Al-mustaqbal University collage
Biomedical Engineering Department
Class: First
Subject: Computer Skills & Programming

Lecture 4: OPERATORS IN C++ LANGUAGE

BY
IT. Zahraa Abdzaid AbdelAbbas
Supervised by: ASS.T. **Hala Fadel Alaiwi**

1. C++ Operators:

An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations. There are four general classes of operators in C++: **arithmetic**, **relational and logical**, and **bitwise**. In addition, there are some special operators for particular tasks.

1.1 Arithmetic Operators:

Arithmetic operators are used to perform the basic arithmetic operations. They areexplained in the following table:

Operator	Usage	Examples
+	Used for addition	`Sum = a + b`
-	Used for subtraction	`Difference = a – b`
*	Used for multiplication	`Product = a * b`
/	Used for division	`Quotient = a / b`
%	This operator is called the remainder or the modulus operator. It is used to find the remainder after the division. This operator cannot be used with floating type variables.	`Remainder = a % b`

Example (1):

```
#include<iostream.h>

#include<conio.h>

main( ){ int x,y;

cout<< "Enter Two Integers:";

cin>>x>>y;

cout<<"The Intergers are:" <<x<<"and"<<y<<endl;

cout<<"The sum is" <<(x+y)<<endl;

cout<<"The difference is " <<(x-y)<<endl;

cout<<"The product is" <<(x*y)<<endl;

cout<<"The division is" <<(x/y)<<endl;

cout<<"The modulus is" <<(x%y)<<endl;

cout<<"The equation  is   " <<((x+y)/3)<<endl;

getch( );}
```

Output:

```
Enter Two Integers:5
4
The Intergers are:5and4
The sum is      9
The difference is1
The product is20
The division is1
The modulus is 1
The equation is 3
```

Example (2):

Write a program in C ++ to calculate the area of a circle in terms of radius

```
#include<iostream.h>
#include<conio.h>
main( )

{
float R, Area;
   float p1 = 3.14 ;
   cout << " Enter radius of circle: " ;
   cin >> R;
   Area = p1 * R * R;
   cout << " Area is: " << Area << endl;
getch( );
}
```

Output:

```
Enter radius of circle: 4
 Area is: 50.24
```

Increment and Decrement Operator:

C++ allows two very useful operators not generally found in other computer languages. These are the **Increment (++)** and **Decrement (– –)** operators. The operation **++** adds 1 to its operand, and – – subtracts 1.Therefore, the following are equivalent operations:

x = x + 1;   is the same as   ++x;   Or   x++;

Also,

x = x - 1;   is the same as   --x;   Or   x--;

However, there is a difference when they are used in an expression. When an increment or decrement operator precedes its operand, C++ performs the increment or decrement operation prior to obtaining the operand's value.

Operator	Pre or post	Description
++k	Pre-increment	First increase the value of k by one then evaluate the current statement by taking incremented value.
k++	Post-increment	First use the current value of k to evaluate the current statements then increase k by unity.
– –k	Pre-decrement	First decrease the value of k by unity then evaluate the statement.
k– –	Post-decrement	First use the current value of k to evaluate the current statements then decrease k by unity.

Example (3):

Write a program in C++ language to test the operation of arithmetic operators with printing the result appearing on the screen of computer.

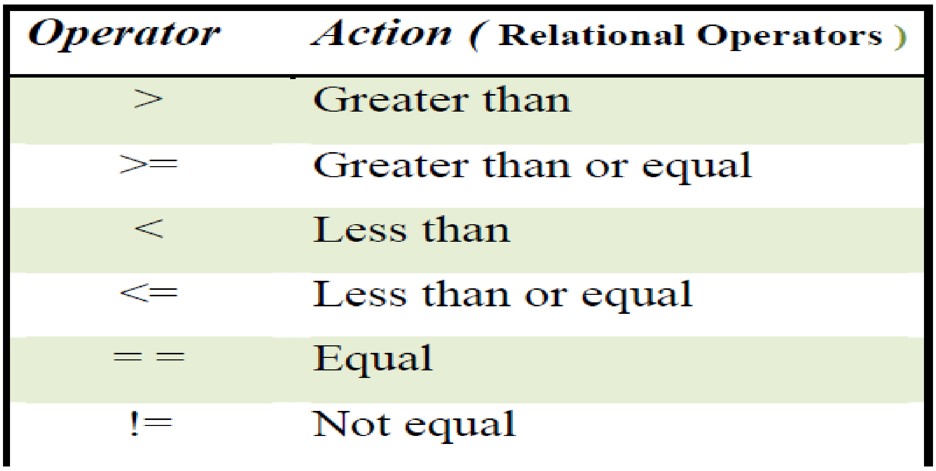
```
#include<iostream.h>
#include<conio.h>
main( )
{
  int a = 21;
   int c ;
   // Value of a will not be increased before assignment.
   c = a++;
   cout << "Line 1 - Value of a++ is :" << c << endl ;
   // After expression value of a is increased
   cout << "Line 2 - Value of a is :" << a << endl ;
   // Value of a will be increased before assignment.
   c = ++a;
   cout << "Line 3 - Value of ++a is  :" << c << endl;
getch( );}
```

