BASICS OF C#

By:

Engr. Fatima Jaffar

Introduction

- C# (pronounced "C-sharp") is a modern, versatile, object-oriented programming language developed by **Microsoft** in **2000** that runs on the **.NET Framework**.

- C# is one of the top choices for developing
  - Windows applications
  - Unity game development
  - Enterprise solutions

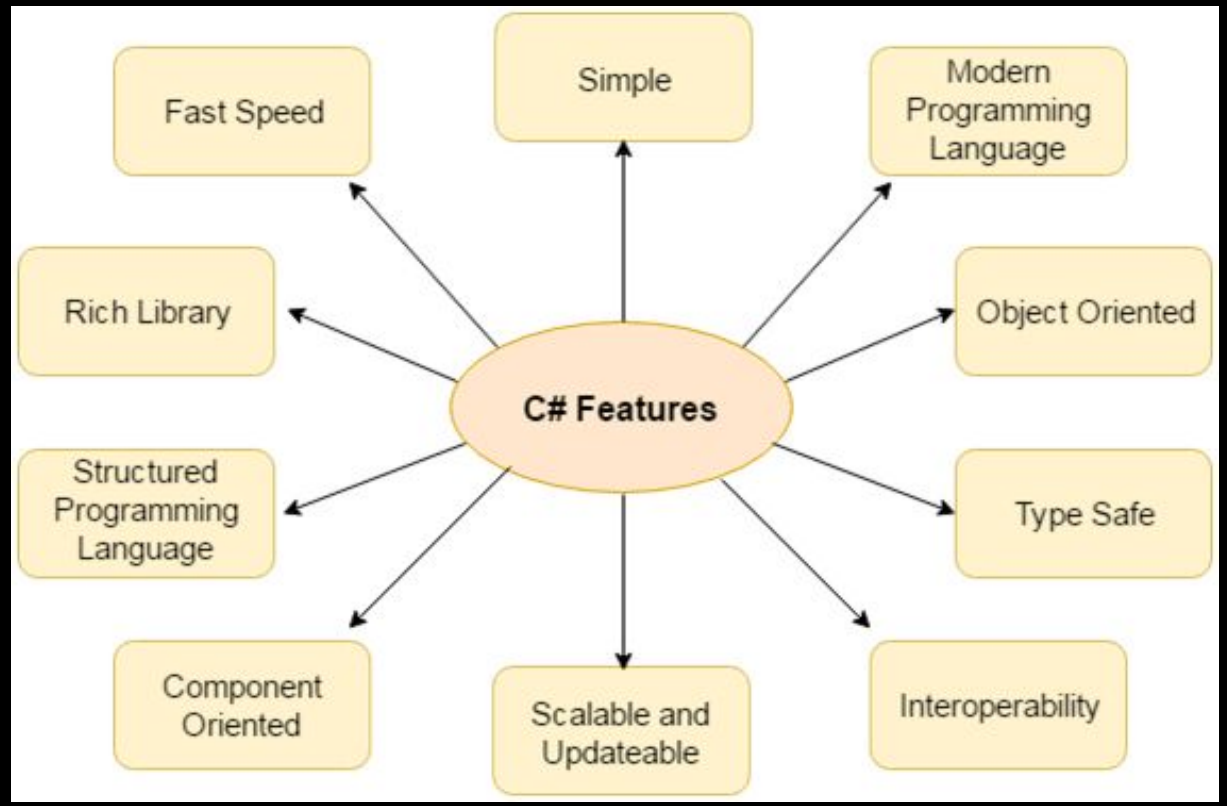
Introduction to C#

- C# originates from the C programming family and shares similarities with other widely-used languages like C++ and Java.

Introduction to C#

- C# is the primary language for developing games using the Unity engine.
- With .**NET Core**, C# applications can run on Windows, macOS, and Linux.
- The latest version, **C# 13** was released in November 2024 alongside **.NET 9**.

