

A collection of various electronic components is scattered across a light-colored surface. These include several resistors with different colored bands, two yellow electrolytic capacitors, a small black integrated circuit, a green printed circuit board (PCB) with multiple pins, and a grey rectangular component. A black rectangular box is overlaid on the right side of the image, containing the title and author information.

PROGRAMMING SKILLS II

~ANURUDDHA ABEYSINGHE

Lecture 03

Foundation Certification in IT – Curtin batch

C# LANGUAGE FUNDAMENTALS

LECTURE 03

OVERVIEW



Operators in C#



Strings Handling in C#

OPERATORS IN C#

- There are 04 Operator Types in C# Language.
 - Arithmetic Operators
 - Relational Operators
 - Logical Operators
 - Assignment Operators

RELATIONAL OPERATORS

- Relational Operators are useful to check the relation between two operands

Operator	Name	Description	Example (a = 6, b = 3)
==	Equal to	It compares two operands, and it returns true if both are the same.	a == b (false)
>	Greater than	It compares whether the left operand greater than the right operand or not and returns true if it is satisfied.	a > b (true)
<	Less than	It compares whether the left operand less than the right operand or not and returns true if it is satisfied.	a < b (false)
>=	Greater than or Equal to	It compares whether the left operand greater than or equal to the right operand or not and returns true if it is satisfied.	a >= b (true)
<=	Less than or Equal to	It compares whether the left operand less than or equal to the right operand or not and returns true if it is satisfied.	a <= b (false)
!=	Not Equal to	It checks whether two operand values equal or not and return true if values are not equal.	a != b (true)

LOGICAL OPERATORS

- Boolean expressions can also use the following *logical and conditional operators*:

!	Logical NOT
&	Logical AND
	Logical OR
^	Logical exclusive OR (XOR)
&&	Conditional AND
 	Conditional OR

- They all take boolean operands and produce boolean results

COMPARISON: LOGICAL AND CONDITIONAL OPERATORS

- Logical **AND (&)** and Logical **OR (|)**
 - Always evaluate both conditions
- Conditional AND (&&) and Conditional OR (||)
 - Would not evaluate the second condition if the result of the first condition would already decide the final outcome.
 - Ex 1: **(false) && (x++ > 10)** --- no need to evaluate the 2nd condition because first condition gets False. Due to use &&, it not wants to check second condition.
 - Ex 2:

```
if (count != 0 && total /count)
{
    ...
}
```

ASSIGNMENT OPERATIONS

- You can consider assignment as another operator, with a lower precedence than the arithmetic operators

```
answer = sum / 4 + MAX * lowest;
```

4 1 3 2



Then the result is stored in the variable on the left hand side

- The right- and left-hand sides of an assignment statement can contain the same variable

First, one is added to the original value of count

```
count = count + 1;
```



Then the result is stored back into count (overwriting the original value)

ASSIGNMENT OPERATORS

Assignment operator	Sample expression	Explanation
+=	c += 7	c = c + 7
-=	d -= 4	d = d - 4
*=	e *= 5	e = e * 5
/=	f /= 3	f = f / 3
%=	g %= 2	g = g % 2

INCREMENT AND DECREMENT OPERATORS

Operator	Called	Sample expression	Explanation
++	preincrement	++a	Increment a by 1, then use the new value of a in the expression in which a resides.
++	postincrement	a++	Use the current value of a in the expression in which a resides, then increment a by 1.
--	predecrement	--b	Decrement b by 1, then use the new value of b in the expression in which b resides.
--	postdecrement	b--	Use the current value of b in the expression in which b resides, then decrement b by 1.
The increment and decrement operators.			



```
using System;
```

```
class Increment
```

```
{
```

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

```
{
```

```
    int c;
```

```
    c = 5;
```

```
    Console.WriteLine( c ); // print 5
```

```
    Console.WriteLine( c++ ); // print 5 then postincrement
```

```
    Console.WriteLine( c ); // print 6
```

```
    Console.WriteLine(); // skip a line
```

```
    c = 5;
```

```
    Console.WriteLine( c ); // print 5
```

```
    Console.WriteLine( ++c ); // preincrement then print 6
```

```
    Console.WriteLine( c ); // print 6
```

```
} // end of method Main
```

```
}
```

STRING HANDLING IN C#

1. Manage Strings in Output - Concatenation

Example 1 –

```
string part1 = "SLIIT";  
string part2 = " Academy";  
string part3 = part1 + part2;  
Console.WriteLine(part3);
```

SLIIT Academy

Example 2 –

```
string part1 = "SLIIT";  
string part2 = " Academy";  
string part3 = string.Concat(part1,part2);  
Console.WriteLine(part3);
```

SLIIT Academy

Example 3 –

```
string x = "1";  
string y = " 2";  
string part3 = part1 + part2;  
Console.WriteLine(part3);
```

12

STRING HANDLING IN C#

2. Manage Strings in Output – Special Characters

Example 1 –

```
Console.WriteLine("using \" in Output");
```

Using “ in output

Example 2 –

```
Console.WriteLine("using \' in Output");
```

Using ‘ in output

Example 3 –

```
Console.WriteLine("using \\ in Output");
```

Using \ in output

Example 4 –

```
Console.WriteLine("using \n new line character");
```

Using
new line character

Example 5 –

```
Console.WriteLine("using \t tab line character");
```

Using tab line character

LET'S SUMMARIZE



Operators in C#



String Handling in C#

THANK YOU

SEE YOU NEXT
WEEK