CS 102 Introduction to Programming Using C++

Chapter 1

Introduction

The Textbook

- It's important to read the book
- The front of the book features a table of contents
- It shows several important items for each chapter
  - Common Errors
  - Worked Examples
  - Programming Tips
- You should check these out for each chapter

What is Programming?

- What is a program?
- Name some examples of programs

Hardware and Software

- What is hardware?
- What is software?

Important Hardware Items—The CPU

- It executes instructions
- Some instructions are
  - Making decisions
  - Doing arithmetic
  - Getting data from memory, external devices
  - Modifying that data
  - Storing data back into memory, external devices
  - Moving to another part of the instructions

More Hardware (Storage)

- Primary Storage
  - Also called memory
  - Just circuits (chips)
  - Is volatile (temporary)
  - It's where the action is

- Secondary Storage
  - Examples
  - Hard disk
  - Flash drive
  - Data stored there is permanent

Programming Language Ladder

- High Level
  - Close to how we think
  - C++, Java
- Low Level
  - Close to how the hardware operates
  - Machine language, Assembly language
- We keep trying to climb higher
  - Why?

Translating C++ to Machine Language

- C++ is a high level language
- It needs to be translated into machine language
    - The computer stores and understands only 0 and 1
- The translator is called a compiler

Portable Languages

- A portable language is one that compiles on any machine
- C++ is mostly a portable language
  - The basic part is portable
  - Extensions, like graphics, may not be portable
    - In the computer world, the word graphics refers to pictures

Creating C++ Programs

- We say you write or create programs
- Writing a program means typing, typing, and typing!
  - We use an editor
  - It’s a very primitive word processor that doesn’t do boldfacing, font style or size, has no graphics, etc.
- Spelling counts!
- Punctuation counts!
- Indentation counts!

Our Programs

- We will start out with very mild ambitions
- We will write programs that print answers to questions and calculations
- We will build up to some sophisticated programming concepts
- We will write some GUI code if we have time
    - GUI stands for Graphical User Interface
    - It means you click on buttons, move icons, etc.

IDEs

- You can create your program using an IDE
    - IDE stands for Integrated Development Environment
- An IDE is a development environment
    - It helps you develop programs
    - It contains an editor
- It's integrated
    - It contains an editor
    - It contains a compiler
    - It may also contain a debugger

A Sample Program

```
#include <iostream>
using namespace std;
int main ()
{
    cout << “My first program” << endl;
    return 0;
}
```

Writing a C++ Program

- I have numbered the lines for discussion
  - Do not number the lines in your own programs
- Some general ideas about C++
- 1. C++ is case-sensitive
  - This means that, for example, the line   int main () must have main in all lower case letters
- 2. Every line ends with a semi-colon
- 3. The braces come in pairs
- 4. You can (and should!) skip lines in your program

More Program Analysis

- Line 1. #include statements tell the compiler where to find various commands
    - Later we will call these commands functions
- Line 2. Using namespaces allows you to build large projects in small pieces
    - We will always type this line just as it is

Still More Program Analysis

- Line 3. main is the name of the function where you put the statements of your program
  - Here is where you have print statements, calculations, etc.
  - Your program will do everything you put into main()
  - main () is an example of a function
- Lines 4 and 7. The braces are a pair.
  - They must be lined up
  - They form the limits of the main() function
  - We indent everything between them

Even More Program Analysis

- Line 5. cout is a print statement
  - It prints whatever is on the right side of the <<
  - Think of cout as the screen and << as an arrow
- Line 6. This indicates your program completed normally and that no errors occurred

Getting Your Program to Run

- You just type in your program
- We call your program source or source code
- You save it in a file
  - This should be a .cpp file
- You compile it
  - I will use the tdm-gcc compiler
    - You can search for it and download it
  - The compiler typically generates a.out (a.exe on Windows)
- You run the result of the compile process

Behind the Scenes

- The actual steps were these
- 1. You stored your program in a file
- 2. You submitted your source file to the compiler
  - The compiler created machine code from your source
- 3. The machine code was submitted to the linker
  - The linker combined needed library files with your machine code
  - The linker created a program that you can run
    - We call this an executable

Errors

- It’s very easy to write a program with errors
- There are two kinds of errors
  - Syntax errors
    - These errors are due to missing semicolons, misspelled words, mismatched braces, etc.
    - Your program won’t compile if it has syntax errors
    - The gcc compiler tells the line number and position of the error
  - Logic errors
    - These errors occur when your program doesn’t do what you expect

Algorithms

- An algorithm is a step-by-step recipe to do something
  - You can think of a recipe as an algorithm
  - Giving directions is an algorithm
  - Many things we do in life involve algorithms
- The challenge in programming is breaking a task down into a step-by-step algorithm
  - It's easy for us to understand and do, but gets complicated when we try to analyze it

The Software Development Process

- The Steps
- 1. Understand the problem
  - Test the algorithm with sample data
- 2. Develop an algorithm that solves the problem
- 3. Translate the algorithm to C++
- 4. Compile and Test the program

- If the program doesn't do what it should, go back to step 2

Homework

- These problems are from p. 23+
- Do R1.1, R1.2, R1.3, R1.4, R1.5, R1.6, R1.7, R1.10, R1.11, R1.12, R1.13, R1.14
- These problems are due next Tuesday

- In the upper right corner, after your name, add
  - The chapter the problems are from
  - The page number the problems are from
  - A list of the problems themselves

Our First Programming Assignment

- Write a program that solves P1.6, P1.7, P1.8, P1.9, P1.10, or P1.11
- Be sure to follow the directions closely
- When writing your program, start with these four+ lines at the beginning of your program

```
// Your name
// The date you wrote the program
// Which lab this is (Lab 1 in this case)
// A brief description of the program
```

Lines Starting with //

- The previous slide says to put four extra lines into your program
- Those lines are comments
  - The point of a comment is to explain code
    - In this case, the explanation is more about the code in general
    - Usually, comments explain routines or tricky code
  - Comments have no rules
    - You can write in “plain English”

Academic Honesty

- This is the academic honesty policy for our class:
  - All programs must be written by you alone
- Exceptions
  - Since this is a class, you may base your programs on programs in the textbook
    - If you do that, you must cite the textbook
  - You can get help from me

Academic Honesty--Details

- Other than that, if a program is not completely your own work, you will receive a zero (0) for it
- The phrase "your own work" means
  - That you wrote the program yourself
  - That you wrote it without coding help from anyone except me
  - That you did not download the program or any part of it from the internet
- Remember that I can ask you to explain any part or parts of your program

Questions?

- Are there any questions?