Lect 03

Today's Outline

- Introduction to the IDE
- Intro to basic program structure in C++
- Header files
- IO functions
- Review

The Integrated Development Environment (IDE)

- An IDE provides many features to ease programming (e.g. C/C++/Java)
- An IDE have
  - Editor
  - Debugger
  - Source Control
  - …

C++ Integrated Development Environment (IDE)

- C++ systems consist of three parts:
  - a program development environment,
  - the language and
  - the C++ Standard Library (rich collections of existing classes and functions)

51

Characteristics of C++

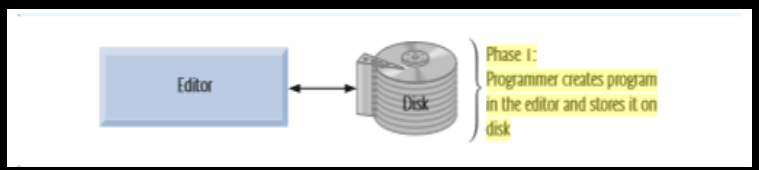
- C++ is not a purely OOP language but a hybrid that contains the functionality of the C programming language.
- Thus, C++ is a language can be used for two methods of writing computer programs:
  - procedural programming and
  - object-oriented programming.

Phases of a Program

- C++ programs typically go through six phases:
  - edit,
  - preprocess,
  - compile,
  - link,
  - load and
  - execute

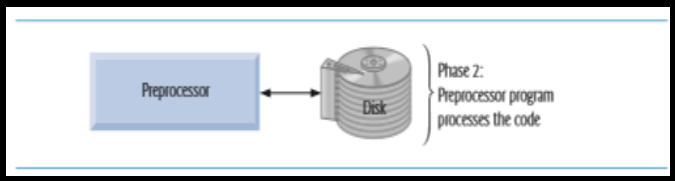
Phase 1: Creating aProgram

- C++ program(source code) can be written using an editor
- C++ source code file names often end with the .cpp, .cxx, .cc or .C extensions(note that C is in uppercase)
- Popular IDEs include Microsoft® Visual Studio 2010 Express Edition,Dev C++, NetBeans, Eclipse and CodeLite.



Phase 2: Preprocessing a C++ Program

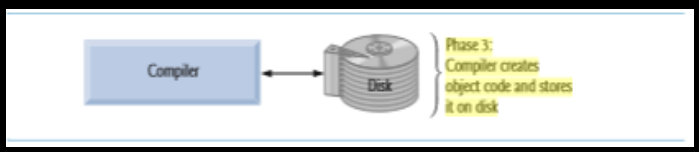
- A preprocess program executes automatically before the compiler's translation phase begins
- The C++preprocessor obeys commands called preprocessor directives, which indicate that certain manipulations are to be performed on the program before compilation.



55

Phase3:Compiling a C++ Program

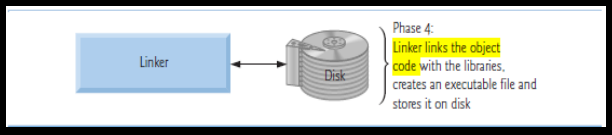
- In Phase 3, the compiler translates the C++ program into machine-language code—also referred to as object code.



56

Phase4:Linking

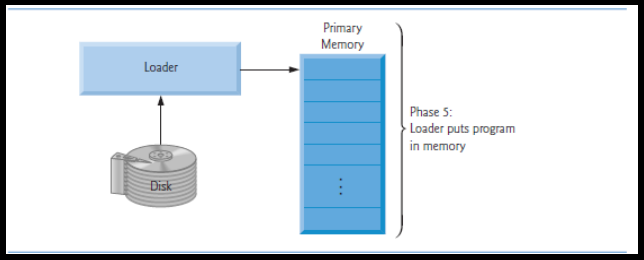
- C++ programs contain references to functions and data defined elsewhere, such as in the standard libraries or in the private libraries.
- The object code produced by the C++ compiler typically contains "holes" due to these missing parts.
- A linker links the object code with the code for the missing functions to produce an executable program (with no missing pieces).
- If the program compiles and links correctly, an executable image is produced.



