GDSC – SHA
C# Programming

Session 2

{ Youssef Adel , Nour Elden Hany }

Today's Plan

- Welcome to C#
- System.Console.WriteLine() Statement
- Variables & Datatypes
- Casting & Type Conversion
- Operators
  (Unary , Arithmetic , Assignment, Relational , Logical)
- Commenting and disabling codes
- ControlFlow Statements
  (IF / Else , Switch case) (Loops -> While ,do while , For)
- Arrays (1D , 2D)

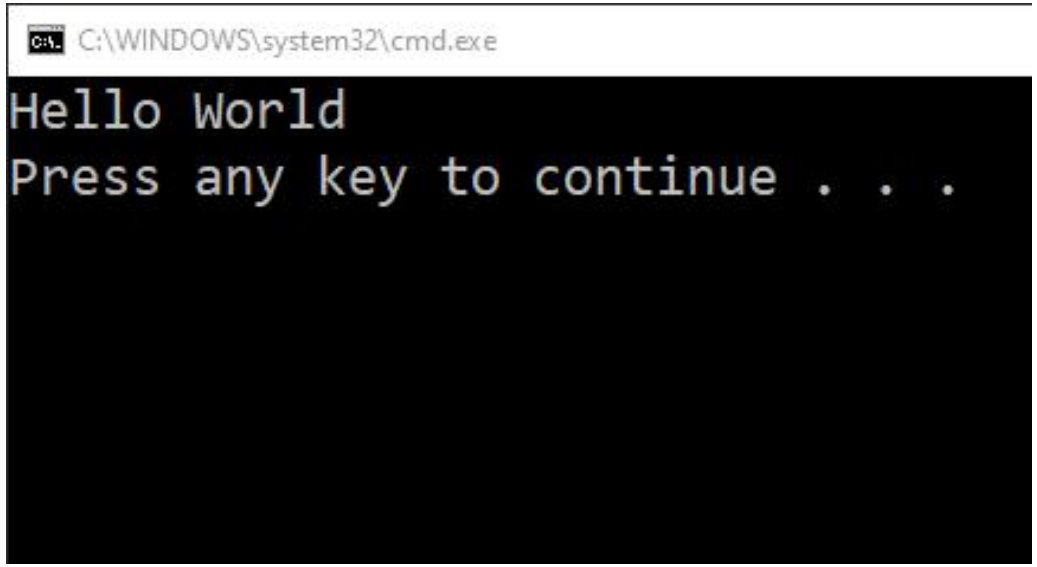
System.Console.Write()

System.Console.WriteLine()

If you want to write a text , You Should make it within “Doubleqoutes”

Like this : System.Console.WriteLine(“Hello World”);

The output will be : Hello World



By writing **Using System ,** **You can use the print statement without starting with System word.**

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

Console.Write("Hello World");
```

/n , /t , /’

Console.WriteLine(“Hello World**”) =** Console.Write(“Hello World/n”)

Console.WriteLine(“Hello World**”)**

Console.WriteLine(“Hello/tWorld**”)**

Variables & Datatypes

Short	2 bytes	short x = 5;
Integer	4 bytes	int x = 99;
Long long	8 bytes	long long x = 1823;
Float	4 bytes	float x = 12.5;
Double	8 bytes	Double x =12.56498;

Variables & Datatypes

Characters **char** Var = ‘a’;

String “Text” **string** Var = “Ali”;

Variables & Datatypes

Boolean :

```
bool x = true;
bool x = false;
```

Variables naming rules

Don’t start with numbers

Don’t start with symbols except (_ or $)

Don’t use spaces

Casting & Typeconversions

If you have an integer , How to save it or print it as a string ?

→by using a methods that convert it to your chosen type.

Any datatype to string → Use .ToString() or Convert.ToString()

Any datatype to int → Use int.Parse(**string**) or Convert.ToInt32()

We have int.Parse(**string**),char.Parse(**string**),double.Parse(**string**)

Casting & Typeconversions

Without Using methods :

double x = 5.6;

If I need to save x in another variable as an integer:

Int y = x ? **Error**

Int y = (**int**) x;

Operators

Operators in C#

	Operators	Type
Unary Operator →	++, --	Unery Operator
Binary Operator	+, -, *, /, %	Arithmetic Operator
Binary Operator	<, <=, >, >=, ==, !=	Relational Operator
Binary Operator	&&, ||, !	Logical Operator
Binary Operator	&, |, <<, >>, ~, ^	Bitwise Operator
Binary Operator	=, +=, -=, *=, /=, %=	Assignment Operator
Ternary Operator →	?:	Ternary or Conditional Operator

Operators

Unary : deal with on variable only .

Int X = 5;

X++; increment

Now X = 6;

X--; decrement

Now X = 5;

Operators

Assignment : += , -= , *= , /= ;

Used to assign update to the current variable value.

Int x = 5;

X += 1 is equal to x = x+1 or x++;

X -= 1 is equal to x = x-1 or x--;

X *= 5 is equal to x = x*5;

X /= 5 is equal to x = x/5;

Operators

Arithmetic : **+ , -, * , / , %**

```
Int x = 5 ; int y = 7;
Int z = x + y;
Int s = x - y
Int m = x * y;
Int d = x / y;
Int h = x % y;
```

Operators

Relational Op. : Returns boolean value;

<, <=, >, >=, ==, !=

```
Bool x = 5 < 79;
//x = True
Bool y = 8 == 96;
Console.WriteLine(y);
```

Output : False;

Operators

Logical : Used between expressions.

→ **&&** and

X > y && X > z

→ **||** or

X != 8 && X != 6

→**!** Not

Returns the opposite boolean value of expression;

Bool x = true ; Console.Write(!x); output = false;

Commenting and disabling codes

Double slash for on statement

Like : //int x = 5;

The compiler doesn't read it.

Slash star for blocks of code

/*

Code blocks

*/

Commenting and disabling codes

What is the output of this code ?

