

**ASSIGNMENTS OPERATORS
INCREMENT/DECREMENT
OPERATORS
RELATIONAL OPERATORS
LOGICAL OPERATORS**

IN C++ LANGUAGE

Setting By : Lect. Waleed Rasheed

Third Lecture

Assignment Operators :

The left-hand operand of an assignment operator must be a nonconst value. Each of these assignments is illegal:

```
int i, j, ival;  
const int ci = i; // ok: initialization not assignment  
1024 = ival; // error: literals are values  
i + j = ival; // error: arithmetic expressions are  
values  
ci = ival; // error: can't write to ci
```

we can perform multiple assignments in a single expression, provided that each of the operands being assigned is of the same general type

```
int ival, jval;  
ival = jval = 0; // ok: each assigned 0
```



This program illustrates the uses of some simple variable.

```
#include <iostream>
using namespace std;
int main ( )
{
int workDays;
float workHours, payRate, weeklyPay;
workDays = 5;
workHours = 7.5;
payRate = 38.55;
weeklyPay = workDays * workHours * payRate;
//arithmetic statement
cout << "Weekly Pay = "; //output statement
cout << weeklyPay;
cout << '\n';
return 0;
}
```



This program illustrates the use of >> for input.

```
#include <iostream>
using namespace std;
int main ( )
{ int workDays = 5;
float workHours = 7.5;
float payRate, weeklyPay;
cout << "What is the hourly pay rate? ";
cin >> payRate;      //Input statement
weeklyPay = workDays * workHours * payRate;
cout << "Weekly Pay = ";
cout << weeklyPay;
cout << '\n';
return 0;
}
```

**H.W.//write program to find circle volume and area?
when radius = 10**



Standard Mathematical Functions

The header `<math.h>` provide what is commonly referred to as “the usual mathematical functions:”

```
double abs(double) ; // absolute value;
```

```
double sqrt(double d) ; // square root of d, d must be  
nonnegative
```

```
double pow(double d, double e) ; // d to the power of e,  
// error if  $d==0$  and  $e \leq 0$  or if  $d < 0$  and  $e$   
isn't an integer.
```

```
double pow(double d, int i) ; // d to the power of i;
```



Q//Find z value, when $z=x^y$?

```
#include<iostream>
#include<math.h>
using namespace std;
int main()
{
int x,y,z;
cout<<"enter base"; cin>>x;
cout<<"enter foundation"; cin>>y;
z=pow(x,y);
cout<<"z="<<z;
return 0;
}
```



Increment/Decrement Operators

The auto increment (++) and auto decrement (--) operators provide a convenient way of, respectively, adding and subtracting 1 from a numeric variable.


The examples assume the following variable definition:

int k = 5;

Increment and decrement operators.

Operator	Name	Example
++	Auto Increment (prefix)	++k + 10 // gives 16
++	Auto Increment (postfix)	k++ + 10 // gives 15
--	Auto Decrement (prefix)	--k + 10 // gives 14
--	Auto Decrement (postfix)	k-- + 10 // gives 15

Both operators can be used in prefix and postfix form. The difference is significant. When used in prefix form, the operator is first applied and the outcome is then used in the expression. When used in the postfix form, the expression is evaluated first and then the operator applied.



Increment/Decrement Operators

X=X+2;	X+=2;
X=X-3;	X-=3;
X=X*2;	X*=2;
X=X/2;	X/=2;

X=X+Y;	X+=Y;
X=X-Y;	X-=Y;
X=X*Y;	X*=Y;
X=X/Y;	X/=Y;



Relational Operators

C++ provides six relational operators for comparing numeric quantities. Relational operators evaluate to 1 (representing the true outcome) or 0 (representing the false outcome).

Operator	Name	Example
<code>==</code>	Equality	<code>5 == 5 // gives 1</code>
<code>!=</code>	Inequality	<code>5 != 5 // gives 0</code>
<code><</code>	Less Than	<code>5 < 5.5 // gives 1</code>
<code><=</code>	Less Than or Equal	<code>5 <= 5 // gives 1</code>
<code>></code>	Greater Than	<code>5 > 5.5 // gives 0</code>
<code>>=</code>	Greater Than or Equal	<code>6.3 >= 5 // gives 1</code>



Logical Operators

C++ provides three logical operators for combining logical expression. Like the relational operators, logical operators evaluate to 1 or 0.

Operator	Name	Example
&&	Logical And	5 < 6 && 6 < 6 // gives 0
 	Logical Or	5 < 6 6 < 5 // gives 1
!	Logical Negation (Not)	!(5 == 5) // gives 0

C++ does not have a built-in boolean type. It is customary to use the type int for this purpose instead.



Q/ W. P. to find y value :

$$y = \sqrt{a^2 + b^2}$$

```
#include<iostream>
#include<math.h>
using namespace std;
int main( )
{
int a,b; float y;
cin>>a>>b;
y=sqrt( pow(a,2) + pow(b,2) );
cout<<" y = " <<y;
return 0;
}
```



Q/ What's Output:

```
#include<iostream>
using namespace std;
int main()
{ int x,y,z;
x=y=z=0;
x=++y + ++z;
cout<<x<<y<<z<<endl;
x=++y - --z;
cout<<x<<y<<z<<endl;
return 0;
}
```

Output

2 1 1

2 2 0

مسودة

0 0 0

2 1 1

2 2 0

