AIOU Stat Deptt

Computer Programming C/C++ (1564)

Zahoor Ahmad

Part : 1

What is this course about?

- Programming Course using C++
- What does good programming involve?
  - Software engineering, structured programming
  - Planning
  - Writing clear, well documented, and well formatted code
  - Clear modularity – clear sections of code doing their job

Computer Programming Languages

- Programmers write programs/instructions in various programming languages – some easier for the computer to understand and some easier for the programmer to understand.
  - Machine languages
  - Assembly languages
  - High-level languages

Machine language

- Only language computer directly understands
- “Natural language” of computer
- Defined by hardware design
  - Machine-dependent
- Generally consist of strings of numbers
  - Ultimately 0s and 1s
- Instruct computers to perform elementary operations
  - One at a time
- Cumbersome for humans

Assembly language

- English-like abbreviations representing elementary computer operations
- Clearer to humans
- Incomprehensible to computers
  - Translator programs (assemblers)
    - Convert to machine language
- Example:

```
LOAD   BASEPAY
ADD    OVERPAY
STORE  GROSSPAY
```

High-level languages

- Similar to everyday English, use common mathematical notations
- Single statements accomplish substantial tasks
  - Assembly language requires many instructions to accomplish simple tasks
- Translator programs (compilers)
  - Convert to machine language
- Interpreter programs
  - Directly execute high-level language programs
- Example:

```
grossPay = basePay + overTimePay
```

Interpreter versus Compiler

Interpreter

- Flexible
- More interactive
- More dynamic behavior
- Rapid development
- Can run program immediately after writing or changing it
- Portable to any machine that has the interpreter

Compiler

- More efficient execution
- Extensive data checking
- More structured
- Usually more scalable (can develop large applications)
- Must (re-)compile program each time a change is made
- Must recompile for new hardware or OS

Background on C++

- One of the most popular software development languages
- Superset of the C language (with object oriented features)
- Be Careful!
  - Does not enforce structured style (e.g., array out of bounds not checked)
  - Gives a lot of control to the programmer
  - Programmer must be responsible for enforcing discipline

