C# and DOT NET Framework

1. **What is C#? Write Use of c#.**

DEFINITION :

C# (pronounced "C-sharp") is a popular programming language developed by Microsoft. It is used to create different kinds of applications, such as websites, mobile apps, games, and desktop software. C# is simple to learn and is based on the **object-oriented programming (OOP)** model, which makes code easier to manage and reuse.

C# is commonly used with the **.NET framework**, which provides tools and libraries to build applications efficiently.

Uses of C#:

1. **Building Websites:**
   - C# is used with technologies like **ASP.NET** to create dynamic and interactive websites.
2. **Developing Games:**
   - Many video games are built using C# with game engines like **Unity**.
3. **Creating Mobile Apps:**
   - You can use C# with **Xamarin** to create apps for Android, iOS, and Windows devices.
4. **Desktop Applications:**
   - C# is widely used for developing Windows desktop applications like tools and utilities.
5. **Database Applications:**
   - C# works with databases like SQL Server to create apps that store and manage data.
6. **Enterprise Software:**
   - Businesses use C# to develop large-scale applications, like inventory management or financial systems.
7. **Cloud-Based Applications:**
   - C# integrates with Microsoft Azure to build scalable, cloud-based solutions.

In Easy Terms:

C# is like a Swiss Army knife for developers. You can use it to make games, websites, mobile apps, and much more. It’s beginner-friendly and powerful, making it a great choice for all kinds of programming projects.

A Basic C# Program

```
using System; // Importing namespaces


class Program // Define a class

{

    static void Main(string[] args) // Main method (entry point of the program)

    {

        Console.WriteLine("Hello, World!"); // Print a message to the console

    }

}
```

2) What are Access Modifiers in C#?

Access modifiers in C# define the **visibility** and **accessibility** of classes, methods, variables, and other members in your code. They determine **who can use** or **see** the code elements, ensuring proper encapsulation and security in your program.

Types of Access Modifiers in C#:

1. **Public (`public`)**
   - Members with the `public` modifier can be accessed **from anywhere**, inside or outside the class, or even in other projects (if referenced).
   - **Use Case:** When you want a method or property to be accessible globally.
2. **Private (`private`)**

- Members with the `private` modifier can only be accessed **within the same class**. It is the **default** access modifier if none is specified.
- **Use Case:** When you want to hide implementation details and prevent outside access.

3. **Protected (`protected`)**

- Members with the `protected` modifier can be accessed **within the same class** and by **derived (child) classes**.
- **Use Case:** Useful in inheritance when child classes need access to parent class members.

3) . Write a Features of C#.

Features of C# (C-Sharp)

C# is a powerful and modern programming language developed by Microsoft. It has many features that make it popular and widely used. Here are some key features of C# explained in easy English:

---

1. Object-Oriented Programming (OOP)

- C# is based on **Object-Oriented Programming**, which means it organizes code into "objects" (like real-world things) that have properties (data) and methods (actions).
- **Key Concepts**:
  - **Classes**: Blueprints to create objects.
  - **Objects**: Real-world instances of classes.
  - **Inheritance**: Reusing code from other classes.

    - **Polymorphism**: Using the same method in different ways.
    - **Encapsulation**: Hiding unnecessary details and protecting data.

---

2. Cross-Platform

- C# can be used to develop applications that run on multiple platforms like **Windows**, **macOS**, and **Linux**. With the help of **.NET Core**, C# applications can run on any system, not just Windows.

---

3. Memory Management

- C# has **automatic memory management** through the **Garbage Collector**. It automatically frees up memory when objects are no longer used, so developers don’t need to manually manage memory, making development easier and safer.

---

4. Strong Typing

- C# is a **strongly-typed language**, which means every variable must be declared with a type (like `int` for integers, `string` for text). This helps prevent errors and makes code easier to understand.

4) Explain Operators in Details?

Operators in C# are symbols used to perform operations on variables and values. They allow us to do things like add, subtract, compare, or assign values. C# provides a wide range of operators to perform different types of operations.

1. Arithmetic Operators

These operators are used for performing mathematical calculations.

Operator	Operation	Example
`+`	Addition	`a + b`
`-`	Subtraction	`a - b`
`*`	Multiplication	`a * b`
`/`	Division	`a / b`
`%`	Modulus (Remainder)	`a % b` (gives remainder of division)

2. Assignment Operators

These operators are used to assign values to variables.