The result appearing on the screen of computer is:

```
Line 1 - Value of a++ is :21
Line 2 - Value of a is :22
Line 3 - Value of ++a is  :23
```

1.2 Relational Operators:

In the term *relational operator* the word *relational* refers to the relationships values can have with one another. The key to the concepts of relational operators is the idea of **true** and **false**. In C++, *true* is any value other than 0. *False* is 0. Expressions that use relational operators will return **0** for false and **1** for true.



Operator	*Action* (**Relational Operators**)
>	Greater than
>=	Greater than or equal
<	Less than
<=	Less than or equal
= =	Equal
!=	Not equal

Example (5):

Write a program in C++ language to test the operation of Relational Operators with printing the result appearing on the screen of computer.

Ans:

```
#include<iostream.h>
#include<conio.h>
// Program to test Relational Operators
main()
{
int A=57, B=57;
char C='9';
cout<<(int(C))<<endl;
cout<<"(A<57)="<<(A<57)<<endl;
cout<<"(A<90)="<<(A<90)<<endl;
cout<<"(A<30)="<<(A<30)<<endl;
cout<<"(A<=57)="<<(A<=57)<<endl;
cout<<"(A>B)="<<(A>B)<<endl;
cout<<"(A>=B)="<<(A>=B)<<endl;
cout<<"(A==B)="<<(A==B)<<endl;
cout<<"(A!=B)="<<(A!=B)<<endl;
cout<<"(A==C)="<<(A==C)<<endl;
getch();
}
```

.   .

Output:

```
57

(A<57)=0

(A<90)=1

(A<30)=0

(A<=57)=1

(A>B)=0

(A>=B)=1

(A==B)=1

(A!=B)=0

(A==C)=1
```

1.3 The sizeof( ) operator

In c++ , the sizeof operator is used to determines the size of a variable or any data type . It is a compile-time operator which return the size of variable or data type in bytes.

Syntax:

```
sizeof(type);

  int x;
 int    y=sizeof(x);
```

Example:

```
  #include<iostream.h>
  #include<conio.h>
  #include<math.h>
  main( )
  {
  cout << "Size of int : " << sizeof(int) << endl;
   cout << "Size of long int : " << sizeof(long int) << endl;
     cout << "Size of float : " << sizeof(float) << endl;
     cout << "Size of double : " << sizeof(double) << endl;
     cout << "Size of char : " << sizeof(char) << endl;
  getch();
}
```

Output:

```
  Size of int : 4
  Size of long int : 4
  Size of float : 4
  Size of double : 8
Size of char : 1
```

Example:

```
  int i; char c;
    cout << "Size of variable i : " << sizeof(i) << endl;
    cout << "Size of variable c : " << sizeof(c) << endl;
```

output: **Size of variable i : 4, Size of variable c : 1**

Most of the mathematical functions are declared in the <math.h> header file, as shown inthe table below :

Function	Description	Example
sin(x)	sine of x (x in radians)	sin(2) returns 0.909297
cos(x)	cosine of x (x in radians)	cos (2) returns -0.416147
tan(x)	tangent of x (x in radians)	tan(2) returns -2.18504
asin(x)	inverse sine of x (x in radians)	asin(0.2) returns 0.201358
acos(x)	inverse cosine of x (x in radians)	acos(0.2) returns 1.36944
atan(x)	inverse tangent of x (x in radians)	atan(0.2) returns 0.197396
log(x)	natural logarithm of x (base e)[Ln(x)]	log(2) returns 0.693147
log10(x)	common logarithm of x (base 10)	Logl0(2) returns 0.30103
sqrt(x)	square root of x	sqrt(2) returns 1.41421
pow(x,p)	x to the power p	pow(2,3) returns 8.0

Example:

```
#include<iostream.h>
#include<conio.h>
#include<math.h>
int main(){
int x = 2;
  double result;
  result = sin(x);
  cout << "sin(x) = " << result << endl;
getch();}
```

output:

```
sin(x) = -0.841471
```

Thank you