Programming Fundamentals (COMP1112)
Lecture 3
Operators and Expressions

Division of Science & Technology
University of Education, Lahore.

Operators

- An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations.
- C++ is rich in built-in operators.
- Some commonly used operators will be discussed in detail.

Types of operators

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Bitwise Operators
- Assignment Operators
- Misc Operators

Binary arithmetic operator

(takes one operator and two operands)

Operator	Description	Example (where A=10, B=20)
+	Adds two operands	A + B will give 30
-	Subtracts second operand from the first	A - B will give -10
*	Multiplies both operands	A * B will give 200
/	Divides numerator by de-numerator	B / A will give 2
%	Modulus operator gives remainder after an integer division	B % A will give 0

Unary arithmetic operator

(takes one operator and one operand)

Operator	Description	Example (where A=10)
++	Increment operator, increases integer value by one	A++ will give 11
--	Decrement operator, decreases integer value by one	A-- will give 9

Unary arithmetic operator

- Post increment (A++) – first use A then increment
- Post decrement (A--) – first use A then decrement
- Pre increment (++A) – first increment then use A
- Pre decrement(--A) – first decrement then use A

Examples

int A=10;

- cout<<A++; // prints 10
- cout<<A--; // prints 10
- cout<<++A; // prints 11
- cout<<--A; // prints 9

Note: these instructions are not in a sequence form; but written for specifying example and must be read individually.

What will be the value of y?

```
x = 3;
y = ++x;   //value of y?
```

```
x = 3;
y = x++; //value of y?
```

Relational operators

- Two values can be compared using relational and equality operators. For example, to know if two values are equal or if one is greater than the other.

- The result of such an operation is either true or false (i.e., a Boolean value).

Relational operators

operator	description
==	Equal to
!=	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

Logical operators

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non-zero, then condition becomes true.	
||	Called Logical OR Operator. If any of the two operands is non-zero, then condition becomes true.	
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true, then Logical NOT operator will make false.	

Bit-wise operators

- Bitwise Operators
- Bitwise operator works on bits and perform bit-by-bit operation. The truth tables for &, |, and ^ are as follows –

p	q	p & q	p | q	p Xor q
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

Bit-wise operators

Operator	Description
&	Binary AND Operator checks if exists in both operands.
|	Binary OR Operator checks if exists in either operand.
^	Binary XOR Operator checks if in one operand but not both.
~	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.

Assignment operators

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand.	C = A + B will assign value of A + B into C
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand.	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand.	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand.	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand.	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand.	C %= A is equivalent to C = C % A

Other operators (examples)

- **Condition ? X : Y**

If Condition is true then it returns value of X otherwise returns value of Y.

- **Sizeof**

returns the size of a variable. For example, sizeof(a), where ‘a’ is integer, and will return 4.

Expressions

A combination of variables, constants and operators that represents a computation forms an expression. Expression evaluation depends on operators precedence and associativity.

Expressions forms (some examples)

- Arithmetic
- Boolean
- Relational
- Bitwise
- Constants

Operators precedence and associativity

- Operator precedence determines the grouping of terms in an expression. This affects how an expression is evaluated. Certain operators have higher precedence than others; for example, the multiplication operator has higher precedence than the addition operator −
- For example x = 7 +3 * 2; here, x is assigned 13, not 20 because operator * has higher precedence than +, so it first gets multiplied with 3*2 and then adds into 7.
- In programming languages, the associativity of an operator is a property that determines how operators of the same precedence are grouped in the absence of parentheses

Operators precedence and associativity

Category	Operator	Associativity
Brackets	() []	Left to right
Unary	++ - -	Left to right
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Equality	=, +=, -=, *=, %=, /=	Right to left

Note: operators with the highest precedence appear at the top of the table, those with the lowest appear at the bottom. Within an expression, higher precedence operators will be evaluated first.

Operators precedence and associativity

Category	Operator	Associativity
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right

Programming example - 1
(operators and expressions)

```
#include <iostream>

using namespace std;
int main ()
 {
   int a, b;
   a = 10;
   b = 4;
   a = b;
   b = 7;
  cout << "a:";
  cout << a;
  cout << " b:";
  cout << b;
}
```

Output:
a:4 b:7

Programming example 2
(operators and expressions)

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

int main ()
{

int a, b=3;
a = b;
a+=1;  //a=a+1;
cout << a;

}
```

Output:
4

Programming example 3
(operators and expressions)

```
#include<iostream>

using namespace std;

main() {

int a = 20;
int b = 10;
int c = 15;
int d = 5;
int e;

e = (a + b) * c / d;
cout << "Value of (a + b) * c / d is :" << e << endl ; //1
e = ((a + b) * c) / d;
cout << "Value of ((a + b) * c) / d is :" << e << endl ;//2
e = (a + b) * (c / d);
cout << "Value of (a + b) * (c / d) is :" << e << endl ;//3
e = a + (b * c) / d;
cout << "Value of a + (b * c) / d is :" << e << endl ;//4

return 0;

}
```

References

- C++ How to Program
  By Deitel & Deitel

- The C++ Programming Language
  By Bjarne Stroustrup

- Object oriented programming using C++ by Tasleem Mustafa, Imran Saeed, Tariq Mehmood, Ahsan Raza

- https://www.tutorialspoint.com/cplusplus/cpp_operators.htm

- http://ecomputernotes.com/cpp/introduction-to-oop/expressions-in-cpp

- http://www.cplusplus.com/doc/tutorial/operators/