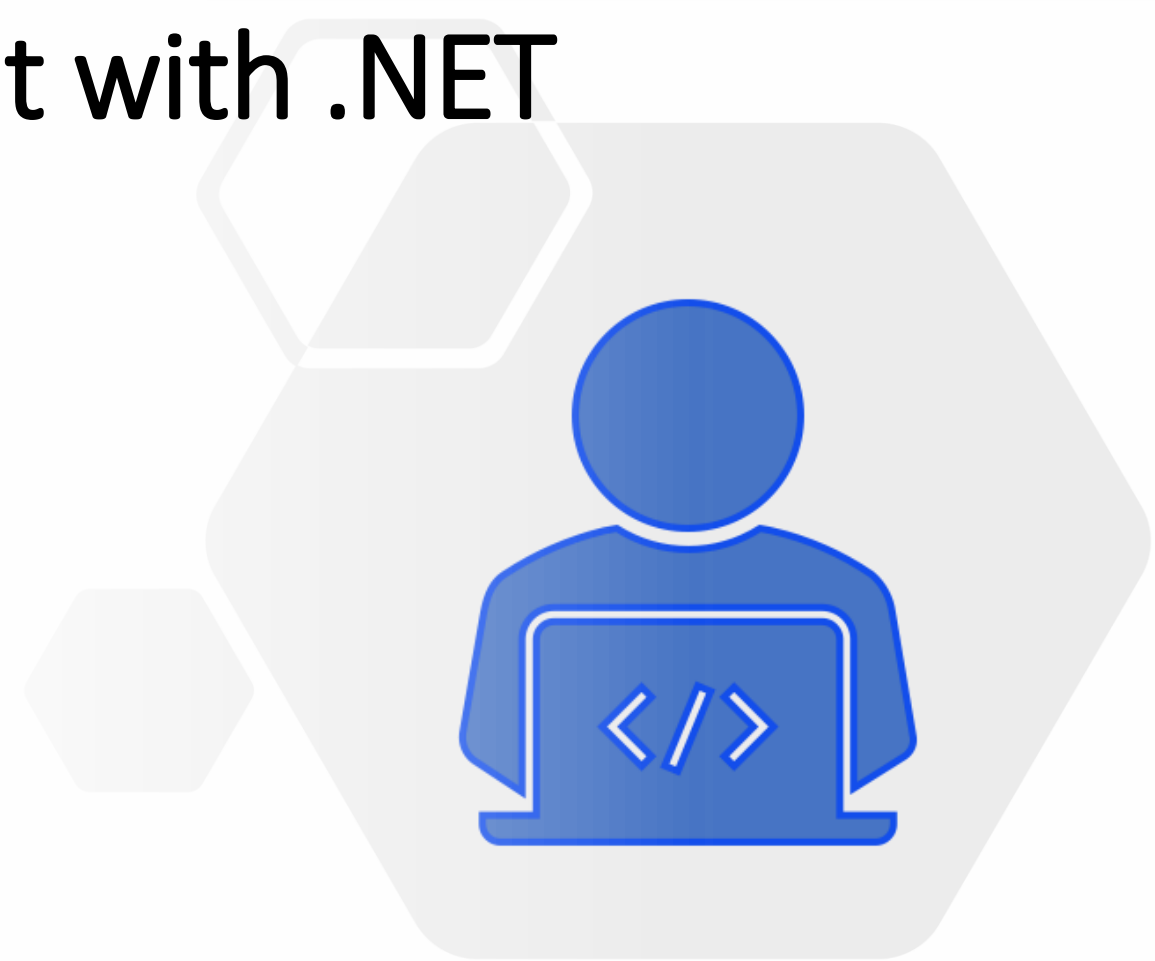
31927 32998:
Application Development with .NET

Week-2 Lecture

C# Programming Basics

Part-1



Outline

- Comments in C#
- Built in Data types
- Variable and Constants
- Value type vs Reference types
- Input / Output in C#
- Operators
- Conditions and loops
- Strings and Enums

