Programming fundamentals

Lecture 6: Operators

Recap

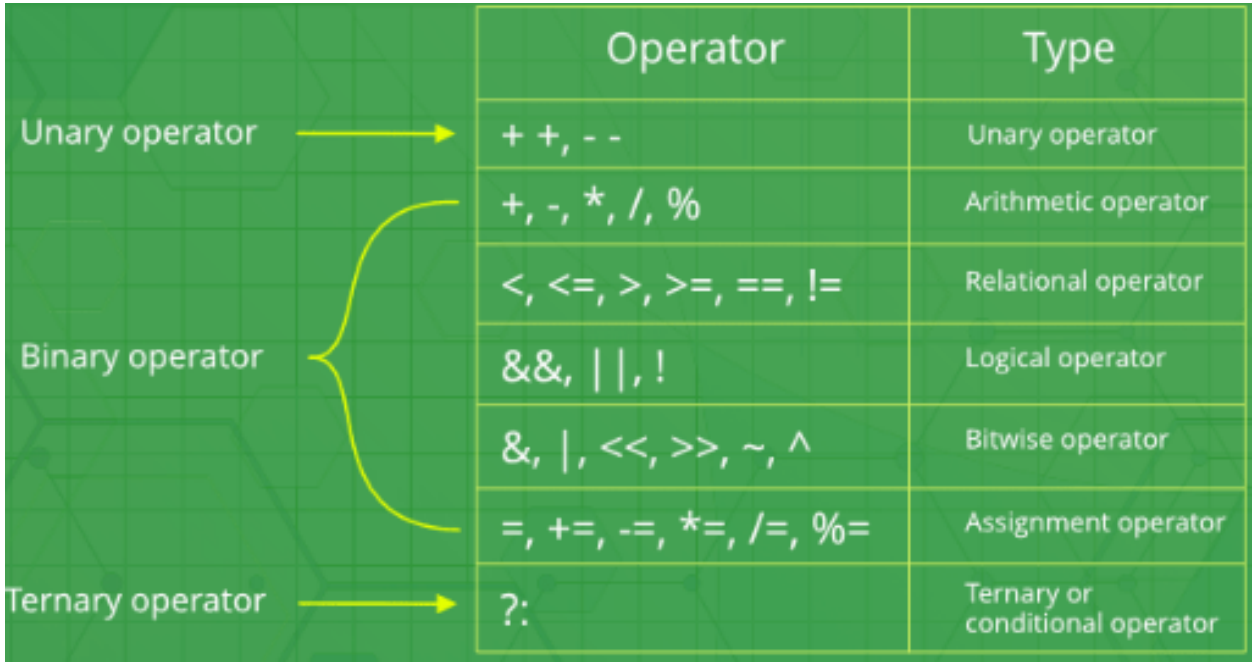
- Data types,
- operators, and
- conditional statements

Agenda

- Operators

Operators

- An *operator* is a symbol that tells the compiler to perform specific mathematical or logical manipulations.
- *Arithmetic, assignment, relational* and *logical,* and *bitwise*



Operator	Type
+ +, - -	Unary operator
+, -, *, /, %	Arithmetic operator
<, <=, >, >=, ==, !=	Relational operator
&&, ||, !	Logical operator
&, |, <<, >>, ~, ^	Bitwise operator
=, +=, -=, *=, /=, %=	Assignment operator
?:	Ternary or conditional operator

Unary

- The increment (++) and decrement (--) operators.

```
int a = 1;
 int b = a++;   // b = 1
 int c = a;     // c = 2
```

- The unary minus (-) operator. ***(changes the sign of its argument. A positive number becomes negative, and a negative number becomes positive.)***

```
int a = 10;
 int b = -a;  // b = -10
```

- The logical not (!) operator. ***(reverse the logical state of its operand. If a condition is true, then Logical NOT operator will make it false)***

```
int a=2;
If(!(a==2)) {            }
```

Increment and decrement

++ and -- (increment and decrement operators

x = x+1;
can be written as

++x; // prefix form

or as

x++; // postfix form

Operator's precedence

highest	++ – –
	– (unary minus)
	* / %
lowest	+ –

Operator(s)	Operation(s)
( )	Parentheses
*	Multiplication
/	Division
%	Modulus
+	Addition
–	Subtraction

e.g. solving expression

Algebra: $z = pr\%q + w/x - y$

C++:

```
z = p * r % q + w / x - y;
```

6 1 2 4 3 5

e.g. same operator used multiple times

Step 1. `y = 2 * 5 * 5 + 3 * 5 + 7;` *(Leftmost multiplication)*

`2 * 5 is 10`

Step 2. `y = 10 * 5 + 3 * 5 + 7;` *(Leftmost multiplication)*

`10 * 5 is 50`

Step 3. `y = 50 + 3 * 5 + 7;` *(Multiplication before addition)*

`3 * 5 is 15`

Step 4. `y = 50 + 15 + 7;` *(Leftmost addition)*

`50 + 15 is 65`

Step 5. `y = 65 + 7;` *(Last addition)*

`65 + 7 is 72`

Step 6. `y = 72` *(Last operation—place 72 in y)*

Relational and Logical Operators

Relational Operators	
Operator	**Meaning**
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to
==	equal to
!=	not equal to

