C++- Other common operators

1. The update family of operators

The family of operators +=, -=, *= and /=   are abbreviated versions of the following:

a = a + k;	a += k;
a = a – k;	a -= k;
a = a * k ;	a *= k;
a = a /k ;	a /= k;

For example, you can write:

```
a = a * 3      or     a  *= 3;
```

2. The increment and decrement operators

The meaning of the ++ and - - symbols changes depending on whether they are written before or after a variable.

Before:

Let’s say we had defined:

```
int x = 7;
int y  = 9;
```

The operations:

```
++x;
 ++y;
```

will increment x and y by one and return the NEW value.

Therefore:

```
cout << ++x << “   “ << ++y << endl;
```

would display 8 followed by 10 , and both x and y would contain the new values.

After:

On the other hand, if we start with the original values of x and y (7 and 9 respectively):

```
  cout << x++  << “   “ << y++ << endl;
```

will display the OLD value, but x and y will actually be incremented by 1. Therefore, in both cases (before and after) x and y will incremented, but in the second case, the old value will be used before incrementing them.

If the increment / decrement expressions are not being used in an assignment or to output their value, their effect is equivalent. That is, the two examples below behave the same, even though in one case we wrote x++ in the first one and ++x in the second one.

```
int x = 0;
while ( x < 5) {
     cout << x;
     x++;
}
```

AND

```
int x = 0;
while ( x < 5) {
     cout << x;
     ++x;
 }
```

3.The ?: operator

This operator performs a test on a condition (much as an if) and returns the value of the first option after the **?** sign if the condition evaluates to true, otherwise returns the value of the option following the colon.

One difference between this operator and the if..then is that the if is not an operator, that is, it does not have a result.

For example the code:

```
#include<iostream>
using namespace std;

main()
{
    int a = 7, b = 9;
    cout << (a *=3) << endl;
    cout << (a > 20 ? b : b+3);
    return 0;
}
```

would print 21, followed by 9.