Comments in C#

- Single line comments

```
// This Program calculates the sum and two numbers
```

- Multi-line comments

```
/*
This Program calculates the sum and two numbers
Author : Mr XYZ ABC
Date: 01/01/2018
*/
```

Anything following // or between /* … */ are ignored by the compiler

Build-in Data types

Keyword	Type of Values	Example	Operations
byte	8-bit unsigned integer	Numbers between 0-255	Add, subtract, multiply, divide, etc.
uint	32-bit unsigned integer type	Numbers between 0 to 4,294,967,295	
int	32-bit signed integers	-12, 0, 3467, etc.	
long	64-bit signed integer		
float	Single-precision Floating point numbers	3.1234, 78.096, etc	
double	Double-precision Floating point numbers		
bool	Boolean value	True or False (default)	or, and, not
char	16 bit Unicode characters	‘A’, ‘#’, ‘@’, ‘\0’(default)	Comparison

Reference: C# Black book, Matt Telles

Variables

- Variables are name given to a storage area to be used/manipulated in a computer program

- Variable declarations take the form:

```
<datatype> <identifier>;    (Example: int a;)
<datatype> <identifier_list>; (Example: int a,b,c;)
```

- All variables are initialized to default values during declaration.

Reference: C# Black book, Matt Telles

Constants

- Constants refer to fixed values that a program may not alter during its execution.
- Constants are also called literals.
- Constants can be of any basic data type such as integer constants, floating constant, a character constant, or a string literal.
- Enumeration constants are also available.

Reference: C# Black book, Matt Telles

Character and String Constants

- **Character literals** are enclosed in single quotes, e.g. ‘a’, ‘A’, etc.

- Character literals can be simple characters, escape sequence, or a universal character.

- **String literals** or constants are enclosed in double quotes ""

Example: “Hello World”

Common Escape sequence characters

Escape sequence	Meaning
\\	\ character
\'	' character
\"	" character
\?	? character
\a	Alert or bell
\b	Backspace
\f	Form feed
\n	Newline
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab

Reference: https://www.tutorialspoint.com/csharp/csharp_constants.htm

Escape Sequence Example

```
using System;

namespace EscapeChar
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            Console.WriteLine("Welcome \t to\nHello\nWorld\n\n");
            Console.ReadLine();
        }
    }
}
```

Output

```
Welcome          to
Hello
World
```

Reference: https://www.tutorialspoint.com/csharp/csharp_constants.htm

Defining Constants

- **Const** keyword is used to define user defined constants

Syntax:

const <data type> <constant name> = value;

Example:

```
const double pi = 3.14159;
const double interestRate = 2.25;
```

Reference: https://www.tutorialspoint.com/csharp/csharp_constants.htm

Reference types

- The reference types do not contain the actual data stored in a variable
- They contain a reference to the variables.
- In short, they refer to a memory location

- Built-in reference types are: object, dynamic and string

- Example:

String Str = “Hello World”;

Reference: https://www.tutorialspoint.com/csharp/csharp_constants.htm

Value type Vs Reference types

Value types	Reference types
Copy semantics: Variables or objects or value types have their own copy of the data.	**Reference semantics:** Points to a location in the memory that contains the actual data.
Allocated on Stack	Allocated on heap
○ Simple type: ▪ bool ▪ byte, int, long, char… ▪ decimal ▪ float, double ○ struct type ○ enum type	• One of: ○ Class ○ Interface ○ Array ○ Delegate ○ String

Reference: https://www.tutorialspoint.com/csharp/csharp_constants.htm

Boxing, unboxing

- Conversion of a Value type to Reference type is Boxing
- The opposite is unboxing!

