WASHINGTON STATE UNIVERSITY
World Class. Face to Face.

Cpt S 122 – Data Structures

Introduction to C++ Part -I

Nirmalya Roy

School of Electrical Engineering and Computer Science
Washington State University

Topics

- Introduction
- Object Oriented Programming
- First Program in C++: Printing a Line of Text
- Another C++ Program: Adding Integers
- Another C++ Program: Using namespace

Introduction

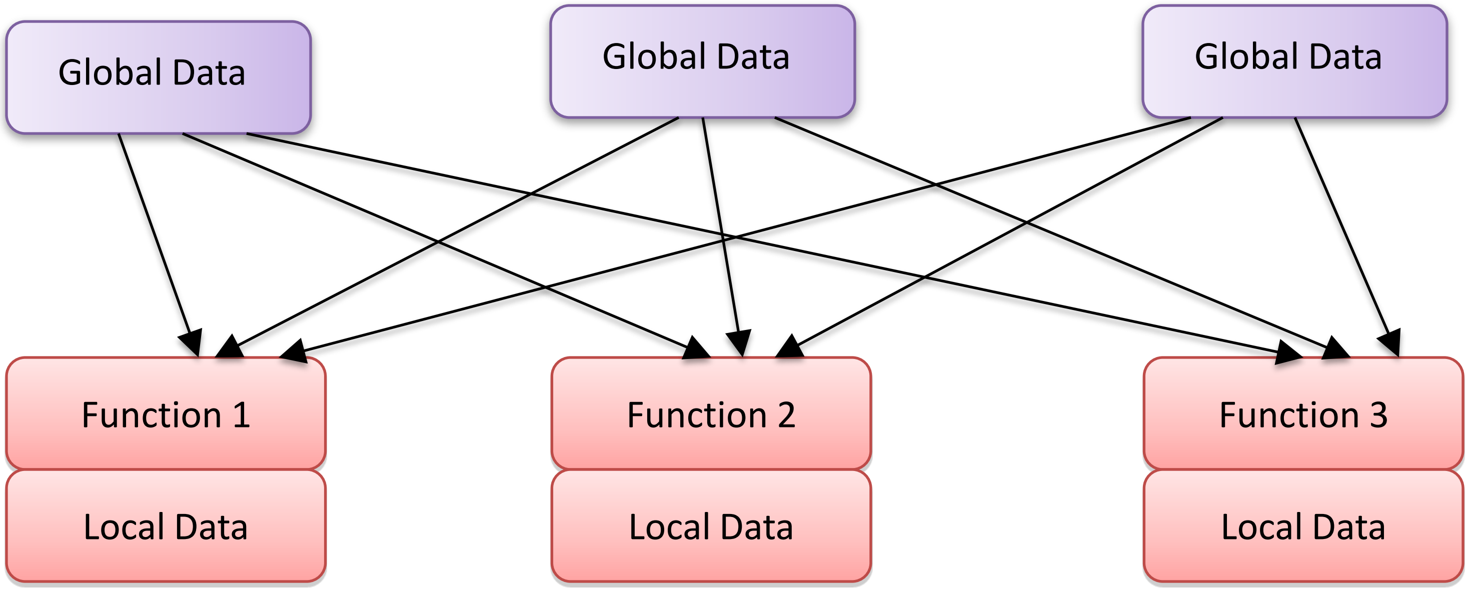
- Programs that employ the basic concepts of object-oriented programming.
  - Typically, consist of function main and
  - one or more classes, each containing data members and member functions.
- Simple, well-engineered framework for organizing object-oriented programs in C++.
- C++ programming facilitates a disciplined approach to program design.

What is an Object Oriented Programming?

- What is a procedure-oriented programming?
    - Example: COBOL, FORTRAN and C etc.
    - Problem is viewed as a sequence of things to be done
        - Example: Reading, calculating and printing
    - A number of functions are written to accomplish this task
    - The primary focus is on functions
    - Very little attention is given to the data that are being used by functions
        - What happens to the data?
        - How are they affected by the functions that work on them?

Procedure-Oriented Programming

- Multi- function program, many important data items are placed as global
  - They may be accessed by all the functions
  - Each function may have its own local data



Relationship of data and functions in procedural programming

Procedure-Oriented Programming (Cont.)

