**S2\_\_OperatorsAndConversion**

**Operators:** Operator it’s a symbol used to perform some operations. Its foundation of any pg Language. Operator is used with operand.

* **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.(= =,= = = : Strict Equality Operator it Checks the datatype as well != , < (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 OR Type Operator :** is used to as per the requirement.
  + **sizeof() :-** return the data type size.
  + **typeof() :-** return the type of class or variable.
  + **Instanceof :-** returns true if an object is an instance of an object type.
  + **?: 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

Program

// Operators

let a = 20, b=20 ,c=0,d="Imran";

c=a+b;

console.log("Addition of ++ :"+c); //40

c++;

console.log("Value of ++ :"+c); // 41

c--;

console.log("Value of --:"+c); // 40

console.log(a==b);

console.log(a===d);

// Implicit Conversion

let implicitConversion = 'Imran' + 2;

console.log(implicitConversion)

implicitConversion = 'DataAlgorithm' + true;

console.log(implicitConversion);

implicitConversion = 'NumberUndefined' + undefined;

console.log(implicitConversion);

// Explicit Conversion

let explicitConversion = "1223";

console.log(parseInt(explicitConversion));

explicitConversion = parseFloat('324e-1')

console.log(explicitConversion);

console.log(typeof(explicitConversion));

// boolean to number

explicitConversion = Number(true);

console.log(explicitConversion);