- Example:

```
int Price = 100;

Object PriceObj = Price; //Boxing

Price = (int) PriceObj; //Unboxing
```

Reference: https://www.tutorialspoint.com/csharp/csharp_constants.htm

Input / Output C#

1. Methods for reading user input and displaying output/text

Method	Description	Example
Console.ReadLine()	Method to read a line of input from standard input stream	String userInput; userInput = Console.ReadLine();
Console.Read ()	Method to read next character from standard input stream	String userInput; userInput = Console.Read();
Console.ReadKey()	Method obtains the next key pressed by user	Console.WriteLine("Press any key to continue..."); Console.ReadKey();
Console.Writeline() Console.Write()	Prints the provided String and adds a new line Only prints the String provided without new line	Console.Writeline("The userInput is {0}", userInput); {0} →placehold for variables

Input / Output C#

2. Methods for reading numeric inputs from user:
   - Reading numeric input is little tricky!
   - Readline() is still used but it reads everything as String!
   - A Conversion from String to required numeric (int, float, etc) is required.
   - Methods from Convert class are used!

Method	Description	Example

ToInt32()	Converts the value to integer	Console.Write("Enter integer value: ");
		userInput = Console.ReadLine();
		intValue = Convert.ToInt32(userInput);

ToDouble()	Converts the value to Double	doubleValue = Convert. ToDouble(userInput);

For Complete list Check → https://msdn.microsoft.com/en-us/library/system.convert(v=vs.110).aspx

Input / Output C#

Example for reading numeric inputs from user

```
using System;
// Program to demonstrate
// reading numeric input from user

namespace Week2Programs
{
    0 references
    class InputOutputDemo
    {
        0 references
        static void Main(string[] args)
        {
            // Initialize variables
            string userInput;
            int intValue;
            double doubleValue;

            // Display instructions for the user
            Console.Write("Enter an integer value: ");
            // Read the user input
            userInput = Console.ReadLine();
            // Converts to integer
            intValue = Convert.ToInt32(userInput);
            Console.WriteLine("You have entered {0}", intValue);

            // For double values
            Console.Write("Enter a double value: ");
            userInput = Console.ReadLine();
            /* Converts to double type */
            doubleValue = Convert.ToDouble(userInput);
            Console.WriteLine("You have entered {0}", doubleValue);

            Console.ReadKey();

        }
    }
}
```

Output

```
Enter an integer value: 12345
You have entered 12345
Enter a double value: 123.123456
You have entered 123.123456
```

Operators in C#

1. Arithmetic operators:

Operator	Description	Usage example
+	Addition	Z = X + Y
-	Subtraction	Z = X – Y
*	Multiplication	Z = X * Y
/	Divides numerator by de-numerator	Z = X / Y
%	Modulus Operator and remainder of after an integer division	Z = X % Y
++	Increment operator, increases value by 1	Z++, ++Z
--	Decrement operator, decreases value by 1	Z--, --Z

Reference: https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/arithmetic-operators

Operators in C#

1. Arithmetic operators: Example

```
using System;

namespace Week2Programs
{
    0 references
    class ArithmeticOperatoDemo
    {
        // The Program Shows a demo on the Arithematic Operator usage

        0 references
        static void Main(string[] args)
        {
            // Variable declaration
            int sum, product, diff, modulo;
            float divResult;
            int X = 10, Y = 20;

            // Perform the Arithematic operations
            sum = X + Y;
            diff = X - Y;
            product = X * Y;
            divResult =  X/Y;
            modulo = X % Y;

            // Display results
            Console.WriteLine("{0} + {1} = {2}", X, Y, sum);
            Console.WriteLine("{0} - {1} = {2}", X, Y, diff);
            Console.WriteLine("{0} * {1} = {2}", X, Y, product);
            Console.WriteLine("{0} / {1} = {2}", X, Y, divResult);
            Console.WriteLine("{0} % {1} = {2}", X, Y, modulo);
            Console.Read();
        }
    }
}
```

Output

```
10 + 20 = 30
10 - 20 = -10
10 * 20 = 200
10 / 20 = 0
10 % 20 = 10
```

Should be 0.5 ?

