Overview of Computers and Programming

Main Components of a computer

- CPU - Central Processing Unit: The "brain" of the computer
  - ISA - Instruction Set Architecture: the specific set of low-level instructions available to a CPU. Differs for various CPU types (Intel Pentium, Mac G4, etc)
- ALU - Arithmetic & Logic Unit
  - responsible for performing arithmetic calculations, as well as logical operations (comparisons for equality, inequality, for instance).
- Main Memory (RAM - Random Access Memory)
  - storage close to CPU
  - Faster to access than hard disk
  - stores executing programs and data being currently worked on
- Secondary Memory
  - hard disk, floppy disk, CD, DVD, etc.
- Input devices
  - mouse, keyboard, scanner, network card, etc.
- Output devices
  - screen/console, printer, network card, etc.
- Operating System
  - Examples: Mac OS, Windows XP, Linux
  - Controls computer operations
  - Manages allocation of resources for currently running applications

Memory Concepts

- **bit**: a binary digit
  - Stores the value 0 or 1
  - Smallest unit of storage in a computer
- **byte**: 8 bits
  - Smallest *addressable* unit of storage in a computer
  - Storage units (variables) in a program are 1 or more bytes
  - Each byte in memory has an *address* (a number that identifies the location)

Programming, and Programming Languages

- **Program** - a set of instructions for a computer to execute
- Evolution of Programming languages
  - **Machine Language**
    - Based on machine's core instruction set
    - Needed by computer, hard for humans to read (1's and 0's)
    - Example: 1110110101010110001101010

        - Say what?
    - **Assembly Language**
        - translation of machine instructions to symbols, slightly easier for humans to read
        - Example: `ADD $R1, $R2, $R3`
            - Well, we know it has something to do with addition!
    - **High-level procedural languages**
        - Abstraction of concepts into more human-readable terms
        - Closer to "natural language" (i.e. what we speak)
        - Easy to write and design, but must be translated for computer
        - Examples include C, Pascal, Fortran
    - **Object-oriented languages**
        - Abstraction taken farther than procedural languages
        - Objects model real-world objects, not only storing data (attributes), but having inherent behaviors (operations, functions)
        - Easier to design and write good, portable, maintainable code
        - Examples include Smalltalk, C++, Java
- Bridging the gap between high-level code and machine code
    - **Interpreted languages** -- source code is directly run on an interpreter, a program that runs the code statements
    - **Compiled Languages**
        - A **compiler** program translates source code (what the programmer writes) to machine language (*object code*)
        - A **linker** program puts various object code files together into an executable program (or other target type, like a DLL)
        - C and C++ are compiled languages

Building Programs

Software development

Involves more than just writing code

- Analysis and problem definition
- Design - includes design of program or system structure, algorithms, user-interfaces, and more
- Implementation (coding)
- Testing - can be done during design, during implementation, and after implementation
- Maintenance - usually the major cost of a software system. Not part of "development", but definitely part of the software life cycle

Basic Creation and Execution of a C++ program

1. Create **source code** with a text editor, store to disk
    - Source code is just a plain text file, usually given a filename extension to identify the programming language (like .c for C, or .cpp for C++)
2. **Preprocessor** -- Part of compiler process, performs any pre-processing tasks on source code
3. **Compilation** -- syntax checking, creation of *object code*

- Object code is the machine code translation of the source code

4. **Linking** -- Final stage of the creation of an executable program. Linking of object code files together with any necessary libraries (also already compiled)
5. Execution of program
    - Program loaded into memory, usually RAM
    - CPU executes code instructions

Integrated Development Environments

- An Integrated Development Environment (IDE) is a software package that includes all necessary tools for creating a program. Typcially includes:
    - Text editor
    - Compiler
    - Linker
    - Debugger
    - ability to launch and run applications from within IDE environment
- Not every compiler is an IDE, but pretty much any compiler software includes: preprocessor, compiler, linker
- Examples of C++ IDEs
    - Microsoft Visual C++ 6.0
    - Metrowerks CodeWarrior
    - Borland Builder

Programming is about Problem Solving

- **Algorithm** - a finite sequence of steps to perform a specific task
    - To solve a problem, you have to come up with the necessary step-by-step process before you can code it
    - This is often the trickiest part of programming
- Some useful tools and techniques for formulating an algorithm
    - **Top-down Refinement**: Decomposing a task into smaller and simpler steps, then breaking down each step into smaller steps, etc
    - **Pseudocode**: Writing algorithms informally in a mixture of natural language and general types of code statements
    - **Flowcharting**: If you can visualize it, it's often easier to follow and understand!
- **Testing** - algorithms must also be tested!
    - Does it do what is required?
    - Does it handle all possible situations?
- **Syntax vs. Semantics**
    - **Syntax** -- the *grammar* of a language.
      A syntax error: "I is a programmer."
    - **Semantics** -- the *meaning* of language constructs
      Correct syntax, but a semantic error: "The car ate the lemur."

A sample C++ program

```
#include <iostream>                       /* pre-processor directive */
using namespace std;

int main()                                // main function, start of program
{
  int year = 2015;                        // an integer variable declaration
  char entry;                             // a character variable

  cout << "Welcome to C++ Programming" << year << '\n';
  cout << "Enter 'x' to quit: ";
  cin >> entry;                           // read input from user

  cout << "Goodbye!\n";
  return 0;                               // return value to operating system
}
```

A link to this code file

- A C++ program has exactly one `main()` function block
  - Format is:

    ```
    int main()
    {
      // code statements here

      return 0;
    }
    ```

- The lines that look like these are known as "comments":

  ```
  /* This is a block comment */
  // This is a line comment
  ```

  They allow users to document code by writing explanatory notes inside. The compiler ignores all comments. Comments are for people.

- A C++ program will usually make use of external libraries of code
  - `iostream` library used here ("input/output stream" library)
  - brought in with the `#include` directive
  - `iostream` has definitions of the objects **`cout`** and **`cin`**
  - The statement `using namespace std;` denotes that we are using items from the "standard namespace".
    In this case, it involves items from the `iostream` library
- Declarations of variables (variables must be declared before they are used)
  - `int year = 2015;`
    `char entry;`
- The statements with `cout` are doing output of data to the screen
  - `cout` -- standard output stream object (think of this as a name for the monitor/console)

  - insertion operator <<
    -- inserts data into a "stream" (a data buffer) bound for the monitor
- The statement with `cin` is reading input from the keyboard
  - `cin` -- standard input stream object (think of this as a name for the keyboard)
  - extraction operator >>
    -- extracts data from the buffer of incoming data (a "stream" of inbound data) and sends it to a variable in memory