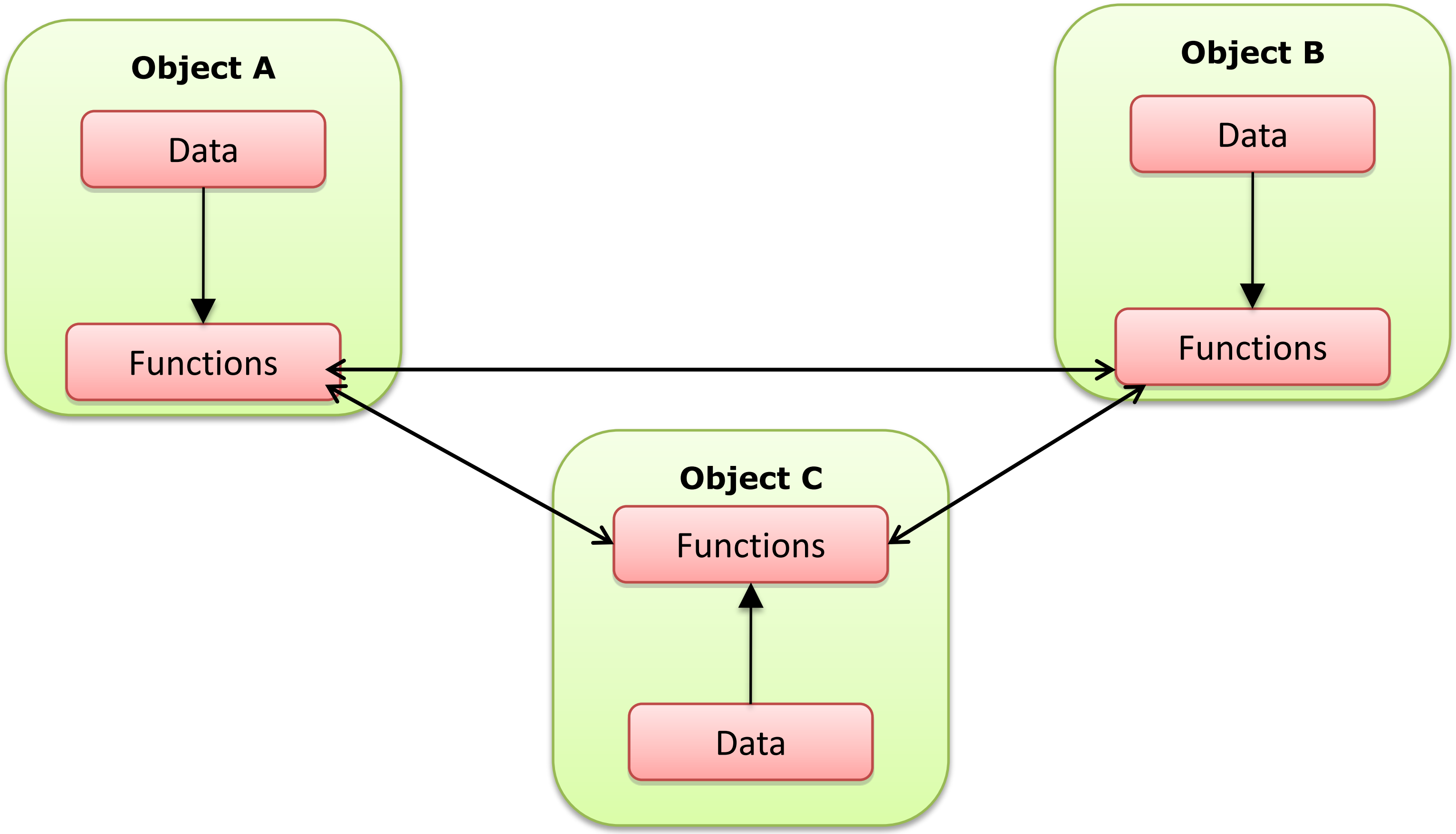
- What are the disadvantages?
  - Global data are more vulnerable to an inadvertent change by a function
  - It is very difficult to identify what data is used by which function in a large program
  - In case we need to revise an external data structure, we should also revise all functions that access the data
  - Opportunity for bugs to creep in
- Large programs are divided into smaller programs known as functions
- Most of the functions share global data
- Data move openly around the system from function to function

What is Object-Oriented Programming(OOP)

- Treats data as a critical element in the program development
- Does not allow it to flow freely around the system
- Ties data more closely to the functions that operate on it
- Protects it from accidental modifications from outside functions
- OOP allows us to decompose a problem
  - Into a number of entities called objects
  - Builds data and functions around those entities

Object-Oriented Programming(Cont)

- Data of an object can be accessed only by the functions associated with that object
- Functions of one object can access the functions of other objects



Object-Oriented Programming(Cont)

- Striking features of object-oriented programming
  - Emphasis on *data* rather than *procedure*
  - Programs are divided into what are known as *objects*
  - Data structure are designed such that they characterize the *objects*
  - Functions that operate on the data of an object are tied together in the data structure
  - Data is *hidden* and cannot be accessed by external functions
  - Objects may communicate through each other through functions
  - *New data* and *functions* can be easily added whenever necessary

What is an Object?