Logical Operators	
Operator	**Meaning**
&&	AND
||	OR
!	NOT

highest	!
	> >= < <=
	= = !=
	&&
lowest	||

Bitwise operators

- The **& (bitwise AND)** in C or C++ takes two numbers as operands and does AND on every bit of two numbers. The result of AND is 1 only if both bits are 1.
- The **| (bitwise OR)** in C or C++ takes two numbers as operands and does OR on every bit of two numbers. The result of OR is 1 if any of the two bits is 1.
- The **^ (bitwise XOR)** in C or C++ takes two numbers as operands and does XOR on every bit of two numbers. The result of XOR is 1 if the two bits are different.
- The **<< (left shift)** in C or C++ takes two numbers, left shifts the bits of the first operand, the second operand decides the number of places to shift.
- The **>> (right shift)** in C or C++ takes two numbers, right shifts the bits of the first operand, the second operand decides the number of places to shift.
- The **~ (bitwise NOT)** in C or C++ takes one number and inverts all bits of it

p	q	p & q	p | q	p ^ q
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

Assume if A = 60; and B = 13; now in binary format they will be as follows −

A = 0011 1100

B = 0000 1101

A&B = 0000 1100

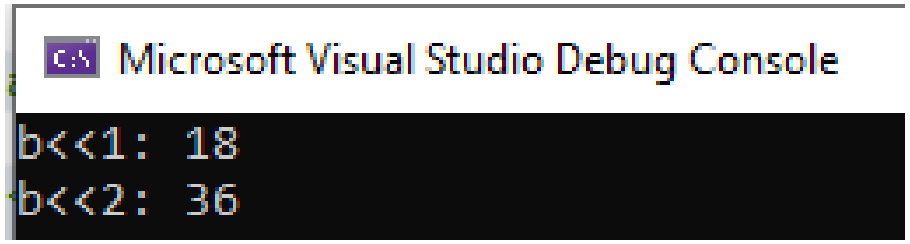
A|B = 0011 1101

A^B = 0011 0001

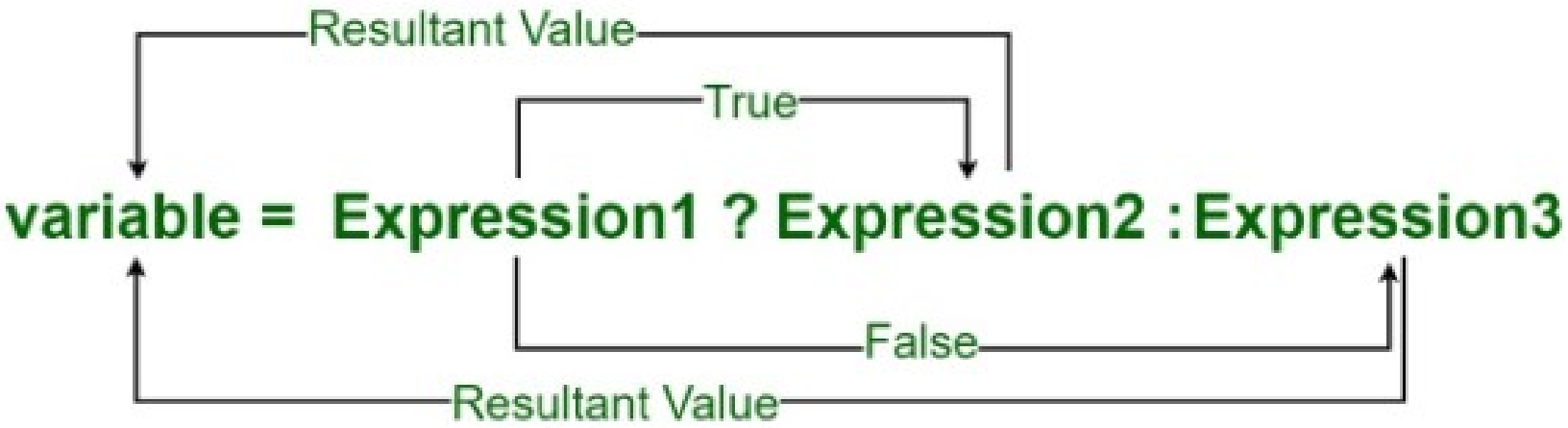
~A = 1100 0011

e.g

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    int b = 9;
    cout << "b<<1: "<<(b<<1)<<endl;
    b = 9;
    cout <<"b<<2: "<< (b << 2)<<endl;
    return 0;
}
```



Ternary operator (conditional operator) *? :*



Consider this code:

```
if ( a < b )
{
  a = b;
}
else
{
  a = -b;
}
```

You can replace the above code with:

```
a = (a < b) ? b : -b;
```

(a<b) is an if condition
b is a true case (if)
-b is a false case (else)

The ternary operator is more readable than a if...else statement for short conditions.

```
#include <iostream>
Using namespace std;
int main()
{
// variable declaration
int n1 = 5, n2 = 10, max;

// Largest among n1 and n2
max = (n1 > n2) ? n1 : n2;

// Print the largest number
Cout<<"Largest number between“;
Cout<<n1<< n2<< max<<endl;

return 0;
}
```

If we want to print something

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    int n1 = 5, n2 = 10, max;

    // Largest among n1 and n2
    string m = (n1 > n2) ? "n1 is greater" : "n2 is greater";
    cout << m;
return 0;
}
```


Punjab University College of Information Technology
PUCIT
Serve Humanity
UNIVERSITY OF THE PUNJAB

Recommended reads

- Dietal & Dietal
  - Chapter 4
    - page 10: **section 4.5:** if selection statement, **4.6**, till page 114
    - Page 139: **Section 4.11:** Assignment operators, **4.12**
- Walter Savitch, Problem Solving with C++ The Object of Programming
  - Chapter 2
    - Page 60-82
- Schildt - C++ From the Ground Up (3rd Edition)
  - Chapter 3