Topic5(Arithmetic Operators)

C++ Operators

> Operators are used to perform operations on variables and values.

In the example below, we use the `+` **operator** to add together two values:

```
int x = 100 + 50;
```

Although the `+` operator is often used to add together two values, like in the example above, it can also be used to add together a variable and a value, or variable and another variable:

```
int sum1 = 100 + 50;        // 150 (100 + 50)
int sum2 = sum1 + 250;      // 400 (150 + 250)
int sum3 = sum2 + sum2;     // 800 (400 + 400)
```

> **C++ divides the operators into the following groups:**
>
> - Arithmetic Operators
> - Assignment Operators
> - Comparison Operators
> - Logical Operators
> - Bitwise Operators

Arithmetic Operators

> Arithmetic operators are used to perform common mathematical operations.

Operator	Name	Desciption	Example
`+`	Addition	Adds together two values	`x + y`
`-`	Subtraction	Subtracts one value from another	`x - y`
`*`	Multiplication	Multiplies two values	`x * y`
`/`	Division	Divides one value by another	`x / y`
`%`	Modulus	Returns the division remainder	`x % y`
`++`	Increment	Increases the value of a variable by 1	`++x`
`--`	Decrement	Decreases the value of a variable by 1	`--x`

```
#include<iostream>
#include<string>
#include<cmath>

using namespace std;


int main(int argc, char const *argv[])
{
    int x = 8;
    int y = 9;

    // Addtion
    int sum = x + y;
    cout << "sum = " << sum << endl;


    // Subtraction
    int sub = x - y;
    cout << "sub = " << sub << endl;


    // Divide by int only
    int div = x / y;
    cout << "div = " << div << endl;

    // Divide under float variable
    float divf = x / y;
    cout << "divf = " << divf << endl;

    // Divide by float value under float variable
    float z = 9;
    float divfz = x / z;
    cout << "divfz = " << divfz << endl;


    // Multiply
    int mult = x * y;
```

```
    cout << "mult = " << mult << endl;

    // Modulus
    x = 5;
    y = 2;
    int mod = x % y;
    cout << "mod = " << mod << endl;


    //  Increment
    x = 5;
    ++x;
    cout << "increment = " << x << endl;


    // Decrement
    x = 5;
    --x;
    cout << "decrement = " << x << endl;

    // Power
    float power = pow(3,2); // power using cmath library by pow function
    cout << "power = " << power << endl;

    return 0;
}
```

Assignment Operators

> Assignment operators are used to assign values to variables.

In the example below, we use the assignment aperator (`=`) to assign the value **10 to a variable called x:**

```
int x = 10;
```

The addition assignment operator (`+=`) adds a value to a variable:

```
int x = 10;
x += 5;
cout << x;
```

> A list of all assignment operators:

Operator	Example	Same As
`=`	`x = 5`	`x = 5`
`+=`	`x += 3`	`x = x + 3`
`-=`	`x -= 3`	`x = x - 3`
`*=`	`x *= 3`	`x = x * 3`
`/=`	`x /= 3`	`x = x / 3`
`%=`	`x %= 3`	`x = x % 3`
`&=`	`x &= 3`	`x = x & 3`
`|=`	`x |= 3`	`x = x | 3`
`^=`	`x ^= 3`	`x = x ^ 3`
`>> =`	`x >> = 3`	`x = x >> 3`
`<< =`	`x << = 3`	`x = x << 3`

```
#include<iostream>
#include<string>

using namespace std;

int main(int argc, char const *argv[])
{
    // Declare variables
    int x;


    // Simple Assignment(x = 5)
    x = 5;
    cout << "Assignment = " << x << endl;


    // Addition Assignment(x = x + 3)
    x = 5;
    x += 3;
    cout << "Addition Assignment = " << x << endl;


    // Subtraction Assignment(x = x - 2)
    x = 5;
    x -= 2;
    cout << "Subtraction Assignment = " << x << endl;


    // Multiply Assignment(x = x * 4)
```

```
    x = 5;
    x *= 4;
    cout << "Multiply Assignment = " << x << endl;


    // Divide Assignment(x = x / 4)
    x = 5;
    x /= 4;
    cout << "Divide Assignment = " << x << endl;


    // Modulus Assignment(x = x % 4) or remainder assignment
    x = 5;
    x %= 2;
    cout << "Modulus Assignment = " << x << endl;

    // Bitwise AND Assignment(a &= b)
    x = 5;
    x &= 3;
    cout << "Bitwise Assignment = " << x << endl;

    // Bitwise OR Assignment(a |= b)
    x = 5;
    x |= 3;
    cout << "Bitwise OR Assignment = " << x << endl;

    // Bitwise XOR Assignment(a ^= b)
    x = 5;
    x ^= 3;
    cout << "Bitwise XOR Assignment = " << x << endl;

    // Bitwise left shift Assignment(a <<= b)
    x = 5;
    x <<= 1;
    cout << "Bitwise Left Shift = " << x << endl;

    // Bitwise right shift Assignment(a >>= b)
    x = 5;
    x >>= 1;
    cout << "Bitwise Right Shift = " << x << endl;

    return 0;
}
```