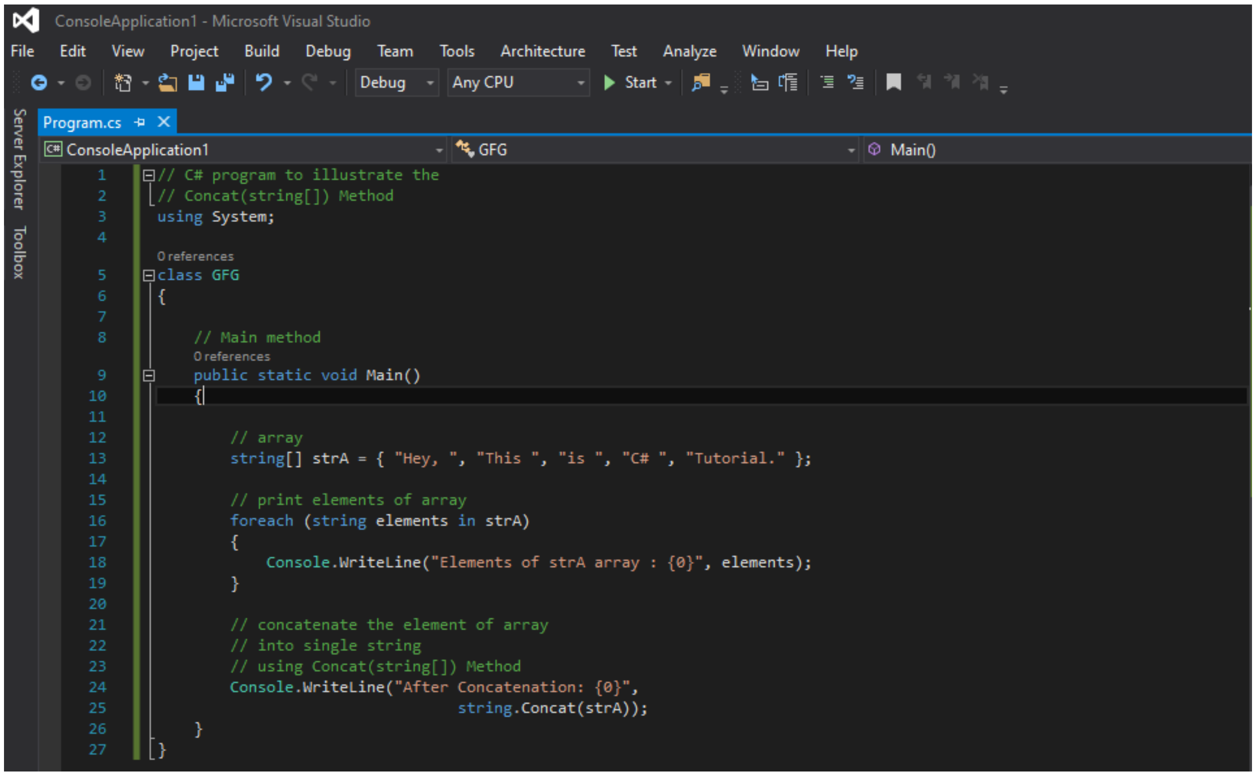
WriteLine is a method of the Console class defined in the System namespace. This statement causes the message "Hello, World!" to be displayed on the screen.

•The last line **Console.ReadKey();** is for the VS.NET Users. This makes the program wait for a key press and it prevents the screen from running and closing quickly when the program is launched from Visual Studio .NET.

Question No 2

a.Write a simple program C # by taking 2 different strings in text boxes concatenate them and display with the help of message box and explain in detail.

