算术运算符

March 21, 2017

运算符是任何程序语言的基础，运算符使得我们能对操作数做不同的运算。在C中，运算符可分为以下几类：

1. 算术运算符（+， -， *， /， %，后缀自增，前缀自增，后缀自减，前缀自减）
2. 复制运算符（=，+=，-=，*=，/=，%=，...）
3. 关系运算符（==，!=，>，<，>=，<=）
4. 逻辑运算符（&&，||，!）
5. 位运算符（&，|，^， ，>>，<<）
6. 其他运算符(条件运算符，逗号运算符，sizeof，取址运算符&，取值运算符*)

算术运算符用于对操作数执行算术/数学操作。二元操作符包括：

- 加法：运算符 + 对两个操作数相加。如x + y。
- 减法：运算符 - 将第一个操作数减去第二个操作数。如x - y。
- 乘法：运算符 + 对两个操作数相加。如x * y。
- 除法：运算符 / 将第一个操作数除以第二个操作数。如x / y。
- 求余：运算符 % 求第一个操作数除以第二个操作数的余数。如x % y。

```
// C program to demonstrate working of binary arithmetic operators
#include<stdio.h>

int main()
{
    int a = 10, b = 4, res;

    //printing a and b
    printf("a␣is␣%d␣and␣b␣is␣%d\n", a, b);

    res = a+b; //addition
    printf("a+b␣is␣%d\n", res);

    res = a-b; //subtraction
    printf("a-b␣is␣%d\n", res);

    res = a*b; //multiplication
    printf("a*b␣is␣%d\n", res);

    res = a/b; //division
    printf("a/b␣is␣%d\n", res);
    res = a%b; //modulus
    printf("a%%b␣is␣%d\n", res);

    return 0;
}
```

```
Output:
a is 10 and b is 4
a+b is 14
a-b is 6
a*b is 40
a/b is 2
a\%b is 2
```

一元运算符包括：

- 自增: 运算符 ++ 使得一个整型值加1。 当它作用于变量名之前时(称为前缀自增运算符)，其值立即自增1；而当它作用于变量名之后时(称为后缀自增运算符)，其值暂时保持不变，直至所在语句执行完毕，而在下一条语句执行之前将自增1。
- 自减：运算符 -- 使得一个整型值减1。 当它作用于变量名之前时(称为前缀自减运算符)，其值立即自减1；而当它作用于变量名之后时(称为后缀自减运算符)，其值暂时保持不变，直至所在语句执行完毕，而在下一条语句执行之前将自减1。

```
// C program to demonstrate working of Unary arithmetic operators
#include<stdio.h>

int main()
{
    int a = 10, b = 4, res;

    // post-increment example:
    // res is assigned 10 only, a is not updated yet
    res = a++;
    printf("a␣is␣%d␣and␣res␣is␣%d\n", a, res); //a becomes 11 now


    // post-decrement example:
    // res is assigned 11 only, a is not updated yet
    res = a--;
    printf("a␣is␣%d␣and␣res␣is␣%d\n", a, res);  //a becomes 10 now


    // pre-increment example:
    // res is assigned 11 now since a is updated here itself
    res = ++a;
    // a and res have same values = 11
    printf("a␣is␣%d␣and␣res␣is␣%d\n", a, res);


    // pre-decrement example:
    // res is assigned 10 only since a is updated here itself
    res = --a;
    // a and res have same values = 10
    printf("a␣is␣%d␣and␣res␣is␣%d\n",a,res);

    return 0;
}
```

```
Output:
a is 11 and res is 10
a is 10 and res is 11
a is 11 and res is 11
a is 10 and res is 10
```