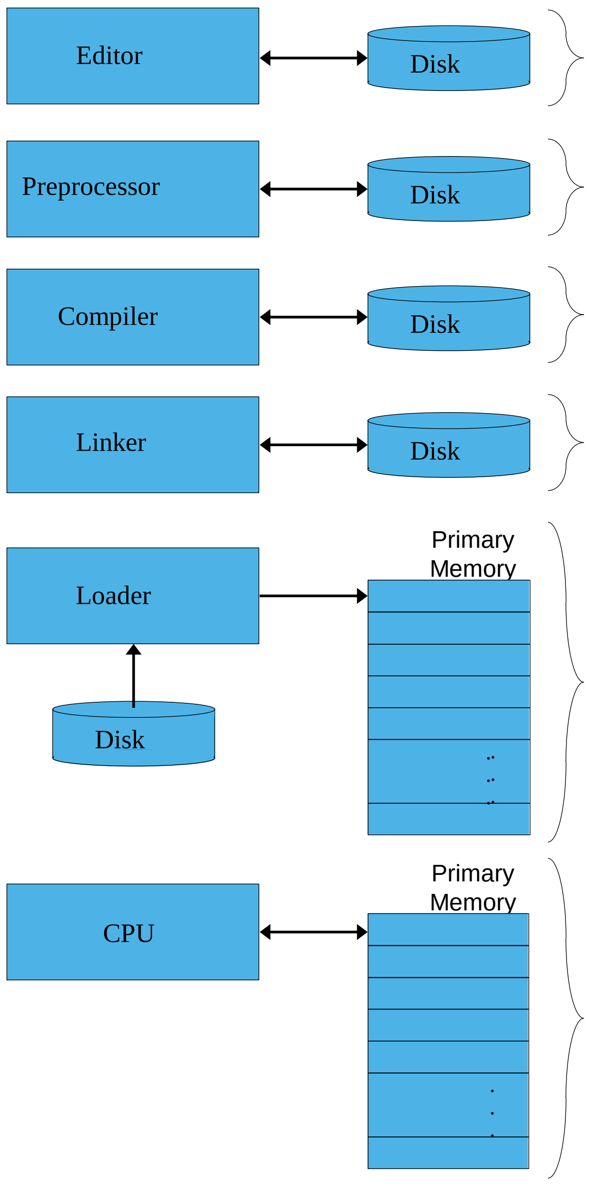
Basics of a Typical C++ Environment

- C++ systems
  - Program-development environment
  - Language
  - C++ Standard Library

Basics of a Typical C++ Environment

- Phases of C++ Programs:

1. Edit
2. Preprocess
3. Compile
4. Link
5. Load
6. Execute



Program is created in the editor and stored on disk.

Preprocessor program processes the code.

Compiler creates object code and stores it on disk.

Linker links the object code with the libraries, creates `a.out` and stores it on disk

Loader puts program in memory.

CPU takes each instruction and executes it, possibly storing new data values as the program executes.

Introduction to C++ Programming

- C++ language
  - Facilitates structured and disciplined approach to computer program design
- Following several examples
  - Illustrate many important features of C++
  - Each analyzed one statement at a time
- Structured programming
- Object-oriented programming

Comments & Preprocessor Directive

- Comments
  - Document programs
  - Improve program readability
  - Ignored by compiler
  - Single-line comment
    - Begin with `//`
  - Multiple-line comment
    - Begin with `/*` and end with `*/`
- Preprocessor directives
  - Processed by preprocessor before compiling
  - Begin with **#**

```
// Fig. 1.2: fig01_02.cpp
// A first program in C++.

#include <iostream>

// function main begins program execution

 int main()

{

    std::cout << "Welcome to C++!\n";

   return 0;   // indicate that program ended successfully

} // end function main
```

Single-line comments.

Preprocessor directive to include input/output stream header file **`<iostream>`**.

Left brace **{** begins function body.

Function **`main`** appears exactly once in every C++ program..

Statements end with a semicolon **;**.

Name **`cout`** belongs to namespace **`std`**.

Stream insertion operator.

Keyword **`return`** is one of several means to exit function; value **0** indicates program terminated successfully.

Corresponding right brace **}** ends function body.

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

Function **`main`** returns an integer value.

Basic Concepts

- Standard output stream object
  - **`std::cout`**
  - “Connected” to screen
  - **`<<`**
    - Stream insertion operator
    - Value to right (right operand) inserted into output stream
- Namespace
  - **`std::`** specifies using name that belongs to “namespace” **`std`**
  - **`std::`** removed through use of **`using`** statements
- Escape characters
  - **`\`**
  - Indicates “special” character output

Escape Sequences

Escape Sequence	Description
`\n`	Newline. Position the screen cursor to the beginning of the next line.
`\t`	Horizontal tab. Move the screen cursor to the next tab stop.
`\r`	Carriage return. Position the screen cursor to the beginning of the current line; do not advance to the next line.
`\a`	Alert. Sound the system bell.
`\\`	Backslash. Used to print a backslash character.
`\"`	Double quote. Used to print a double quote character.

A simple Program

```cpp
// Fig. 1.4: fig01_04.cpp
// Printing a line with multiple statements.
#include <iostream>

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

   return 0;   // indicate that program ended successfully

} // end function main
```

Multiple stream insertion statements produce one line of output.

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

A simple Program

```cpp
// Fig. 1.5: fig01_05.cpp
// Printing multiple lines with a single statement
#include <iostream>
Using namespace std;
// function main begins program execution
int main()
{
   cout << "Welcome\nto\n\nC++!\n";

   return 0;   // indicate that program ended successfully

} // end function main
```

Using newline characters to print on multiple lines.

```
Welcome
to

C++!
```