```
Int X = 5;
X*=2;
X--;
// X*=3;
Console.WriteLine(X);
```

Control Flow Statements (Selective Statements)

IF Statement

```
If( condition )
{
        If condition return true
}
```

Problem : Write an if statement that checks if a number is positive. If it is, print "Number is positive"

Control Flow Statements (Selective Statements)

Answer : if (number > 0) { Console.WriteLine("Number is positive."); }

Control Flow Statements (Selective Statements)

IF – else Statement

```
If( condition )
{
        If condition return true
}
Else
{
        If condition return false
}
```

Control Flow Statements (Selective Statements)

Problem: Write an if – else statement that checks if a number is positive or negative. If it is positive, print “Number is positive” else print “Negative”

Problem: Write an if-else statement that checks if a number is even. If it is, print “Number is even.”; otherwise, print 'Number is odd.'

Control Flow Statements (Selective Statements)

Switch case :

Problem: Write a switch statement that checks the value of a variable **‘day’** and prints the corresponding day of the week.

```
Switch ( var )
{
 case 1:
                    Console.WriteLine("Sunday");
                    break;
 case 2:
                    Console.WriteLine("Monday");
                    break;
case 3:
                    Console.WriteLine("Tuesday");
                    break;
case 4:
                    Console.WriteLine("Wednesday");
                     break;
case 5:
                    Console.WriteLine("Thursday");
                     break;
case 6:
                    Console.WriteLine("Friday");
                     break;
 case 7:
                    Console.WriteLine("Saturday");
                    break;
default:
                    Console.WriteLine("Invalid day");
                     break;
}
```

Control Flow Statements(Repetitive Statements)

Loops : **while , do while , For loop ,nested For loop**

Why we use loop ?

→ **To repeat some blocks of code.**

Control Flow Statements(Repetitive Statements)

While Loop;

```
While( Condition is true )
{
        // some code;
}
```

We can depend on a **condition** to repeat codes , or depend on a "**counter**"

To stop at a certain point.

Control Flow Statements(Repetitive Statements)

```
Int i = 1;                  //initialized variable ,start point
While ( i <= 3 )            //condition
{
 //some code;

i++;                        //update
}
```

Repeated 3 times;

Control Flow Statements(Repetitive Statements)

Problem: Write a while loop that prints the numbers from 1 to 5.

Problem: Write a while loop that prints the sum of numbers from 1 to n.

Control Flow Statements(Repetitive Statements)

What about Do While loop ?

```
do
{
//some code
}
While(Condition);
```

The **do scope** will execute **even condition is true or not**;

If condition is true , **we will return to the do scope again** .

Control Flow Statements(Repetitive Statements)

Problem: Write a do-while loop that prints the numbers from 1 to 5.

Control Flow Statements(Repetitive Statements)

For Loop :

```
for ( initialized variable ; condition ; update )
{
//Some code
}
```

Note : if you know your start and your end , its better to use for loop;

Control Flow Statements(Repetitive Statements)

Problem: Write a for loop that prints the numbers from 1 to n.

Problem: Write a for loop that prints the multiplication of numbers from 1 to n.

Control Flow Statements(Repetitive Statements)

While loop	Do- While	For loop

```
static void Main(string[] args)
{

    int i = 1;
    while (i < 5)
    {
        Console.WriteLine(i);//from 1 to 4
        i++;
    }



}
```

```
static void Main(string[] args)
{

    int i = 1;
    do
    {
        Console.WriteLine(i);//from 1 to 4
        i++;
    }
    while (i < 5);


}
```

```
static void Main(string[] args)
{


    for (int i = 1; i< 5; i++)
    {
        Console.WriteLine(i);//form 1 to 4
    }


}
```

All of them print numbers from 1 to 4 , Use what you like

Control Flow Statements(Repetitive Statements)

Nested For Loop :

```
for ( int  i = 1 ; i < 5 ; i++ )
{
        repeat this loop 4 times .
        for ( int j = 1 ; j < 5 ; j++ )
        {
        some code;
        }
}
```

Control Flow Statements(Repetitive Statements)

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

Trace & Guess the output !

Control Flow Statements(Repetitive Statements)

The output is → 11234

21234

31234

41234

Summary : we use nested for loop to repeat the integrated loop, *Nested for loops are used when we need to iterate over multiple dimensions or levels of data structures, such as nested arrays or matrices. They allow us to perform repetitive tasks for each combination of elements in the nested structures.*

Arrays (1D)

Definition: A 1D array is a collection of elements of the same data type arranged in a single row or column , Elements can be accessed by index .

Int x	Int y	Int z	Int s	Int k

Array of 5 integers

string x	string y	string z	string s	string k	String l

Array of 6 strings

Arrays (1D)

How to write an array ?

We have 3 steps.
First Declaration ,
→ Datatype [ ] array name;

Second Creation,
→Arrayname = new datatype [size] ;
Let size = 3 ;

0	0	0
Index [0]	Index [1]	Index [2]

Int [] Arr ;
Arr = new int [3];

Third Initialization (By adding elements) here we have a lot of methods to add elements
Like insertion by index ;
Arrname[0] = 5;
Arrname[1] = 3;
Arrname[2] = 7;
……

5	3	7
Index [0]	Index [1]	Index [2]

Arrays (1D)

We can initialize the array or add elements in creation step , Like:

Int [] arr { 5, 7 , 3 } ;

Now the compiler determines the size of array by counting the elements .

Arrays (1D)

Problem : make an array of size 5 ;

Then give 5 elements to it , by index and by initialize ;

Then print the third element ;

```
Console.WriteLine("Thank You");
```