ARITHMETIC EXPRESSIONS IN C PROGRAMMING - I

C has a wide range of operators. An arithmetic expression is composed of operators and operands. Operators act on operands to yield a result. Commonly used arithmetic operators are +, -, *, / and %.

The plus sign (+) is used to add two values, the minus sign (-) to subtract one value from another, the asterisk(*) to multiply two values, the division (/) to divide a value and the modulus (%) to obtain the reminder of integer division. These are known as *binary operators* since they operate on two values or variables.

Following are examples of arithmetic expressions :

```
result = x - y;
total = principle + interest;
numsquare = x * x;
celcius = (fahrenheit - 32) / 1.8
```

Notice the equal sign (=) in the above expressions, it is known as the *assignment operator*. It assigns the value on the right hand side of the equal sign to the variable on the left hand side.

In the last expression, parentheses are used to perform a certain operation first. This is because in C, operators follow a precedence rule. *, / and % have a higher precedence over + and -. Hence to override the precedence, parentheses should be used. Expressions having operators of the same precedence are generally evaluated from left to right. Another point to note is that in an expression which involves division, care should be taken to avoid a division by zero, since this results in infinity or an abnormal value. In Chapter 5 on control statements, we will see how a check can be done before a division occurs and prevent such operations.

Program 4.1

```
#include <stdio.h>

main()
{
    int var1 = 10;
    int var2 = 2;
    int var3 = 35;
    int var4 = 8;
    int result;
      result = var1 + var2;

      printf("Sum of var1 and var2 is %d\n", result);
      result = var3 * var3;
      printf("Square of var3 is %d\n", result);
      result = var2 +var3 * var4; /* precedence */
      printf("var2 + var3 * var4 =%d\n", result);
}
```

Unary operator

A unary operator is one which operates on one value or operand. The minus sign (-) plays a dual role, it is used for subtraction as a binary operator and for negation as a unary operator. This operator has a precedence higher than the rest of the arithmetic operators.

result = -x * y;

in the above expression, if x has a value 20 and y has a value 2, then result will contain a negative value of 40 which is -40.

In C, some operators are a shorthand equivalent. These are increment (++) and decrement (--) operators. These can be pre-fixed or post-fixed. When these are pre-fixed to a variable in an expression, then the value is computed before the expression is evaluated. When these are post-fixed, the value is computed after the expression is evaluated.

Consider the output of the following program :

Program 4.2

```
/* Usage of pre-fixing and post-fixing the increment operator */

#include <stdio.h>

main()

{

    int x = 10;

    printf("Value of x after pre-fixing ++ is %d\n", ++x);

    printf("Value of x after post-fixing ++ is %d\n",x++);

}
```

The above program will yield the following output:

Value of x after pre-fixing ++ is 11
is 11

Notice that even after we post-fix an increment operator to x, we still get the result as 11. This is because x is evaluated before the increment operation occurs.

Comma operator

Comma can be used in an expression. Each comma seperated expression is evaluated and the value of the rightmost expression will be returned.

Consider the following example :

Program 4.3

```
/* Comma seperated expressions */

#include <stdio.h>
main()
{
    int Total, Val, Count;
        Total = 100;
        Val = 10;
        Count = 50;
        Total =(Val++,Count-2);
        printf("%d\n", Total);
}
```

The above program displays a value of 48, since the last expression Count-2 is considered last. If the parentheses had been absent, then the value in Val before increment would have been returned to Total, this is because the assignment operator has a higher precedence then the comma operator. Precedence of operators is explained in the last section of this chapter.

Source : http://www.peoi.org/Courses/Coursesen/cprog/frame4.html