C# Features


C# Features
Fast Speed
Simple
Modern Programming Language
Rich Library
Object Oriented
Structured Programming Language
Type Safe
Component Oriented
Scalable and Updateable
Interoperability

Hello World Program in C#

```
class Program
  {
    static void Main(string[] args)
    {
      System.Console.WriteLine("Hello World!");
    }
  }
```

Using System

```
using System;
  class Program
  {
    static void Main(string[] args)
    {
      Console.WriteLine("Hello World!");
    }
  }
```

Using namespace

```
using System;
namespace ConsoleApplication1
{
   public class Program
   {
      public static void Main(string[] args)
      {
         Console.WriteLine("Hello World!");
      }
   }
}
```

C# Output

- To output values or print text in C#, you can use the **WriteLine()** method:

```
Console.WriteLine("Hello World!");
```

The Write Method

- There is also a Write() method, which is similar to WriteLine().
- The only difference is that it does not insert a new line at the end of the output:

```
Console.Write("Hello World! ");
Console.Write("I will print on the same line.");
```

C# Comments

- Single-line Comments:

  Single-line comments start with two forward slashes (//).

  Any text between // and the end of the line is ignored by C# (will not be executed).

```
// This is a comment
Console.WriteLine("Hello World!");
```

C# Comments

- Multi-line Comments:
- Multi-line comments start with /* and ends with */.
- Any text between /* and */ will be ignored by C#.

```
/* The code below will print the words Hello World
to the screen, and it is amazing */
Console.WriteLine("Hello World!");
```

C# Variables

- Variables are containers for storing data values.
- In C#, there are different types of variables (defined with different keywords), for example:

- **int** - stores integers (whole numbers), without decimals, such as 123 or -123
- **double** - stores floating point numbers, with decimals, such as 19.99 or -19.99
- **char** - stores single characters, such as 'a' or 'B'. Char values are surrounded by single quotes
- **string** - stores text, such as "Hello World". String values are surrounded by double quotes
- **bool** - stores values with two states: true or false

Declaring (Creating) Variables

- To create a variable, you must specify the type and assign it a value:

Syntax:

```
type variableName = value;
```

Example:

```
string name = "Abc";
Console.WriteLine(name);
```

Declare Many Variables

- To declare more than one variable of the same type, use a comma-separated list:
- Example:

```
int x = 5, y = 6, z = 50;
Console.WriteLine(x + y + z);
```

C# Identifiers

- All C# variables must be identified with unique names.
- These unique names are called identifiers.

Rules for naming variables

- The general rules for naming variables are:

  Names can contain letters, digits and the underscore character (_)

  Names must begin with a letter or underscore

  Names should start with a lowercase letter, and cannot contain whitespace

  Names are case-sensitive ("myVar" and "myvar" are different variables)

  Reserved words (like C# keywords, such as int or double) cannot be used as names

C# User Input

- Console.WriteLine() is used to print output.
- Console.ReadLine() is used to take input from user.

C# User Input

- Example:

```
// Create a string variable and get user input from the keyboard and store it in the variable
string userName = Console.ReadLine();
```

User Input and Numbers

- The Console.ReadLine() method returns a string. Therefore, you cannot get information from another data type, such as int. The following program will cause an error:

```
Console.WriteLine("Enter your age:");
int age = Console.ReadLine();
Console.WriteLine("Your age is: " + age);
```

- The error message will be something like this:

```
Cannot implicitly convert type 'string' to 'int'
```

- you can convert any type explicitly, by using one of the Convert.To methods:

```
Console.WriteLine("Enter your age:");
int age = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("Your age is: " + age);
```

C# Operators

- Operators are used to perform operations on variables and values.

- In the example below, we use the + operator to add together two values:

```
int x = 100 + 50;
```

Arithmetic Operators

Operator	Name	Description	Example
+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y
++	Increment	Increases the value of a variable by 1	x++
--	Decrement	Decreases the value of a variable by 1	x--

Assignment Operators

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

Comparison operators

Operator	Name	Example
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Logical Operators

Operator	Name	Description	Example
&&	Logical and	Returns True if both statements are true	x < 5 && x < 10
||	Logical or	Returns True if one of the statements is true	x < 5 || x < 4
!	Logical not	Reverse the result, returns False if the result is true	!(x < 5 && x < 10)

C# Math

- The C# Math class has many methods that allows you to perform mathematical tasks on numbers.

  Math.Max(x,y)

  Math.Min(x,y)

  Math.Sqrt(x)

  Math.Abs(x)

  Math.Round()