CSE 232

Programming with C++

Lecture 4

C++ Operators, Input-Output

Prepared by

Md. Mijanur Rahman, Prof. Dr.
Dept. of Computer Science and Engineering
Jatiya Kabi Kazi Nazrul Islam University, Bangladesh
Email: mijan@jkkniu.edu.bd
Web: www.mijanrahman.com

Contents

C++ Operators, Input and Output

C++ Operators

- An operator is a symbol that operates on a value to perform specific mathematical or logical computations. They form the foundation of any programming language. In C++, we have built-in operators to provide the required functionality.
- An operator operates the operands. For example,

```
int c = a + b;
```

- Operators in C++ can be classified into 6 types:
  1. Arithmetic Operators
  2. Relational Operators
  3. Logical Operators
  4. Bitwise Operators
  5. Assignment Operators
  6. Ternary or Conditional Operators

Arithmetic Operators…

- C++ Arithmetic operators are of 2 types:
  - Unary Arithmetic Operator
  - Binary Arithmetic Operator

Arithmetic Operators…

- **Binary Arithmetic Operator**
- These operators operate or work with two operands. C++ provides 5 Binary Arithmetic Operators for performing arithmetic functions:

+	Addition	Used in calculating the Addition of two operands	x+y
–	Subtraction	Used in calculating Subtraction of two operands	x-y
*	Multiplication	Used in calculating Multiplication of two operands	x*y
/	Division	Used in calculating Division of two operands	x/y
%	Modulus	Used in calculating Remainder after calculation of two operands	x%y

Arithmetic Operators

- **Unary Operator**
- These operators operate or work with a single operand.

Operator	Symbol	Operation	Implementation
Decrement Operator	—	Decreases the integer value of the variable by one	–x or x —
Increment Operator	++	Increases the integer value of the variable by one	++x or x++

Unary Operator

- Types of Unary Operators in C++: C++ has a total of 8 unary operators:
  1. Increment Operator (++)
  2. Decrement Operator (–)
  3. Unary Plus Operator (+)
  4. Unary Minus Operator (-)
  5. Logical NOT Operator (!)
  6. Bitwise NOT Operator (~)
  7. Addressof Operator (&)
  8. Indirection Operator (*)

C++ Bitwise Operators…

- Bitwise Operators are the operators that are used to perform operations on the bit level on the integers. While performing this operation integers are considered as sequences of binary digits. In C++, we have various types of Bitwise Operators.
  - Bitwise AND (&)
  - Bitwise OR (|)
  - Bitwise XOR (^)
  - Bitwise NOT (~)
  - Left Shift (<<)
  - Right Shift (>>)

C++ Bitwise Operators

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

int main()
{
    int a = 5; //  101
    int b = 3; //  011

    int bitwise_and = a & b;
    int bitwise_or = a | b;
    int bitwise_xor = a ^ b;
    int bitwise_not = ~a;
    int left_shift = a << 2;
    int right_shift = a >> 1;

    cout << "AND: " << bitwise_and << endl;
    cout << "OR: " << bitwise_or << endl;
    cout << "XOR: " << bitwise_xor << endl;
    cout << "NOT a: " << bitwise_not << endl;
    cout << "Left Shift: " << left_shift << endl;
    cout << "Right Shift: " << right_shift << endl;

    return 0;
}
```

Relational Operators in C++

- C++ Relational operators are used to compare two values or expressions, and based on this comparison, it returns a boolean value (either true or false) as the result. Generally false is represented as 0 and true is represented as any non-zero value (mostly 1).

S. No.	Relational Operator	Meaning
1.	>	Greater than
2.	<	Less than
3.	>=	Greater than equal to
4.	<=	Less than equal to
5.	==	Equal to
6.	!=	Not equal to

C++ Logical Operators

- In C++ programming languages, logical operators are symbols that allow you to combine or modify conditions to make logical evaluations. They are used to perform logical operations on boolean values (true or false).

- In C++, there are three logical operators:
  - Logical AND (&&) Operator
  - Logical OR (||) Operator
  - Logical NOT (!) Operator

Assignment Operators In C++…

- In C++, the assignment operator forms the backbone of many algorithms and computational processes by performing a simple operation like assigning a value to a variable.
- It is denoted by equal sign (=) and provides one of the most basic operations in any programming language that is used to assign some value to the variables in C++ or in other words, it is used to store some kind of information.
- Syntax:
  - variable = value;
- The right-hand side value will be assigned to the variable on the left-hand side. The variable and the value should be of the same data type.
- The value can be a literal or another variable of the same data type.

Assignment Operators In C++

- **Compound Assignment Operators**
- In C++, the assignment operator can be combined into a single operator with some other operators to perform a combination of two operations in one single statement. These operators are called Compound Assignment Operators. There are 10 compound assignment operators in C++:
  - Addition Assignment Operator (+=)
  - Subtraction Assignment Operator (-=)
  - Multiplication Assignment Operator (*=)
  - Division Assignment Operator (/=)
  - Modulus Assignment Operator (%=)
  - Bitwise AND Assignment Operator (&=)
  - Bitwise OR Assignment Operator (|=)
  - Bitwise XOR Assignment Operator (^=)
  - Left Shift Assignment Operator (<<=)
  - Right Shift Assignment Operator (>>=)

C++ Ternary or Conditional Operator…

- In C++, the ternary or conditional operator (? :) is the shortest form of writing conditional statements. It can be used as an inline conditional statement in place of if-else to execute some conditional code.
- The syntax of the ternary (or conditional) operator is:
  - expression ? statement_1 : statement_2;

- As the name suggests, the ternary operator works on three operands where
  - expression: Condition to be evaluated.
  - statement_1: Statement that will be executed if the expression evaluates to true.
  - statement_2: Code to be executed if the expression evaluates to false.

C++ Ternary or Conditional Operator

- **C++ Nested Ternary Operator**
- A nested ternary operator is defined as using a ternary operator inside another ternary operator.
- Like if-else statements, the ternary operator can also be nested inside one another.

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

int main()
{

    int A = 45, B = 10, C = 15;

    int maxNum = (A > B) ? ((A > C) ? A : C) : ((B > C) ? B : C);
    cout << "Largest number is " << maxNum << endl;

    return 0;
}
```