Ans)


ConsoleApplication1 - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Architecture Test Analyze Window Help
Debug
Any CPU
Start
Program.cs
ConsoleApplication1
GFG
Main()
Server Explorer
Toolbox

```
// C# program to illustrate the
// Concat(string[]) Method
using System;

0 references
class GFG
{

    // Main method
    0 references
    public static void Main()
    {

        // array
        string[] strA = { "Hey, ", "This ", "is ", "C# ", "Tutorial." };

        // print elements of array
        foreach (string elements in strA)
        {
            Console.WriteLine("Elements of strA array : {0}", elements);
        }

        // concatenate the element of array
        // into single string
        // using Concat(string[]) Method
        Console.WriteLine("After Concatenation: {0}",
                                string.Concat(strA));
    }
}
```

In the following code there is coma between each string which helps it to make many sub strings in one string without writing each string in another line

Code

```
// C# program to illustrate the
// Concat(string[]) Method
using System;

class GFG {

    // Main method
    public static void Main()
    {

        // array
        string[] strA = {"Hey, ", "This ","is ", "C# ", "Tutorial."};

        // print elements of array
        foreach(string elements in strA)
        {
            Console.WriteLine("Elements of strA array : {0}", elements);
        }

        // concatenate the element of array
        // into single string
        // using Concat(string[]) Method
        Console.WriteLine("After Concatenation: {0}",
                                string.Concat(strA));
    }
}
```

b. Write about different types of type conversions in available in C #.

Ans) Type conversion is converting one type of data to another type. It is also known as Type Casting. In C#, type casting has two forms ;

Implicit type conversion

These conversions are performed by C# in a type-safe manner. For example, are conversions from smaller to larger integral types and conversions from derived classes to base classes.

Explicit type conversion

These conversions are done explicitly by users using the pre-defined functions. Explicit conversions require a cast operator.

Types Of Conversion

1 ToBoolean

Converts a type to a Boolean value, where possible.

2 ToByte

Converts a type to a byte.

3 ToChar

Converts a type to a single Unicode character, where possible.

4 ToDateTime

Converts a type (integer or string type) to date-time structures.

5 ToDecimal

Converts a floating point or integer type to a decimal type.

6 ToDouble

Converts a type to a double type.

7 ToInt16

Converts a type to a 16-bit integer.

8 ToInt32

Converts a type to a 32-bit integer.

9 ToInt64

Converts a type to a 64-bit integer.

10 ToSbyte

Converts a type to a signed byte type.

11 ToSingle

Converts a type to a small floating point number.

12 ToString

Converts a type to a string.

13 ToType

Converts a type to a specified type.

14 ToUInt16

Converts a type to an unsigned int type.

15 ToUInt32

Converts a type to an unsigned long type.

16 ToUInt64

Converts a type to an unsigned big integer.

Question No 3

a.What are constant in C# discuss in detail?

Ans) **Constants** can be marked as public, private, protected, internal, protected internal or private protected. These access modifiers define how users of the class can access the constant

Constants are accessed as if they were static fields because the value of the constant is the same for all instances of the type. You do not use the static keyword to declare them. Expressions that are not in the class that defines the constant must use the class name, a period, and the name of the constant to access the constant. For example:

int birthstones = Calendar.Months;

Integer Literals

An integer literal can be a decimal, or hexadecimal constant. A prefix specifies the base or radix: 0x or 0X for hexadecimal, and there is no prefix id for decimal.

An integer literal can also have a suffix that is a combination of U and L, for unsigned and long, respectively. The suffix can be uppercase or lowercase and can be in any order.

Here are some examples of integer literals –

```
212        /* Legal */
215u       /* Legal */
0xFeeL     /* Legal */
```

Following are other examples of various types of Integer literals –

```
85         /* decimal */
0x4b       /* hexadecimal */
30         /* int */
30u        /* unsigned int */
30l        /* long */
30ul       /* unsigned long */
```

Floating-point Literals

A floating-point literal has an integer part, a decimal point, a fractional part, and an exponent part. You can represent floating point literals either in decimal form or exponential form.

Here are some examples of floating-point literals

3.14159   /* Legal */

314159E-5F   /* Legal */

510E   /* Illegal: incomplete exponent */

210f   /* Illegal: no decimal or exponent */

.e55   /* Illegal: missing integer or fraction */

While representing in decimal form, you must include the decimal point, the exponent, or both; and while representing using exponential form you must include the integer part, the fractional part, or both. The signed exponent is introduced by e or E.

Character Constants

Character literals are enclosed in single quotes. For example, 'x' and can be stored in a simple variable of char type. A character literal can be a plain character (such as 'x'), an escape sequence (such as '\t'), or a universal character (such as '\u02C0').

There are certain characters in C# when they are preceded by a backslash. They have special meaning and they are used to represent like newline (\n) or tab (\t).