Try and see if this works:
divResult = (float) X/Y ;

Reference: https://www.tutorialspoint.com/csharp/csharp_constants.htm

Operators in C#

2. Relational operators

Let, X = 10 and Y =20

Operator	Description	Example
==	Check for Equality	X==Y is false
!=	Check for inequality	X==Y is true
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	X>Y is false
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true	X>=Y is false
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true	X<Y is true
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	X<=Y is true

Reference: https://www.tutorialspoint.com/csharp/csharp_operators.htm

Operators in C#

3. Logical operators

Let, X = true and Y = false

Operator	Description	Example
&&	Logical AND, if both the operands are non-zero then condition is true	(X && Y) is false
||	Logical OR, if any on of the operands are non-zero then condition is true	(X || Y) is true
!	Logical NOT, used to reverse the logical state of its operand, e.g.: if a condition is true, NOT makes it false.	!Y is true !(X && Y) is true

Reference: https://www.tutorialspoint.com/csharp/csharp_operators.htm

Operators in C#

3. Relational and Logical operators example

```
using System;
/* The Program shows a demo on Relational
   and Logical Operator usage

    Problem: Check whether a number is even
             and if it is greater than 10
*/
namespace Week2Programs
{
    0 references
    class RelationAndLogicalOperatorDemo
    {
        0 references
        static void Main(string[] args)
        {
            // Variable Declaration
            int numToCheck = 12;

            // Check for even and odd
            if((numToCheck % 2) == 0 && numToCheck > 10)
            {
                Console.WriteLine("The number is even and greater than 10");
            }
            else
            {
                Console.WriteLine("The number is ether not even or not greater than 10");
            }
            Console.Read();
        }
    }
}
```

Output

```
The number is even and greater than 10
```

Reference: https://www.tutorialspoint.com/csharp/csharp_operators.htm

Operators in C#

4. Assignment operators

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	X =10
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left oper	X +=10, same as X = X +10
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	X -=10, same as X = X -10
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	X *=10, same as X = X *10
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	X /=10, same as X = X /10
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	X %=10, same as X = X %10

Reference: https://www.tutorialspoint.com/csharp/csharp_operators.htm

Operators in C#

4. Assignment operators example

```
using System;
// The Program shows a demo on assignment operators

namespace Week2Programs
{
    0 references
    class AssignmentOperatorDemo
    {
        0 references
        static void Main(string[] args)
        {
            //Variable Declaration
            int number = 10;

            // Display the results of
            // various assignment operation
            Console.WriteLine("Number = {0}", number);

            number += 5;
            Console.WriteLine("Number after += operation = {0}", number);

            number -= 5;
            Console.WriteLine("Number after -= operation = {0}", number);

            number *= 5;
            Console.WriteLine("Number after *= operation = {0}", number);

            number /= 5;
            Console.WriteLine("Number after /= operation = {0}", number);

            number %= 5;
            Console.WriteLine("Number after += operation = {0}", number);

            Console.Read();
        }
    }
}
```

Output

```
Number = 10
Number after += operation = 15
Number after -= operation = 10
Number after *= operation = 50
Number after /= operation = 10
Number after += operation = 0
```

Reference: https://www.tutorialspoint.com/csharp/csharp_operators.htm

Operators in C#

5. Special/other operators

Operator	Description	Example
.	Member access operator	Console.Writeline(), etc
[]	Index operator used in arrays and collections	A[1], etc
()	Cast operator	Type casting
?:	Ternary operator	If Condition is true ? Then value X : Otherwise value Y A = (5 > 6) ? 5 : 6;
sizeof()	Returns the size of a data type.	sizeof(int), returns 4
typeof()	Returns the type of a class.	typeof(StreamReader);