57

Phase5: Loading

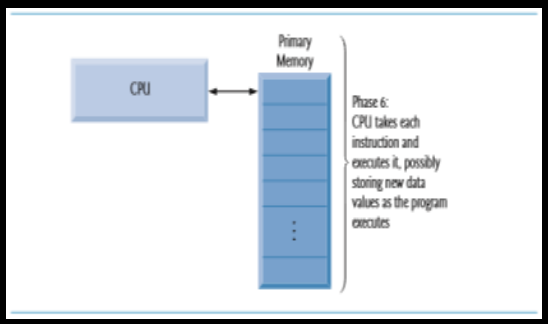
- Before a program can be executed, it must first be placed in memory. This is done by the loader, which takes the executable image from disk and transfers it to memory.
- Additional components from shared libraries that support the program are also loaded.



58

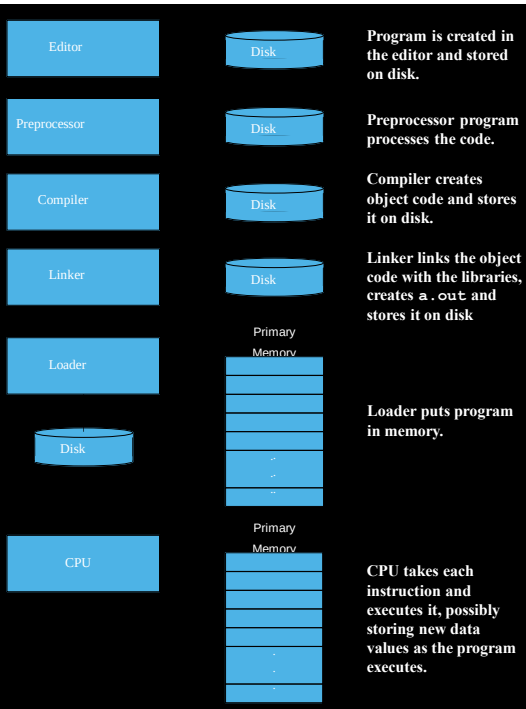
Phase6: Execution

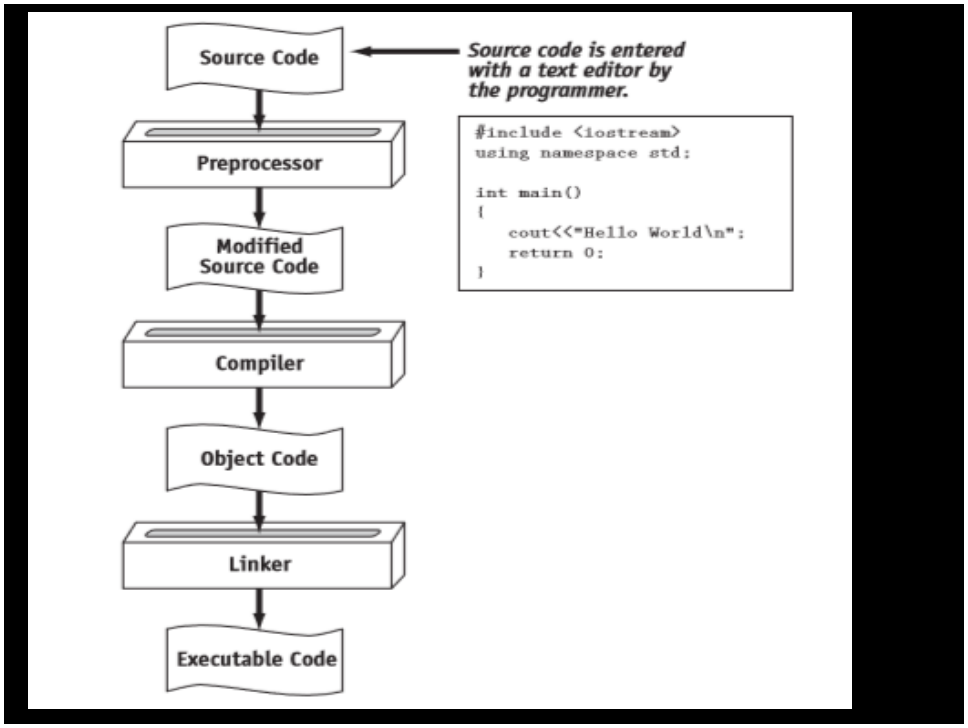
- Finally, the computer, under the control of its CPU, executes the program one instruction at a time.