Scope Resolution Operator in C++…

- In C++, the scope resolution operator is ::. It is used for the following purposes.
- 1) To access a global variable when there is a local variable with same name:

```
1  #include<iostream>
2  using namespace std;
3
4  int x = 100; // Global x
5
6  int main()
7  {
8  int x = 10; // Local x
9  cout << "Value of global x is " << ::x;
10 cout << "\nValue of local x is " << x;
11 return 0;
12 }
```

Scope Resolution Operator in C++…

- 2) To define a function outside a class.

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

class A {
public:
    void display();
};

// Definition outside class using ::
void A::display() { cout << "Function body"; }

int main()
{
    A a;
    a.display();
    return 0;
}
```

Scope Resolution Operator in C++...

- 3) To access a class's static variables.

```
Value of static x is 10
Value of local x is 30
Test::y = 20

=== Code Execution Successful ===
```

```
1  #include<iostream>
2  using namespace std;
3
4  class Test
5  {
6      static int x;
7  public:
8      static int y;
9      void func(int x)
10     {
11     cout << "Value of static x is " << Test::x;
12     cout << "\nValue of local x is " << x;
13     }
14 };
15
16 int Test::x = 10;
17 int Test::y = 20;
18
19 int main()
20 {
21     Test obj;
22     int x = 30 ;
23     obj.func(x);
24     cout << "\nTest::y = " << Test::y;
25     return 0;
26 }
```

Scope Resolution Operator in C++…

- 4) In case of multiple Inheritance: If the same variable name exists in two ancestor classes, we can use scope resolution operator to distinguish.

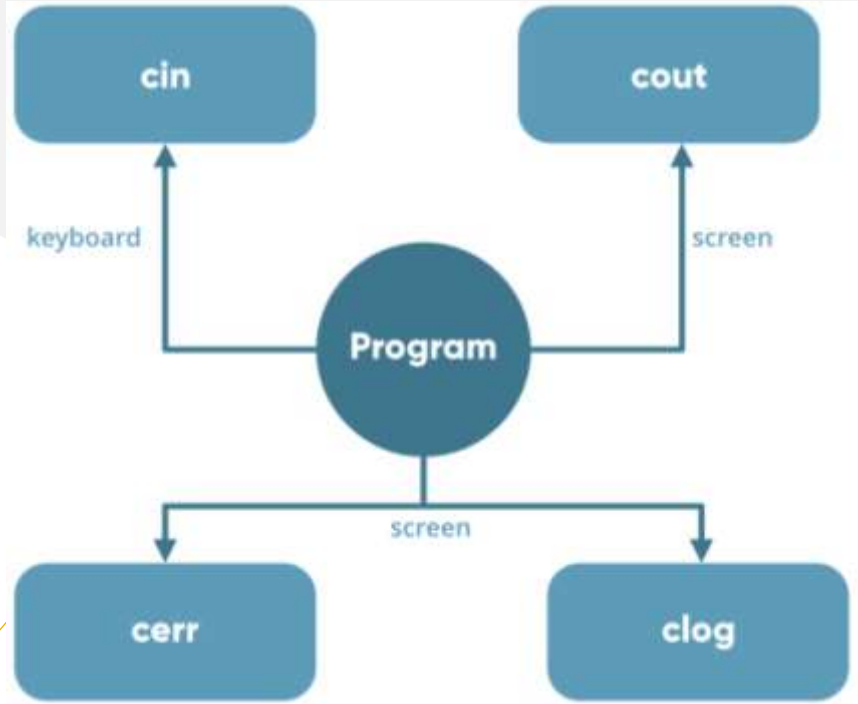
```
A's x is 10
B's x is 20