Reference: https://www.tutorialspoint.com/csharp/csharp_operators.htm

Operators in C#

5. Special/other operators example

```
using System;
// The Programs shows a demo on Special operators

namespace Week2Programs
{
    0 references
    class SpecialOperatorDemo
    {
        0 references
        static void Main(string[] args)
        {
            // Variable Declaration
            int number1 = 10, number2 = 15;

            // Check the size using sizeof()
            Console.WriteLine("The size of int is {0}", sizeof(int));
            Console.WriteLine("The size of bool is {0}", sizeof(bool));

            // Find the highest number using ternary operator
            // and display the result
            int highestNumber = number1 > number2 ? number1 : number2;
            Console.WriteLine("The numbers are {0} and {1}", number1, number2);
            Console.WriteLine("The higest number is {0}", highestNumber);
            Console.Read();
        }
    }
}
```

Output

```
The size of int is 4
The size of bool is 1
The numbers are 10 and 15
The higest number is 15
```

Reference: https://www.tutorialspoint.com/csharp/csharp_operators.htm

Operator Precedence

- Expressions are evaluated from left to right. However, the following general precedence follows.

1. Evaluate expressions in parentheses first.
2. Evaluate * / and %operators second.
3. Evaluate the + and –binary operators third
4. Evaluate Relational operators fourth.
5. Evaluate Logical operators last.

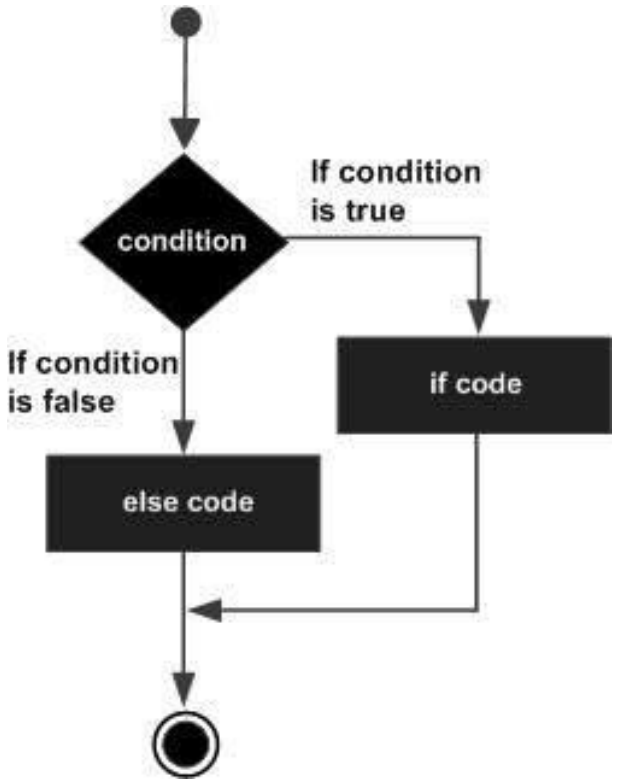
Conditional statements in C#

1. If … Else statement

Syntax:

```
if (Expression){
    // Statements to be executed if the expression is true
}
else{
   // Statements to be executed if the expression if false
}
```

Flow Diagram



Reference: https://www.tutorialspoint.com/csharp/if_else_statement_in_csharp.htm

Conditional statements in C#

1. If … Else variations

If… else if… else Syntax:

```
if (Expression 1){
    // Statements to be executed if the expression 1 is true
}
Else if (Expression 2){
   // Statements to be executed if the expression 2 if false
}
Else if (Expression 3){
   // Statements to be executed if the expression 3 if false
}
Else {
   // Statements to be executed if the none of the
      expressions is true
}
```

Nest if Syntax:

```
if (Expression 1){
    // Statements to be executed if the expression 1 is true

    if (Expression 2){
      // Statements to be executed if the expression 2 if true
    }

}
```

Reference: https://www.tutorialspoint.com/csharp/if_else_statement_in_csharp.htm

Conditional statements in C#

1. If … Else example

```
using System;
/*
 Program to Check whether a number is even or odd or
 equals to zero
*/

namespace Week2Programs
{
    0 references
    class IfElseDemo
    {
        0 references
        static void Main(string[] args)
        {
            // Variable declaration
            int numberToCheck = 10;

            // Check if the number of even or odd and
            // Display the result
            if(numberToCheck == 0)
            {
                Console.WriteLine("The number is zero!");
            }
            else if (numberToCheck % 2 == 0)
            {
                Console.WriteLine("The Number is even!");
            }
            else
            {
                Console.WriteLine("The Number is odd!");
            }
            Console.Read();
        }
    }
}
```