59

Phases of C++ Environment





My First Program in C++

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

int main()
{
   cout <<  "Enjoy yourself with C++!"  << endl;
   return 0;
}
```

Screen output:

Enjoy yourself with C++!

Preprocessor Directive- #include

- #include <iostream>
- #include is a preprocessor directive, which is a message to the C++ preprocessor
- Lines that begin with # are processed by the preprocessor before the program is compiled

63

Header File iostream

- This line notifies the preprocessor to include in the program the contents of the input/output stream header <iostream>.
- This header must be included for any program that outputs data to the screen or inputs data from the keyboard using C++'s stream input/output.
- Two variants
  - <header>
  - "myHeader"

Main Function

- int main() is a part of every C++ program. The parentheses after main indicate that main is a program building block called a function.
- Exactly one function in every program must be named main.
- It return an integer.

65

Namespace std

- A namespace is a collection of name definitions.
- One name, such as a function name, can be given different definitions in two namespaces.
- A program can then use one of these namespaces in one place and the other in another location.
  - using std::cin;
  - using std::cout;
  - using std::endl;
- Or once define at the top
  - using namespace std;
- Namespace
  - **`std::`** specifies using name that belongs to “namespace” **`std`**
  - **`std::`** removed through use of **`using`** statements

IO Functions

- Input/output
  - **`cin`**
    - Standard input stream
    - Normally keyboard
  - **`cout`**
    - Standard output stream
    - Normally computer screen

IO Functions (Contd.)

- Standard Input stream object
  - **>>** (stream extraction operator)
    - Used with **`std::cin`**
    - Waits for user to input value, then press *Enter* (Return) key
    - Stores value in variable to right of operator

IO Functions (Contd.)

- Standard output stream object
  - **`std::cout`**
  - "Connected" to screen
  - **`<<`**
    - Stream insertion operator
    - Value to right (right operand) inserted into output stream

69

return0 Statement

- The return value of 0 indicates normal termination; while non-zero (typically 1) indicates abnormal termination.
- C++ compiler will automatically insert a return 0 at the end of the main () function, if this statement is omitted.

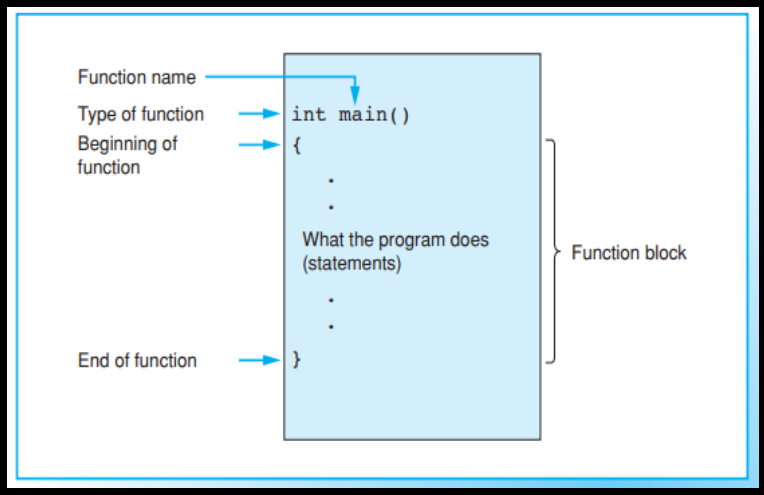
Comments

- Document programs
- Improve program readability
- Ignored by compiler
- Single-line comment
  - Begin with //

Another simple program. What does it do?