- Objects contain data and code to manipulate the data
- The entire set of data and code of an object can be made a *user-defined data type* with the help of a *class*
- Objects are *variable of type class*
- Each object is associated with data of type class
- Once a class has been defined, we can create any number of objects belonging to that class
  - A class is thus a collection of objects of similar types
  - For example, *mango, apple* & *orange* are member of the class *fruit*
  - If fruit has been defined as a class, then
    - *fruit mango*; will create an *object mango* belonging to the *class fruit*.

C++ Program: Prints a Line of Text

- Simple program that prints a line of text.

```cpp
// Fig. 2.1: fig02_01.cpp
// Text-printing program.
#include <iostream> // allows program to output data to the screen

// function main begins program execution
int main()
{
   std::cout << "Welcome to C++!\n"; // display message

   return 0; // indicate that program ended successfully
} // end function main
```

```
Welcome to C++!
```

Fig. 2.1 | Text-printing program.

First Program in C++: Printing a Line of Text (Cont.)

- A preprocessor directive is a message to the C++ preprocessor.
- Lines that begin with # are processed by the preprocessor before the program is compiled.
- #include <iostream> notifies the preprocessor to include in the program the contents of the input/output stream header file <iostream>.
    - Must be included for any program that outputs data to the screen or inputs data from the keyboard using C++-style stream input/output.

First Program in C++: Printing a Line of Text (Cont.)

- When a cout statement executes, it sends a stream of characters to the standard output stream object
  - std::cout; normally “connected” to the screen.

- The std:: before cout is required when we use names that we’ve brought into the program by the preprocessor directive #include <iostream>.
  - The notation std::cout specifies that we are using a name, in this case cout, that belongs to “namespace” std.
  - The names cin (the standard input stream) and cerr (the standard error stream) also belong to namespace std.

- The << operator is referred to as the stream insertion operator.
  - The value to the operator’s right, the right operand, is inserted in the output stream.

Another C++ Program: Adding Integers

- The input stream object std::cin and the stream extraction operator, >>, can be used to obtain data from the user at the keyboard.

C++ Program: Adding Integers

```cpp
// Fig. 2.5: fig02_05.cpp
// Addition program that displays the sum of two integers.
#include <iostream> // allows program to perform input and output

// function main begins program execution
int main()
{
   // variable declarations
   int number1; // first integer to add
   int number2; // second integer to add
   int sum; // sum of number1 and number2

   std::cout << "Enter first integer: "; // prompt user for data
   std::cin >> number1; // read first integer from user into number1

   std::cout << "Enter second integer: "; // prompt user for data
   std::cin >> number2; // read second integer from user into number2

   sum = number1 + number2; // add the numbers; store result in sum

   std::cout << "Sum is " << sum << std::endl; // display sum; end line
} // end function main
```

Fig. 2.5 | Addition program that displays the sum of two integers entered at the keyboard. (Part 1 of 2.)

Another C++ Program: Adding Integers (Cont.)

- A prompt directs the user to take a specific action.
- A `cin` statement uses the input stream object `cin` (of namespace `std`)
  - the stream extraction operator, `>>`, is used to obtain a value from the keyboard.
- Using the stream extraction operator with `std::cin` takes character input from the standard input stream (the keyboard).

Another C++ Program: Adding Integers (Cont.)

