

C#

Software - IDE visual studio not an open source code editor bcoz IDE has more feature than open source code editor

\* .NetFramework : It is the original implementation of .NET. It supports running websites, services, desktop apps and more on windows.

\* .NETcore : It is a cross-platform implementation for running websites, services and console apps on windows, linux and macos. .NETcore is open source on "Github".

\* Architecture of .NET Framework :-

The two major components of .NET Framework are the common language platform Runtime and the .NET Framework class library.

• The Common Language Runtime (CLR) : It is the execution engine that handles running applications. It provides services like thread management, garbage collection, type safety, exception handling and more.

• The class library provides a set of APIs and types for common functionality. It provides types for strings, dates, numbers etc. The class library includes APIs (Application programming interface) for reading and writing files, connecting to databases



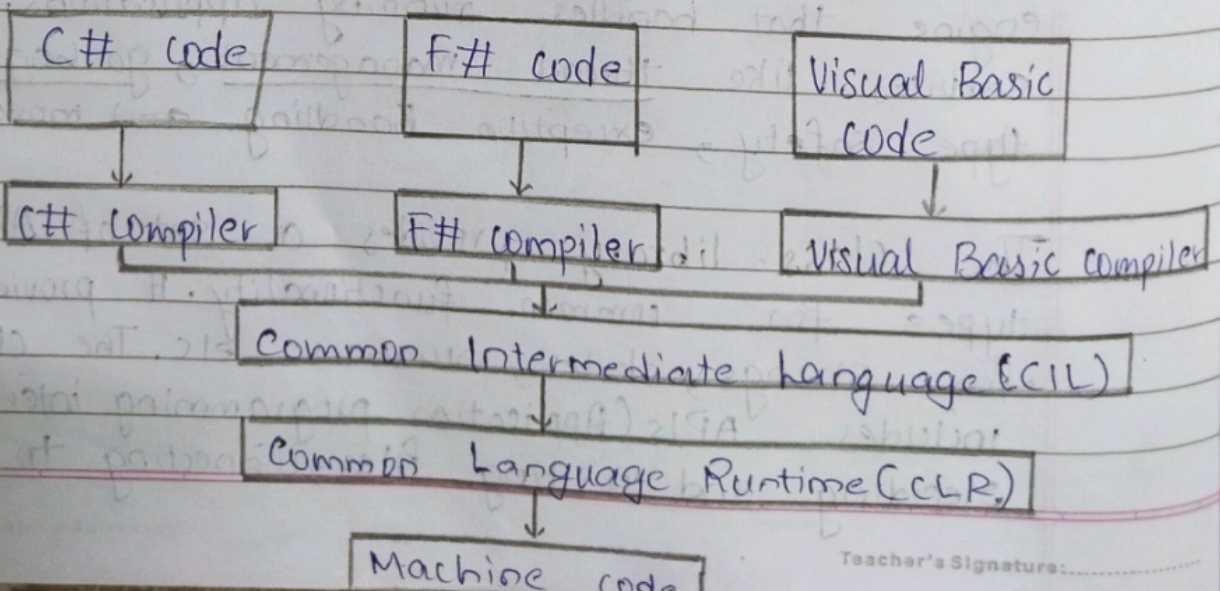
drawing and more.

• NET applications are written in the C#, F# or Visual Basic programming language. Code is compiled into a language-agnostic common intermediate language (CIL). Compiled code is stored in assemblies - files with a .dll or .exe file extension.

When an app runs, the CLR takes the assembly and uses a just-in-time compiler (JIT) to turn it into machine code that can execute on the specific architecture of the computer it is running on.

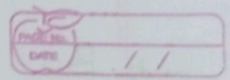
\* Thread management - Provides parallel execution.

\* Garbage collection (GC) - It is a memory management feature in C# that automatically frees up memory for objects that are no longer needed by a program.





C# is a type safe language.



## \* Application of C#:

Website development  
Desktop application  
Database application  
Console application  
WPF

\* Code: "Hello world" = print

a) Input code: In void main.

\*) `Console.WriteLine("Hello world");` ] — should leave line  
`Console.WriteLine` uses new line every line

\*\*) `Console.Write("Hello world");` ] — not leave line  
Output will be displayed in one line.

\*\*\*) `Console.WriteLine("Hello world");` ] — should leave new line

b) To hold the output screen: `Console.ReadLine();`  
↓  
Hold output screens.

N.T \*) Comments: It is used to give comments or instruction to in code, which does not execute.

\*) Single line comment: `(//)`

\*\*) Multiple line comment `(/* ... */)`

d) Variables:

```
int a = 34;  
Console.WriteLine("My lucky number" + a);  
Console.ReadLine();
```



1 bit = 8 bytes

→ By default decimal no. is "double".

