C# Programming Language Part1

1. Basic Structure of a C# Program

A **C# program** follows a specific structure, consisting of:

- **Namespaces** – Used to organize code and avoid name conflicts.
- **Classes** – A **blueprint** for objects, defining properties and methods.
- **Methods** – Blocks of code that perform specific tasks.
- **Statements** – Instructions executed by the program.

```
using System;  // Importing built-in namespace

namespace MyFirstApp  // Namespace declaration

{

  class Program  // Class declaration

  {

    static void Main()  // Main method (Entry point of the program)

    {

      Console.WriteLine("Hello, World!");  // Print statement

    }

  }

}
```

---

2. Comments in C#

Comments help to document the code and make it more understandable.
C# supports **three types of comments**:

1. Single-Line Comment
Used for short explanations.

```
// This is a single-line comment
Console.WriteLine("Hello, World!");  // This prints text to the console
```

2. Multi-Line Comment
Used for **longer explanations**.

```
/*
 This is a multi-line comment.
 It can span multiple lines.
*/
Console.WriteLine("Multi-line comment example");
```

3. XML Documentation Comments

Used for generating **documentation**.

```
/// <summary>
/// This method prints a greeting message.
/// </summary>
static void Greet()
{
   Console.WriteLine("Hello!");
}
```

---

3. Data Types & Variables

A variable is a container that holds a value, each variable has a data type that defines what kind of data it can store.

1. Value Types

These store actual values in memory.

Data Type	Size	Example
int	4 bytes	int age = 25;
double	8 bytes	double pi = 3.14159;
float	4 bytes	float price = 99.99f;
char	2 bytes	char letter = 'A';
bool	1 byte	bool isPassed = true;

2. Reference Types

These store memory addresses instead of actual values.

Data Type	Description	Example
string	Sequence of characters	string name = "John";
object	Base type for all types	object obj = 42;

4. Constants & Read-Only Variables

1. Constants (const)

- Constants **cannot be changed** after assignment.

```
const double Pi = 3.14159;
Console.WriteLine("Value of Pi: " + Pi);
```

2. Read-Only Variables (readonly)

- Can be assigned **only inside a constructor**.

```
class Example
{
  readonly int year;

  public Example(int y)
  {
    year = y;  // Assigned in constructor
  }

  public void Display()
  {
    Console.WriteLine("Year: " + year);
  }
}
```

---

5. Input & Output

1. Printing Output (Console.WriteLine)

```
Console.WriteLine("Hello, World!");
Console.Write("Enter your name: ");
```

2. Taking User Input (Console.ReadLine)

```
Console.Write("Enter your name: ");
string name = Console.ReadLine();
Console.WriteLine("Hello, " + name);
```

3. Converting Input (Convert.ToInt32)

```
Console.Write("Enter your age: ");
int age = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("You are " + age + " years old.");
```

6. Operators

1. Arithmetic Operators

Operator	Example	Result
+	a + b	Addition
-	a - b	Subtraction
*	a * b	Multiplication
/	a / b	Division
%	a % b	Modulus (Remainder)

```
int a = 10, b = 5;
Console.WriteLine(a + b);  // Output: 15
```

2. Logical Operators

Operator	Example	Description
&&	a > 0 && b > 0	AND
`		`
!	!(a > b)	NOT

```
bool result = (10 > 5) && (5 < 3);  // false
```

---

7. Conditional Statements

1. if-else Statement

```
int num = 10;
if (num > 0)
{
   Console.WriteLine("Positive Number");
}
else if ( num == 0 )
{
   Console.WriteLine("Zero");
}
else
{
   Console.WriteLine("Negative Number");
}
```

2. switch-case Statement

```
char grade = 'B';
switch (grade)
{
   case 'A':
      Console.WriteLine("Excellent");
      break;
   case 'B':
      Console.WriteLine("Good");
      break;
   default:
      Console.WriteLine("Invalid Grade");
      break;
}
```

---

8. **Loops :** Loops **repeat a block of code** multiple times.

1. For Loop

```
for (int i = 1; i <= 5; i++)
{
   Console.WriteLine(i);
}
```

2. While Loop

```
int i = 1;
while (i <= 5)
{
   Console.WriteLine(i);    i++;
}
```

3. Do-While Loop

```
int i = 1;
do
{
   Console.WriteLine(i);     i++;
} while (i <= 5);
```

4. Foreach Loop (for arrays & collections)

```
string[] fruits = { "Apple", "Banana", "Cherry" };
foreach (string fruit in fruits)
{
   Console.WriteLine(fruit);
}
```