**Session 3: Operators, Type Conversion: implicit & Explicit OR Boxing Unboxing**

**Sortcut for Comment :- Ctrl + k + c Uncomment : - Ctrl + k + u**

**Operators:** Operator it’s a symbol used to perform some operations. Its foundation of any pg Language. Operator is used with operand (int c(Opernad) =(Operator) 0). C# has built in operators.

Eg:- int a + b ;

**C# oprators can be categorised based on no of Operand:**

**Unary Operator:** Operator that takes one operand to perform the operation.

Increament Operator ++ Decrement Operator – -

EG:- a++(Post Fix) : (a=a+1);

++a(Prefix) : (a=1, b= ++a >>> a=2 then assigned to b)

**Binary Operator:** Operator that takes two operands to perform the operation.

**Ternary Operator:** Operator that takes **three** operands to perform the operation. ? :

* **Types of Operator: Arithmatic Operator:** - used to Perform Mathematical Calcualtion. (+,- ,\* ,% , /, --(Decrement) ,++(Increment))

Eg:- a +b OR a\*b;

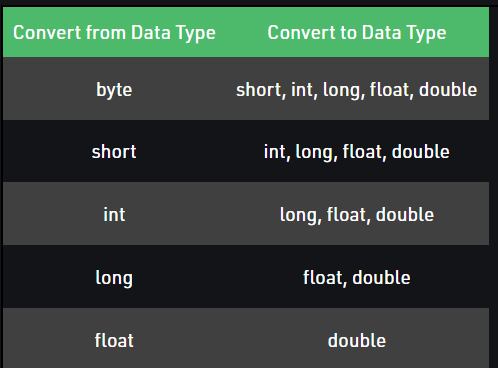
* **Relational Operator:-** use to display the relation or compare the value or check the condition.(==, != , < (LessThan), > (GreaterThan), <=, >=).

**EG: a=10; b=20 >>> a>b >>False**

* **Logical Operator:** used to combine two or more values. It checks the condition. (&& Logical And, || Logical OR, ! Logical Not)
  + **&& AND: -** if both operands are non-zero then condition will True. >> 0 1
    - **EG: int** a=5, b=5,c=5; (a==b && a==c) True
  + **| | OR: -** if anyone operand become zero then it will return True.
    - **EG: int** a=5, b=10,c=5; (a==b || a==c) True
  + **! Not: -** if condtion true then return false else return true.
    - **EG: : int** a=5, b=10,c=5; (!(a==b || a==c)) False
* **Assignment Operator: -** it’s used to assign values. Left side is operand and right side is value.
  + **=**
  + **(Arithmatic Assignment ) += :-** it add value first to the left. and then assign the result to the right. And then assign the result to left.
    - * EG :- a+=b; a=a+b >> b=1; a =0 ; >> a=1+2
* **Miscellaneous Operators :** is used to as per the requirement.
  + **sizeof() :-** return the data type size.
  + **typeof() :-** return the type of class or variable.
  + **\* Pointer :-** point to a variable.
  + **& Address of :-** return variable address.
  + **?: Ternary or Conditional Operator :** determines if any condition is true? If yes, then first value: otherwise second value.
  + **Is** "is" finds out if an object is of a specific type or not.

**Type Conversion:** - converting data type or typecasting variable in another data type.

* **Implicit Conversion:** - when the two data types are compatible. When we assign small data type to bigger. EG:- numeric data types are compatible with each other but no auto conversion is supported. It can identify the exception.



* **Explicit Conversion: -** manually converting data type value to another data type. When we assign large data type to small .we can called it cast.

EG:- double >> float >> long >> int >> char

**Boxing:-** converting value type to object or to any interface type implemented by value type. When CLR boxes value type it stored the value in System.Object instance. Its implicit. You can do it explicitly too. No data loss. No possibility of throwing exception during conversion.

EG:- string name = “Imran” (VlueTp) >> object o =(object)name; (Bxng)

EG:- string name = “Imran”; >> object o = name; >> name= “Abhilasha”

**Unboxing:-**  its explicit conversion from type object to value type. It consist:- checked value is boxed. Copying value from instance into value type variable.

EG:- string name = “Imran” (VlueTp) >> object o =(object)name; (Bxng), string lName = (string) name (unbxng).