Output: My Lucky number 34

\* C# datatypes :-

|               |                               |                                          |
|---------------|-------------------------------|------------------------------------------|
| viii] Long    | % long a;                     | → 8 bytes                                |
| ix] Integer   | % int a;                      | → 4 bytes                                |
| x] Floating   | % float a;                    | → 4 bytes                                |
| xi] Character | % char a = 'A';               | → 2 bytes                                |
| xii] Boolean  | % bool %great = true;         | provide only true or false value (1 bit) |
| xiii] String  | % String inp = "Hello world"; | → 2 bytes per character.                 |
| xiv] Double   | % double a;                   | → 8 bytes                                |
| * String:     | (15 float)                    |                                          |

Syntax Example: String inp = Console.ReadLine();

Console.WriteLine(inp);

Console.ReadLine();

Output Screen:

First it will be blank and as the user provides input in output screen it repeat that in string input in new line.

E.g: Hello World - Input by user in output screen

Hello World - output

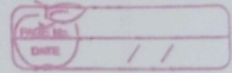
\* Example datatypes :-

\* Type casting :-

i] int a = 34; → if not written, shows error  
float b = 34.4; → and directly converts to double.  
double c = 34.0; → if not written, nothing issue  
Console.WriteLine(a);  
Console.WriteLine(b);  
Console.WriteLine(c);  
Console.ReadLine();



→ Ctrl + Alt - comment out all  
visual studio shortcut.



99. 

```
int a = 34;
float b = 34.4F;
double c = 34.4D;
boolean isGreat = true;
char d = 'r';

Console.WriteLine(a);
Console.WriteLine(b);
Console.WriteLine(c);
Console.WriteLine(isGreat);
Console.WriteLine(d);
Console.ReadLine();
```

Output:

34

34.4

34.4

True

r

\* Type casting: It is process in which the value (variable) is converted from one datatype to another datatype.

E.g:

```
int a = (int) 3.5;
Console.WriteLine(a);
Console.ReadLine();
```

Types of Type casting:

1] Implicit casting

2] Explicit casting



→ C + D - To replicate line to new line.



9) Implicit casting % Type casting done by compiler

char to int to long to float to double

E.g: a) `int x = 3;`  
`double y = x;`  
`console.WriteLine(x);`  
`console.WriteLine(y);`  
`console.ReadLine();`

b) `int x = 3;`  
`double y = x;`  
`int z = y;`  
`console.WriteLine(x);`  
`console.WriteLine(y);`  
`console.WriteLine(z);`  
`console.ReadLine();`

This is not possible. bcoz the promotion from int to double is possible but from "double to int" is not possible.

c) `int x = 3;`  
`double y = x;`  
`int z = 'y';`  
`console.WriteLine(x);`  
`console.WriteLine(y);`  
`console.WriteLine(z);`  
`console.ReadLine();`

Int. to double ✓  
char to double ✓

Output:

3  
3

121 - ASCII code of Y.

used to convert character to binary code.

Teacher's Signature: \_\_\_\_\_



99) Explicit Casting :- Type casting manual from user side

```
int x = (int) 3.5;
```

```
double x1 = (double) 3.5;
```

\* Extra Method. (Typecasting using Methods):

9) convert . To Double Itz Another way to convert

99) Convert . To String any variable from one

999) Convert . To Int64 datatype to another datatype.

E.g:- int x = 3;

double y = 4;

float z = 'y';

```
float yvarr = Convert.ToInt32(3.55);
```

```
console.WriteLine(x);
```

```
console.WriteLine(y);
```

```
console.WriteLine(z);
```

```
console.WriteLine(yvarr);
```

```
console.ReadLine();
```

\* To get user input :

E.x :-

```
1) console.WriteLine("Enter your name");
```

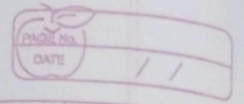
```
String name = console.ReadLine();
```

```
console.WriteLine("Hey hello" + name);
```

```
console.ReadLine();
```



If an operation first is string then all remaining operations will be converted into string.



Q. 2. How many candies do you want? write Typecasting code using METHODS.

→ ~~int~~ ~~st~~

```
String can = Console.ReadLine();
```

```
Console.WriteLine("How many candies do you want");
```

```
Console.WriteLine("You will get 4 more candies:" + (Convert.ToInt32(can) + 4));
```

```
Console.ReadLine();
```

First Priority during compilation.

Output: How many candies do you want  
4

You will get 4 more candies: 8.

(OR)

Program without using Typecasting.

```
int a = console.ReadLine();
```

```
console.Write("How many candies do you want");
```

```
console.Write("You will get 4 more:" + (a + 4));
```

```
console.ReadLine();
```



## \* Operators %

- i. Arithmetic operators - (+, -, \*, /, %)
- ii. Assignment operators - Assign no. to another no. (=)
- iii. Comparison operators - (<, <=, >, >=, ==, !=)
- iv. Logical operators - (&, OR (|), NOT (!))