Output

```
The Number is even!
```

Reference: https://www.tutorialspoint.com/csharp/if_else_statement_in_csharp.htm

Conditional statements in C#

2. Switch statement

Syntax:

```
switch (Expression){

    case constant-expression:
        //Statements to be executed
        break;

    case constant-expression:
        //Statements to be executed
        break;

    default :
         //Statements to be execute when none of the cases are true
}
```

Reference: https://www.tutorialspoint.com/csharp/switch_statement_in_csharp.htm

Conditional statements in C#

2. Switch statement example

```
/*
Program to find final result based on grade obtained
by a student, based on the following criteria

    Grade = A or B : High Distinction
    Grade = C : Distinction
    Grade = D : Pass
    Grade = F : Fail

*/
```

```
using System;
/* ...
namespace Week2Programs
{
    0 references
    class SwitchDemo
    {
        0 references
        static void Main(string[] args)
        {
            //Variable Declaration
            char grade = 'C';

            //Find the grade based on the GPA
            switch (grade)
            {
                case 'A':
                    Console.WriteLine("Grade: High Distinction");
                    break;
                case 'B':
                    Console.WriteLine("Grade: High Distinction");
                    break;
                case 'C':
                    Console.WriteLine("Grade: Distinction");
                    break;
                case 'D':
                    Console.WriteLine("Grade: Pass");
                    break;
                default:
                    Console.WriteLine("Grade: Fail");
                    break;
            }
            Console.Read();
        }
    }
}
```

Output

```
Grade: Distinction
```

Reference: https://www.tutorialspoint.com/csharp/switch_statement_in_csharp.htm

Loops in C#

- Loop statements allows to execute a statement or a group of statements multiple times
- C# support both entry controlled and exit controlled loops
- Entry controlled loops:
  - May not be executed at all
  - for loop and while loop
- Exit controlled loops:
  - Will be executed at least once.
  - do … while loop

Loops in C#

1. While loop:

Syntax:

```
while ( condition ) {

    Statements for be executed

}
```

Example

```
using System;
/*
Program to display all odd numbers
between 1 to 10, using While loop
*/

namespace Week2Program
{
    0 references
    class WhileLoopDemo
    {
        0 references
        static void Main(string[] args)
        {
            // Variable declaration
            int controlVar = 1;

            Console.WriteLine("The odd number between 1 and 10 are:");
            // While loop starts here -->
            while (controlVar <= 10)
            {
                // Check for odd number
                if(controlVar % 2 != 0)
                {
                    Console.Write(controlVar + ",");
                }

                // Increment the loop control variable
                controlVar++;
            }
            Console.Read();
        }
    }
}
```

Output

```
The odd number between 1 and 10 are:
1,3,5,7,9,
```

Reference: https://www.tutorialspoint.com/csharp/csharp_loops.htm

Loops in C#

2. For loop:

Syntax:

```
for ( <initialize loop control variable>; <condition>; increment ) {

    Statements for be executed until the condition is true

}
```

Reference: https://www.tutorialspoint.com/csharp/csharp_loops.htm

Loops in C#

2. For loop example:

```
using System;
/*
Program to display all odd number
between 1 to 10, using for loop
*/

namespace Week2Programs
{
    0 references
    class ForLoopDemo
    {
        0 references
        static void Main(string[] args)
        {
            Console.WriteLine("The odd numbers between 1 to 10 are:");

            //For loop starts here -->
            // 1. initial the loop control variable
            // 2. add terminating condition
            // 3. increment
            for(int controlVar=1; controlVar<=10; controlVar++)
            {
                // Check for odd number
                if(controlVar%2 != 0)
                {
                    Console.Write(controlVar + ",");
                }
            }
            Console.Read();
        }
    }
}
```

