Operator & Expression Part – 1

An operator is a symbol that tells the compiler to perform specific mathematical or logical functions. C language is rich in built-in operators and provides the following types of operators −

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Bitwise Operators
- Assignment Operators

Arithmetic Operators

An arithmetic operator performs mathematical operations such as addition, subtraction and multiplication on numerical values (constants and variables).

Operator	Meaning of Operator
+	addition or unary plus
-	subtraction or unary minus
*	multiplication
/	division
%	remainder after division(modulo division)

Example : Arithmetic Operators

```
// C Program to demonstrate the working of arithmetic operators
#include <stdio.h>
int main()
{
    int a = 9,b = 4, c;

    c = a+b;
    printf("a+b = %d \n",c);

    c = a-b;
    printf("a-b = %d \n",c);

    c = a*b;
    printf("a*b = %d \n",c);

    c=a/b;
    printf("a/b = %d \n",c);

    c=a%b;
    printf("Remainder when a divided by b = %d \n",c);

    return 0;
}
```

Output

```
a+b = 13
a-b = 5
a*b = 36
a/b = 2
Remainder when a divided by b=1
```

The operators +, - and * computes addition, subtraction and multiplication respectively as you might have expected.

In normal calculation, `9/4 = 2.25`. However, the output is 2 in the program.

It is because both variables a and b are integers. Hence, the output is also an integer. The compiler neglects the term after decimal point and shows answer 2 instead of 2.25.

The modulo operator % computes the remainder. When `a = 9` is divided by `b = 4`, the remainder is 1. The % operator can only be used with integers.

```
Suppose a = 5.0, b = 2.0, c = 5 and d = 2. Then in C programming,
a/b = 2.5  // Because both operands are floating-point variables
a/d = 2.5  // Because one operand is floating-point variable
c/b = 2.5  // Because one operand is floating-point variable
c/d = 2     // Because both operands are integers
```

Program: Converting given days into months and days

```c
#include<stdio.h>
int main()
{
    int months, days;

    printf("Enter Days: ");
    scanf("%d",&days);

    months = days/30;
    days = days%30;

    printf("Months: %d\nDays: %d\n",months,days);
    return 0;
}
```

Increment and decrement operators

C programming has two operators increment ++ and decrement -- to change the value of an operand (constant or variable) by 1.

Increment ++ increases the value by 1 whereas decrement -- decreases the value by 1. These two operators are unary operators, meaning they only operate on a single operand.

Example: Increment and Decrement Operators

```c
#include <stdio.h>
int main()
{
    int a = 10, b = 100;
    float c = 10.5, d = 100.5;

    printf("++a = %d \n", ++a);

    printf("--b = %d \n", --b);

    printf("++c = %f \n", ++c);

    printf("--d = %f \n", --d);
    return 0;
}
```

Output

```
++a = 11
--b = 99
++c = 11.500000
++d = 99.500000
```

Here, the operators ++ and -- are used as prefix. These two operators can also be used as postfix like `a++` and `a--`.

++ and -- operator as prefix and postfix

Suppose you use ++ operator as prefix like: `++var`. The value of *var* is incremented by 1 then, it returns the value.

Suppose you use ++ operator as postfix like: `var++`. The original value of *var* is returned first then, *var* is incremented by 1.

This is demonstrated examples in 4 different programming languages.

Example:

```
#include <stdio.h>
int main()
{
  int var=5;

  // 5 is displayed then, var is increased to 6.
  printf("%d\n",var++);

  // Initially, var = 6. It is increased to 7 then, it is displayed.
  printf("%d",++var);

  return 0;
}
```

Exercise: *** Deadline: 5 - 4 - 2018 ***

1. **Open an account in URI online judge**
2. **Solve First 10 problems (1-10) from beginner section <<MUST>>**
3. **Solve Next 10 problems (11-20) from beginner section <<optional>>**