→ works on two boolean operators.

Examples %

```
1) int a = 3;  
   int b = 4;  
   Console.WriteLine("The value of a+b is:" +  
   (a + b));  
   Console.ReadLine();
```

ii) Assignment operators:

```
int a = 3;  
int b = a;  
b += 4;  
Console.WriteLine(b);  
Console.ReadLine();
```

iii) Logical operators.

a) AND (&) - Output will true when both are true.

```
Console.WriteLine(true & false); - false  
Console.WriteLine(true & true); - True  
Console.WriteLine(false & false); - False  
Console.ReadLine();
```



b. OR (||): If any 1 is true then O.P. will be true

console.WriteLine(true || false); - True

console.WriteLine(true || true); - True

console.WriteLine(false || false); - False

console.ReadLine();

c NOT (!): Re Invert the input.

console.WriteLine(!False); - True

console.WriteLine(!True); - False

console.ReadLine();

Q] Comparison Operators & (>, <, >=, <=, !=, ==)

a) console.WriteLine(324 > 5);

console.ReadLine();

Output Screen  
True

b) console.WriteLine(324 > 555);

console.ReadLine();

False

\* Math class in C#:

a) int a = Math.Max(34, 345);

console.WriteLine(a);

console.ReadLine();

345

b) int a = Math.Min(34, 345);

console.WriteLine(a);

console.ReadLine();

34



c. double a = Math.sqrt(36);  
 console.WriteLine(a);  
 console.ReadLine();

6

d. int a = Math.Abs(-39);  
 console.WriteLine(a);  
 console.ReadLine();

39

- Absolute value.

## \* String Functions

String a = "Hello world";

- a) console.WriteLine(a.Length);
- b) console.WriteLine(a.ToUpper());
- c) console.WriteLine(a.ToLower());

"  
 HELLO WORLD  
 hello world

d) string concatenation: console.WriteLine(a + "You are nice");

(OR)

console.WriteLine(String.Concat(a, "You are nice"));

Hello world You are nice

IMP

## \* String interpolation

```
String name = Console.ReadLine();
String candies = Console.ReadLine();
console.WriteLine($"Your name is {name},
You will get {candies} candies");
console.ReadLine();
```



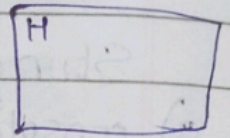
NOTE: Counting in C# starts from 0 and whenever the counting starts from i.e., known as indexing. 0 (zero) means first character.

0 - First character

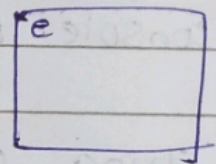
1 - Second character

2 - Third character

E.g: `String hello = "Hello World";`  
`console.WriteLine(hello(0));`  
`console.ReadLine();`



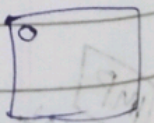
`String hello = "Hello world";`  
`console.WriteLine(hello(1));`  
`console.ReadLine();`



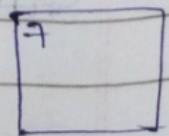
## \* Methods of String:-

### 1. IndexOf():

E.g: `String a = "Hello world";`  
`console.WriteLine(a.IndexOf("Hello"));`  
`console.ReadLine();`



`String a = "Hello is word";`  
`console.WriteLine(a.IndexOf("is"));`  
`console.ReadLine();`





## 2. Substring()

E.g a) String a = "Hello world";  
console.WriteLine(a.Substring(1));  
console.ReadLine();

ello world

b) String a = "Hello world";  
console.WriteLine(a.Substring(5));  
console.ReadLine();

world

\* Escape sequence character :-

→ Double quote in string: (\")%

String a = "Hello world \" is Harry";

Hello world " is Harry

→ New line in string: (\n)%

String a = "Hello world \n is Harry";

Hello world  
is Harry

→ It - Insert tab in string or add more space in string.

These 3 are Escape Sequence character.