Output

```
The odd numbers between 1 to 10 are:
1,3,5,7,9,
```

Reference: https://www.tutorialspoint.com/csharp/csharp_loops.htm

Loops in C#

3. do … while loop:

Syntax:

```
do {

    //Statements for be executed till the condition is true

} while (condition);
```

Reference: https://www.tutorialspoint.com/csharp/csharp_loops.htm

Loops in C#

3. do … while loop example:

```
using System;
/*
Diaplay all even number between 1 to 10
using do...while loop
*/

namespace Week2Programs
{
    0 references
    class doWhileDemo
    {
        0 references
        static void Main(string[] args)
        {
            // Variable declaration
            int controlVar = 1;
            Console.WriteLine("The even number between 1 and 10 are:");

            // do...while loop starts here
            do
            {
                if (controlVar % 2 == 0)
                {
                    Console.Write(controlVar + ",");
                }

                // IMPORTANT: increment the loop control variable
                // to avoid creating an infinite loop!
                controlVar++;
            } while (controlVar <= 10);

            Console.Read();
        }
    }
}
```

Output

```
The even number between 1 and 10 are:
2,4,6,8,10,
```

Reference: https://www.tutorialspoint.com/csharp/csharp_loops.htm

Loops in C#

4. Nested loops:

- Loop inside another loop
- Applicable to all type of loop

```
/*
Write a program to create a number triangle, e.g:
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
*/
```

Output

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

Example:

```
using System;
/*  ...

namespace Week2Programs
{
    0 references
    class NestedLoopDemo
    {
        0 references
        static void Main(string[] args)
        {
            //Initialize the outer loop
            for(int outerLoop=1; outerLoop<=5; outerLoop++)
            {
                // Outer loop body
                // Initialize the inner loop
                for(int innerLoop=1; innerLoop<=outerLoop; innerLoop++)
                {
                    // Inner loop body
                    // Print the numbers and a space
                    Console.Write(innerLoop + " ");
                } // Inner loop end
                // Transfer the conrol to next line for printing.
                Console.WriteLine();
            }// Outer loop end

            Console.Read();
        }
    }
}
```

Reference: https://www.tutorialspoint.com/csharp/csharp_loops.htm

Loops controls in C#

Loop control statements are used to alter the normal execution sequence of any loop

Statement	Description
break	Terminates the loop execution and transfers the control to the statement immediately after the loop
continue	Causes the loop to skip the remaining statements and continue the loop execution

Reference: https://www.tutorialspoint.com/csharp/csharp_loops.htm

Loops controls in C#

Loop control statements example:

```
/*
Write a program to print all
number between 1 to 10
except 3, 4 and 9
*/
```

```
using System;
/* ...

namespace Week2Programs
{
    0 references
    class LoopControlDemo
    {
        0 references
        static void Main(string[] args)
        {
            Console.WriteLine("The numbers are:");
            // Initialize the for loop here
            for(int controlVar =1; controlVar <=10; controlVar++)
            {
                // Skipping numbers 3, 4, and 9 from displaying.
                if(controlVar==3 || controlVar==4 || controlVar == 9)
                {
                    continue;
                    // Continue skips the printing and
                    // resumes the loop execution from
                    // next iteration
                }
                // Display the numbers
                Console.Write(controlVar + ",");
            }

            Console.Read();
        }
    }
}
```

Output

```
The numbers are:
1,2,5,6,7,8,10,
```

Reference: https://www.tutorialspoint.com/csharp/csharp_loops.htm

Strings in C#

- Strings are reference types (but behave a little like value types)
- string keyword is used to declare string variable
- A string can be created by assigning a string literal to a String variable, and in other ways.
- Example:

```
string subjectName = ".Net Application Development";

string firstName = "Hello";
string lastName = "World";
```

Reference: https://www.tutorialspoint.com/csharp/csharp_loops.htm

Strings in C#

- Properties in String Class:

```
string subjectName = ".Net Application Development";

string firstName = "Hello";
string lastName = "World";
```