=== Code Execution Successful ===
```

```
1  #include<iostream>
2  using namespace std;
3
4  class A
5  {
6  protected:
7      int x;
8  public:
9      A() { x = 10; }
10 };
11
12 class B
13 {
14 protected:
15     int x;
16 public:
17     B() { x = 20; }
18 };
19
20 class C: public A, public B
21 {
22 public:
23 void fun()
24 {
25     cout << "A's x is " << A::x;
26     cout << "\nB's x is " << B::x;
27 }
28 };
29
30 int main()
31 {
32     C c;
33     c.fun();
34     return 0;
35 }
```

Basic Input / Output in C++

- C++ comes with libraries that provide us with many ways for performing input and output. In C++ input and output are performed in the form of a sequence of bytes or more commonly known as streams.
  - Input Stream: If the direction of flow of bytes is from the device(for example, Keyboard) to the main memory then this process is called input.
  - Output Stream: If the direction of flow of bytes is opposite, i.e. from main memory to device (display screen) then this process is called output.



Basic Input / Output in C++

- Header files available in C++ for Input/Output operations are:
  - iostream: iostream stands for standard input-output stream. This header file contains definitions of objects like cin, cout, cerr, etc.
  - iomanip: iomanip stands for input-output manipulators. The methods declared in these files are used for manipulating streams. This file contains definitions of setw, setprecision, etc.
  - fstream: This header file mainly describes the file stream. This header file is used to handle the data being read from a file as input or data being written into the file as output.
  - bits/stdc++: This header file includes every standard library. In programming contests, using this file is a good idea, when you want to reduce the time wasted in doing chores; especially when your rank is time sensitive.

Basic Input / Output in C++

- **cin in C++**
- The cin object in C++ is an object of class iostream. It is used to accept the input from the standard input device i.e. keyboard. It is associated with the standard C input stream stdin. The extraction operator(>>) is used along with the object cin for reading inputs. The extraction operator extracts the data from the object cin which is entered using the keyboard.

Basic Input / Output in C++

- **cout in C++**
- The cout object in C++ is an object of class iostream. It is defined in iostream header file. It is used to display the output to the standard output device i.e. monitor. It is associated with the standard C output stream stdout. The data needed to be displayed on the screen is inserted in the standard output stream (cout) using the insertion operator(<<).

Basic Input / Output in C++

- Example: cin and cout in C++

```
Rahman
40
Name : Rahman
Age : 40


=== Code Execution Successful ===
```

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

int main()
{
    string name;
    int age;

    // Take multiple input using cin
    cin >> name >> age;

    // Print output
    cout << "Name : " << name << endl;
    cout << "Age : " << age << endl;

    return 0;
}
```

Basic Input / Output in C++

- The cin can also be used with some member functions which are as follows:
  - cin.getline(char *buffer, int N):

- It reads a stream of characters of length N into the string buffer, It stops when it has read (N – 1) characters or it finds the end of the file or newline character(\n).

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      char name[10];
7
8      cin.getline(name, 4);
9
10     cout << name << endl;
11
12     return 0;
13 }
```

```
Rahman
Rah


=== Code Execution Successful ===
```

Basic Input / Output in C++

- cin.get(char& var):
- It reads an input character and stores it in a variable.

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      char ch[30];
7      cin.get(ch, 10);
8
9      cout << ch;
10 }
```

```
Welcome to JKKNIU
Welcome t

=== Code Execution Successful ===
```

Basic Input / Output in C++

- cin.read(char *buffer, int N):
- Reads a stream of characters of length N.

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      char txt[20];
7
8      // Reads stream of characters
9      cin.read(txt, 10);
10
11     cout << txt << endl;
12
13     return 0;
14 }
```

Basic Input / Output in C++

- The cout statement can also be used with some member functions:

  - cout.write(char *str, int n): Print the first N character reading from str.
  - cout.put(char &ch): Print the character stored in character ch.
  - cout.precision(int n): Sets the decimal precision to N, when using float values.

Basic Input / Output in C++

- cout.write(char *str, int n): Print the first N character reading from str.
- cout.put(char &ch): Print the character stored in character ch.

```
Welcome
A

=== Code Execution Successful ===
```

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

int main()
{
    char txt[] = "Welcome at JKKNIU";
    char ch = 'A';

    // Print first 7 characters
    cout.write(txt, 7);
    cout << endl;
    // Print the character ch
    cout.put(ch);
    return 0;
}
```

Basic Input / Output in C++

- cout.precision(int n): Sets the decimal precision to N, when using float values.

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

int main()
{
    double pi = 3.14159783;

    // Set precision to 5
    cout.precision(5);
    cout << pi << endl;

    // Set precision to 10
    cout.precision(10);
    cout << pi << endl;

    return 0;
}
```

```
3.1416
3.14159783


=== Code Execution Successful ===
```

cerr – Standard Error Stream Object in C++

- **Standard error stream (cerr):** cerr is the standard error stream which is used to output the errors. It is an instance of the **ostream** class.
- As cerr stream is un-buffered so it is used when we need to display the error message immediately and does not store the error message to display later.
- The object of class **ostream** that represents the standard error stream oriented to narrow characters(of type char).

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6
7      cout << "Welcome to JKKNIU!" << endl;
8      cerr << "This is an error message.";
9
10     return 0;
11 }
```

Lecture 4

C++ Operators, Input and Output

THE END