**Program**

S3\_OperatorAndTypeConversion.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace AllSession

{

class S3\_OperatorAndTypeConversion

{

static int no1 = 10, no2 = 10, no3 = 21, no4=22, sum;

public void OperatorAndConversion()

{

int num = 0; int[] num1 = { };

//Console.WriteLine("\*\*\*\*\*\*\*\*\*Arithmatic Operation\*\*\*\*\*\* \n");

sum = no1 + no2;

Console.WriteLine("Addition: \t" + sum);

Console.WriteLine($"Substraction: \t {no3 - no2}");

Console.WriteLine($"Multiplication:\t {no3 \* no1}");

Console.WriteLine($"Division is: \t {no3 / no2} \n");

Console.WriteLine("\*\*\*\*\*\*\*\*\*Assignment Operation\*\*\*\*\*\* \n");

Console.WriteLine($"Assign Value \t {no1 = no3}");

Console.WriteLine($"Add Equal \t {sum += no3}");

Console.WriteLine($"Sub Equal \t {sum -= no3} \n");

Console.WriteLine("\*\*\*\*\*\*\*Prefix PostFix Operation\*\*\*\*\*\*\* \n");

//EG: a = 1; >> b = a++; >> b = 1 then a will incremenet .: a = 2 and b = 1 // Postfix

Console.WriteLine($"Postfix -- :\t {sum = no3--} ");

Console.WriteLine($"Postfix ++ : \t {sum = no3++} \n");

//EG: a = 1; >> b = ++a; >> 1st a will incremenet .: a = 2 then b = a and b = 2 // Prefix

Console.WriteLine($"--Prefix : \t {sum = --no3}");

Console.WriteLine($"++ Prefix :\t {sum = ++no3} \n");

Console.WriteLine("\*\*\*\*\*\*\*\*\*Logical & Relational\*\*\*\*\*\* \n");

if (no1 == no2 || no3 == no4)

{

//Console.WriteLine("The Number is Equal");

Console.WriteLine(">>>>>>>>>>The Number is Greater");

}

else

{

Console.Write(">>>>>>>>>>>>>>Less Number \n");

//Console.Write("Not Equal");

}

Console.WriteLine();

Console.WriteLine("\*\*\*\*\*\*\*\*\*Misslenius Operation\*\*\*\*\*\* \n");

Console.WriteLine("Size of DataType int is: " + sizeof(int));

Console.WriteLine("Size of DataType int is: " + sizeof(float));

Console.WriteLine("Type of : "+ typeof(S3\_OperatorAndTypeConversion));

Console.WriteLine("Type of ArrayDataType : " + typeof(int[]));

Console.WriteLine("Type of integer Data : " + typeof(int));

//Console.WriteLine("sizeof Array : " + sizeof(string[]));

Console.WriteLine();

Console.WriteLine("\*\*\*\*\*\*\*\*\*Ternary or Condition Operator \*\*\*\*\*\* \n");

sum = no2 > no3 ? no2 : no3;

Console.WriteLine("Conditional Operator " +sum+ "\n");

Console.WriteLine("\*\*\*\*\*\*\*\*\*TYPE CONVERSION : Implicit \*\*\*\*\*\* \n");

int number = 66;

long dataNo = number;

float value = dataNo;

Console.WriteLine($"Value of Number : {number} \t value of Data : {dataNo} \t value : {value}");

Console.WriteLine("\*\*\*\*\*\*\*\*\*TYPE CONVERSION : Explicit \*\*\*\*\*\* \n");

double decimalValue = 1122.880F;

int a;

a = Convert.ToInt32(decimalValue);

//a = (int)decimalValue;

//a = Convert.ToInt32(decimalValue);

Console.WriteLine("Value of A after Convertion: " +a );

Console.WriteLine("\*\*\*\*\*\*\*\*\*Boxing Unboxing \*\*\*\*\*\* \n");

//Boxing

string name = "Imran";

object fName = name;

name = "Aniket";

Console.WriteLine("Hello : " +name);

Console.WriteLine("Hello Boxer Object : " + fName);

//Unboxing

string lName = Convert.ToString(fName);

Console.WriteLine("Hello Unboxer Object : " + lName);

}

}

}

**Program.cs**

using AllSession;

using Session1.nestedNamespace; // Assembly Refrence OR Namespace OR PAckage

using System;

namespace Session1

{

class Program

{

static void Main(string[] args)

{

// Session 1

//Namespace nameSpace = new Namespace();

//NestedName nest = new NestedName();

//nameSpace.Multiplication();

//nameSpace.Addition();

//nest.Multiplication();

//Console.ReadLine();

//Session 2 Varibale and Data Types

//S2\_VaribaleAndDatatypes varibaleDatatype = new S2\_VaribaleAndDatatypes();

//varibaleDatatype.VariablesAndUses();

//Session 3 Operator and Conversion

S3\_OperatorAndTypeConversion operators = new S3\_OperatorAndTypeConversion();

operators.OperatorAndConversion();

}

}

}