Properties	Description	Example
Length	Returns the length of the string	firstName.Lenght

Reference: https://www.tutorialspoint.com/csharp/csharp_strings.htm

Strings in C#

- Methods in String Class:

```
string subjectName = ".Net Application Development";

string firstName = "Hello";
string lastName = "World";
```

Method	Description	Syntax
Compare()	Compares two specified String objects	String.Compare(String1, String2) e.g. String.Compare(firstName, lastName)
Concat()	Concatenates two Strings	String.Concat(String1, String2) e.g. String.Concat(firstName, lastName)
Contains()	Checks whether a specified string is present within another string. Returns true/false	String1.Contains(String2) e.g. subjectName.Contains(“Application”)
ToLower(), ToUpper()	Returns the copy of the string in lowercase Retuns the copy of the string in uppercase	String1.ToLower() String1.ToUpper() e.g. firstName.ToUpper() : Output: HELLO lastName.ToLower(): Output: world

Reference: https://www.tutorialspoint.com/csharp/csharp_strings.htm

Strings in C#

- Example:

```
using System;
// Program illustrate String creation and operations

namespace Week2Programs
{
    0 references
    class StringDemo
    {
        0 references
        static void Main(string[] args)
        {
            // Create Strings
            string firstName = "Hello";
            string lastName = "World";

            // Demonstration of basic String operations
            //1. Find the lenght of a string
            Console.WriteLine("The lenght of {0} is {1}", firstName, firstName.Length);

            // 2. Compare two Strings
            if (string.Compare(firstName, lastName) == 0)
            {
                Console.WriteLine("{0} and {1} are equal", firstName, lastName);
            }
            else
            {
                Console.WriteLine("{0} and {1} are not equal", firstName, lastName);
            }

            // 3. Concat two string
            string fullName = string.Concat(firstName, lastName);
            Console.WriteLine("The string after concatenation is {0}", fullName);

            // 4. Printing full name in upper case
            Console.WriteLine("The upper case version is {0}", fullName.ToUpper());

            // 5. Check if a string is present in another string
            if (fullName.Contains("World"))
            {
                Console.WriteLine("World is present in {0}", fullName);
            }
            Console.Read();
        }
    }
}
```

Output

```
The lenght of Hello is 5
Hello and World are not equal
The string after concatenation is HelloWorld
The upper case version is HELLOWORLD
World is present in HelloWorld
```

Reference: https://www.tutorialspoint.com/csharp/csharp_strings.htm

Enums in C#

- Enumerations are set of named integer constants
- **emun** keyword is used to declare an enumerated datatype
- **enums** are value types
- Syntax:

```
enum <enum_name> {
    enumeration_list
}
```

Example:

```
enum grade
{
    Fail,
    Pass,
    Credit,
    Distinction,
    HighDistinction
}
```

Reference: https://www.tutorialspoint.com/csharp/csharp_enums.htm

Enums in C#

- Enumerations Example

```
namespace Week2Programs
{
    0 references
    class EnumDemo
    {
        6 references
        enum Priority
        {
            Basic,
            Intermediate,
            High,
            Veryhigh
        }
        0 references
        static void Main(string[] args)
        {
            // Create an enum variable
            Priority PriorityValue = Priority.High;

            //Check for the enum values
            if(PriorityValue == Priority.Veryhigh)
            {
                Console.WriteLine("Solve the issue in 1 hours");
            }
            else if(PriorityValue == Priority.Intermediate)
            {
                Console.WriteLine("Solve the issue in 12 hours");
            }
            else if (PriorityValue == Priority.High)
            {
                Console.WriteLine("Solve the issue in 4 hours");
            }
            else if (PriorityValue == Priority.Basic)
            {
                Console.WriteLine("Solve the issue in 48 hours");
            }

            Console.Read();
        }
    }
}
```

Output

```
Solve the issue in 4 hours
```

Reference: https://www.tutorialspoint.com/csharp/csharp_enums.htm