## IMP \* Decision control :-

### 1) if else statements :-

E.x: 1. `int age = 56;`

```
if (age > 18)
```

```
{
```

```
    console.WriteLine("You can drive");
```

```
}
```

```
else
```

```
{
```

```
    console.WriteLine("You can not drive");
```

```
}
```

```
console.ReadLine();
```

E.x. 2:

```
String ageStr = console.ReadLine();
```

```
console.WriteLine("Enter your name");
```

```
int age = Convert.ToInt32(ageStr);
```

```
if (age < 10)
```

```
{
```

```
    console.WriteLine("Finish high school");
```

```
}
```

```
else if (age < 18)
```

```
{
```

```
    console.WriteLine("below 18 can't drive");
```

```
}
```

```
else if (age > 18)
```

```
{
```

```
    console.WriteLine("can drive");
```

```
}
```

```
console.ReadLine();
```



## ~~IMP~~ \* Loops %

### \* Switch Statement %

```
int age = 18;
```

```
switch (age)
```

```
{
```

```
    case 18:
```

```
        console.WriteLine("Please wait for an year");  
        break;
```

```
    case 20:
```

```
        console.WriteLine("You are 20");  
        break;
```

```
    default:
```

```
        console.WriteLine("Enjoy");  
        break;
```

```
} console.ReadLine();
```

## ~~IMP~~ \* Loops % in C#

If the Question is given that show 1 to 5 numbers in output-

```
console.WriteLine(1);
```

```
console.WriteLine(2);
```

```
    " (3);
```

```
    " (4);
```

```
    " (5);
```

```
console.ReadLine();
```



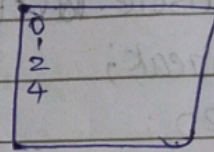
That's not a perfect standard to write a code

Better Alternative is "Loop": It is a better alternative repetitive statement.

1) While Loops: It runs only if the cond<sup>n</sup> is true.

a) `int i = 0;  
while (i < 5)`

`{  
 console.WriteLine(i);  
 i++;  
}`



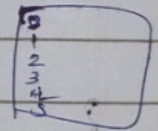
|   |
|---|
| 0 |
| 1 |
| 2 |
| 3 |
| 4 |

b) `int i = 0;  
while (i <= 5)`

`{  
 console.WriteLine(i);  
 i++;  
}`

(OR) `int i = 0;  
while (i < 5)`

`{  
 console.WriteLine(i+1);  
 i++;  
}`



|   |
|---|
| 0 |
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |

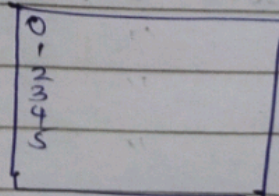
2) Do while Loops: It runs atleast once even though the condition is false.

`int i = 0;  
do {`

`console.WriteLine(i+1);  
 i++;`

`} while (i < 5);`

`console.ReadLine();`



|   |
|---|
| 0 |
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |



3) For Loops :-

```
for (i = 0 ; i < 5 ; i++)  
{  
    console.WriteLine(i+1);  
}  
console.ReadLine();
```

\* NOTE:

if break : It is used to exit loop.

eg: for (i = 0 ; i < 5 ; i++)

```
{  
    console.WriteLine(i+1);  
    break;  
}  
console.ReadLine();
```

(Skip the particular loop).

if continue : Skip that step and continue next next or upgraded one.

eg: for (i = 0 ; i < 5 ; i++)

```
{  
    if (i == 0)  
    {  
        console.WriteLine("Hello");  
        continue;  
    }  
    console.WriteLine(i+1);  
    console.ReadLine();  
}
```

|   |
|---|
| 2 |
| 3 |
| 4 |
| 5 |



\* Methods : Also known as function.

a) namespace Hello

```
{  
class program
```

```
{  
    static void Greet()
```

```
{  
        Console.WriteLine("Good morning");  
    }
```

```
static void main(String[] args)
```

```
{  
    Greet();
```

```
    Console.ReadLine();  
}
```

```
}
```

```
}
```

b. class program

```
{
```

```
static void Greet(String name)
```

```
{
```

```
    Console.WriteLine("Good Morning" + name);  
}
```

```
static void main(String[] args)
```

```
{
```

```
    Greet("Harry");
```

```
    Greet("Mohan");
```

```
    Console.ReadLine();  
}
```

```
}
```

|                    |
|--------------------|
| Good Morning Harry |
| Good Morning Mohan |



c) class Program

```

{
    static float Average(int a, int b, int c)
    {
        int Sum = a + b + c;
        return Sum / 3;
    }

    static void main(String[] args)
    {
        Console.WriteLine(Average(2, 3, 4));
        Console.ReadLine();
    }
}

```

d) Assigning variable:

```

class Program
{
    static float Average(int a, int b, int c)
    {
        int Sum = a + b + c;
        return Sum / 3;
    }

    static void main(String[] args)
    {
        float temp = Average(9, 3, 0);
        Console.WriteLine(temp);
        Console.ReadLine();
    }
}

```



# \* Object Oriented programmings (OOPs) %

classes and objects

```
namespace Hello
```

```
{  
    class player
```

```
{  
    public string name = "Harry";  
    private int health = 49;
```

```
    public int gethealth()  
    {
```

```
        return health;  
    }
```

```
    public void setHealth(int h)  
    {
```

```
        health = h;  
    }
```

```
}
```