Code

```
using System;

namespace EscapeChar {
   class Program {
      static void Main(string[] args) {
         Console.WriteLine("Hello\tWorld\n\n");
         Console.ReadLine();
      }
   }
}
```

String Literals

String literals or constants are enclosed in double quotes "" or with @"". A string contains characters that are similar to character literals: plain characters, escape sequences, and universal characters.

You can break a long line into multiple lines using string literals and separating the parts using whitespaces.

Code

```
"hello, dear"

"hello, \

dear"

"hello, " "d" "ear"

@"hello dear"
```

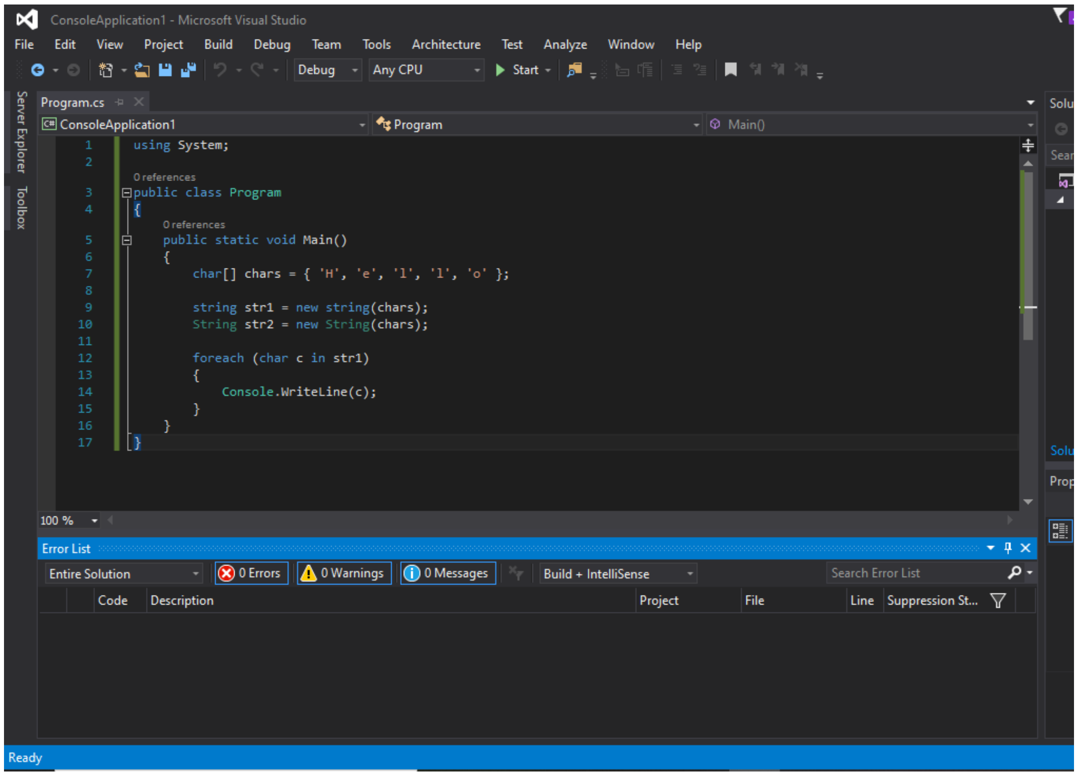
Defining Constants

Constants are defined using the const keyword. Syntax for defining a constant is −

```
const <data_type> <constant_name> = value;
```

b. Write a program on string literal and explain in detail.

Ans) **Code**



```
using System;

0 references
public class Program
{
    0 references
    public static void Main()
    {
        char[] chars = { 'H', 'e', 'l', 'l', 'o' };

        string str1 = new string(chars);
        String str2 = new String(chars);

        foreach (char c in str1)
        {
            Console.WriteLine(c);
        }
    }
}
```

```
using System;

public class Program
{
        public static void Main()
        {
                char[] chars = {'H', 'e', 'l', 'l', 'o'};

                string str1 = new string (chars);
                String str2 = new String(chars);

                foreach (char c in str1)
                {
                        Console.WriteLine(c);
                }
        }
} // Each string will be in a new line 'H' will be in first 'e' will be in second line and so on
```