```
#include <iostream.h>

int main ( )
{
   double x, y, z, avg;
   cout << "Enter 3 values: ";
   cin >> x >> y >> z;
   avg = ( x + y + z ) / 3;
   cout << "The average of three arbitrary values is "
        << avg;
   return 0;
}
```

Structure of function main()



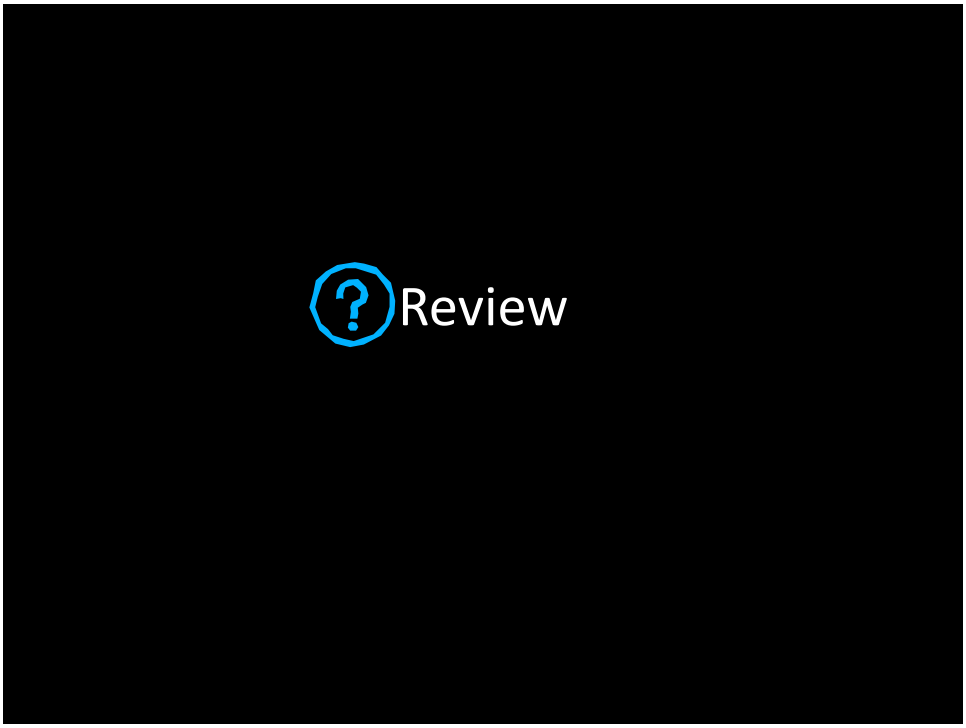

```
/*************************************************************
  A program with some functions and comments
*************************************************************/

#include <iostream>
using namespace std;

void line(), message();              // Prototypes

int main()
{
    cout << "Hello! The program starts in main()."
         << endl;
    line();
    message();
    line();
    cout << "At the end of main()." << endl;

    return 0;
}

void line()                          // To draw a line.
{
    cout << "---------------------------------" << endl;
}

void message()                 // To display a message.
{
    cout << "In function message()." << endl;
}
```

Question?

- What does ***#include <iostream>*** mean?
- What are the rest of the lines in the program for?
- Why ***return 0***?
- What are ***cin*** and ***cout***?
- What is ***>>***?
- What is ***<<***?
- Are the ordering of lines (instructions) in the program important?
- Can one use u, v, and w in place of x, y, and z?

Your Turn

- Write a C++ program that outputs the following text on screen: Oh what a happy day! Oh yes, what a happy day! Use the manipulator endl where appropriate.


Review