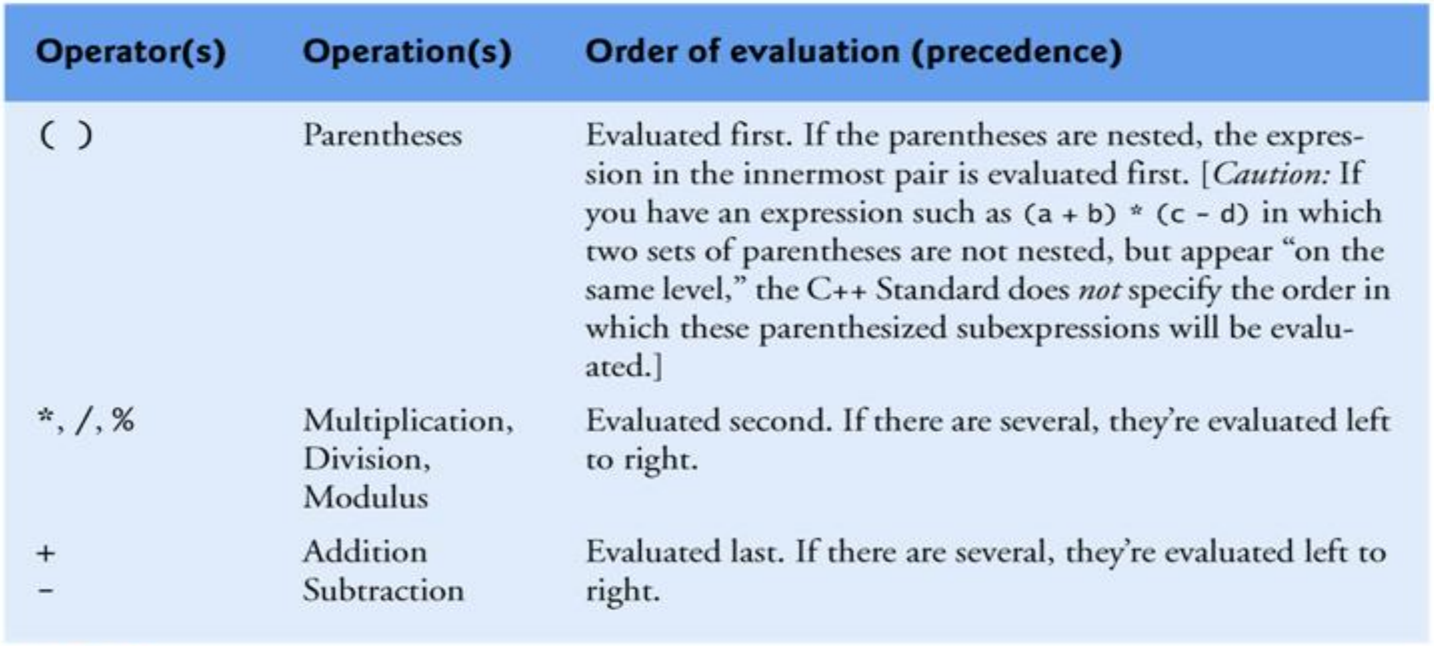
- `std::endl` is a so-called stream manipulator.
- The name `endl` is an abbreviation for “end line”
  - belongs to namespace `std`.
- The `std::endl` stream manipulator outputs a newline, then “flushes the output buffer.”
  - This simply means that, on some systems where outputs accumulate in the machine until there are enough to “make it worthwhile” to display them on the screen, `std::endl` forces any accumulated outputs to be displayed at that moment.
  - This can be important when the outputs are prompting the user for an action, such as entering data.

Another C++ Program: Adding Integers (Cont.)

- Using multiple stream insertion operators (<<) in a single statement is referred to as concatenating, chaining or cascading stream insertion operations.
- Calculations can also be performed in output statements.

Arithmetic

- C++ applies the operators in arithmetic expressions in a precise sequence determined by the following rules of operator precedence
  - generally the same as those followed in algebra.



Operator(s)	Operation(s)	Order of evaluation (precedence)
( )	Parentheses	Evaluated first. If the parentheses are nested, the expression in the innermost pair is evaluated first. [*Caution:* If you have an expression such as `(a + b) * (c - d)` in which two sets of parentheses are not nested, but appear "on the same level," the C++ Standard does *not* specify the order in which these parenthesized subexpressions will be evaluated.]
*, /, %	Multiplication, Division, Modulus	Evaluated second. If there are several, they're evaluated left to right.
+ -	Addition Subtraction	Evaluated last. If there are several, they're evaluated left to right.

Fig. 2.10 | Precedence of arithmetic operators.

Another C++ Program

```
// Fig. 2.13: fig02_13.cpp
// Comparing integers using if statements, relational operators
// and equality operators.
#include <iostream> // allows program to perform input and output

using std::cout; // program uses cout
using std::cin; // program uses cin
using std::endl; // program uses endl

// function main begins program execution
int main()
{
   int number1; // first integer to compare
   int number2; // second integer to compare

```

Fig. 2.13 | Comparing integers using `if` statements, relational operators and equality operators. (Part 1 of 3.)

Another C++ Program

```
   cout << "Enter two integers to compare: "; // prompt user for data
   cin >> number1 >> number2; // read two integers from user

   if ( number1 == number2 )
      cout << number1 << " == " << number2 << endl;

   if ( number1 != number2 )
      cout << number1 << " != " << number2 << endl;

   if ( number1 < number2 )
      cout << number1 << " < " << number2 << endl;

   if ( number1 > number2 )
      cout << number1 << " > " << number2 << endl;

   if ( number1 <= number2 )
      cout << number1 << " <= " << number2 << endl;

   if ( number1 >= number2 )
      cout << number1 << " >= " << number2 << endl;
} // end function main
```

Fig. 2.13 | Comparing integers using `if` statements, relational operators and equality operators. (Part 2 of 3.)

Using namespace

- using declarations
  - eliminate the need to repeat the `std::` prefix as we did in earlier programs.

- Once we insert these `using` declarations, we can write
  - `cout` instead of `std::cout`,
  - `cin` instead of `std::cin` and
  - `endl` instead of `std::endl`

- Many programmers prefer to use the declaration `using namespace std;`
  - enables a program to use all the names in any standard C++ header file (such as `<iostream>`) that a program might include.

Conclusion

- What is an object-oriented programming?
- Using namespace
- A basic program in C++ using `cin` and `cout`