Operator	Operation	Example
`=`	Assignment	`a = b`
`+=`	Add and assign	`a += b`
`-=`	Subtract and assign	`a -= b`
`*=`	Multiply and assign	`a *= b`
`/=`	Divide and assign	`a /= b`
`%=`	Modulus and assign	`a %= b`

3. Comparison (Relational) Operators

These operators are used to compare two values and return a `bool` result (true or false).

Operator	Operation	Example
==	Equal to	a == b
!=	Not equal to	a != b
>	Greater than	a > b
<	Less than	a < b
>=	Greater than or equal to	a >= b
<=	Less than or equal to	a <= b

4. Logical Operators

These operators are used to combine or invert Boolean values (true/false).

Operator	Operation	Example
&&	AND (both must be true)	a && b
`		`
!	NOT (inverts the value)	!a

5. Increment and Decrement Operators

These operators are used to increase or decrease a variable’s value by 1.

Operator	Operation	Example
++	Increment	a++ or ++a
--	Decrement	a-- or --a

6. Conditional (Ternary) Operator

This is a shorthand way to perform a simple `if-else` condition.

Operator	Operation	Example
?:	Conditional expression	condition ? expr1 : expr2

Summary:

- **Arithmetic operators** are used for math operations like addition, subtraction, multiplication, etc.
- **Assignment operators** are used to assign values to variables.
- **Comparison operators** compare values and return a boolean result (true or false).
- **Logical operators** are used to combine multiple conditions.
- **Increment and Decrement operators** are used to increase or decrease values by 1.
- **Bitwise operators** operate on individual bits of data.
- **Conditional operator** is a shorthand for `if-else`.
- **Null-coalescing operator** provides a default value when a variable is null.
- **Type-casting operators** convert one data type into another.

5) Explain Looping Statement in C# ?

Looping statements in C# are used to repeat a block of code multiple times until a specific condition is met. They allow you to avoid writing repetitive code. There are several types of loops in C# to handle different situations. Let's explore the most commonly used looping statements.

1. `for` Loop

The `for` loop is used when you know in advance how many times you want to repeat a block of code.

Syntax:

```
csharp
Copy code
for (initialization; condition; increment/decrement)
{
    // Code to be executed
}
```

- **Initialization:** Set a starting point.
- **Condition:** The loop will continue running as long as this condition is true.
- **Increment/Decrement:** This changes the loop variable after each iteration.

Example:

```
for (int i = 1; i <= 5; i++)
{
    Console.WriteLine(i);  // Prints 1, 2, 3, 4, 5
}
```

2. `while` Loop

The `while` loop repeats the block of code as long as the specified condition is **true**. The condition is checked **before** each iteration.

Syntax:

```
while (condition)
{
    // Code to be executed
}
```

- **Condition:** The loop will continue as long as this condition is true.

What are Looping Statements in C#?

Looping statements in C# are used to repeat a block of code multiple times until a specific condition is met. They allow you to avoid writing repetitive code. There are several types of loops in C# to handle different situations. Let's explore the most commonly used looping statements.

---

1. `for` Loop

The `for` loop is used when you know in advance how many times you want to repeat a block of code.

Syntax:

```
csharp
Copy code
for (initialization; condition; increment/decrement)
{
    // Code to be executed
}
```

- **Initialization:** Set a starting point.
- **Condition:** The loop will continue running as long as this condition is true.
- **Increment/Decrement:** This changes the loop variable after each iteration.

Example:

```
csharp
Copy code
for (int i = 1; i <= 5; i++)
{
    Console.WriteLine(i);  // Prints 1, 2, 3, 4, 5
}
```

In this example:

- The loop starts with `i = 1`.
- It continues as long as `i <= 5`.
- After each loop, `i` is increased by 1 (`i++`).

---

2. `while` Loop

The `while` loop repeats the block of code as long as the specified condition is **true**. The condition is checked **before** each iteration.

Syntax:

```
csharp
Copy code
while (condition)
{
    // Code to be executed
}
```

- **Condition:** The loop will continue as long as this condition is true.

Example:

```
csharp
Copy code
int i = 1;
while (i <= 5)
{
    Console.WriteLine(i);  // Prints 1, 2, 3, 4, 5
    i++;  // Increase i by 1 after each loop
}
```

In this example:

- The loop starts with `i = 1`.
- It checks if `i <= 5`. If true, it runs the block of code and increases `i` by 1 after each iteration.

---

3. `do-while` Loop

The `do-while` loop is similar to the `while` loop, but the condition is checked **after** the block of code is executed. This means the code will always run at least once, even if the condition is false.

Syntax:

```
csharp
Copy code
do
{
    // Code to be executed
} while (condition);
```

- **Condition:** The loop will continue as long as the condition is true. The condition is checked **after** the first iteration.