Introduction to C++

CS 16: Solving Problems with Computers I
Lecture #2

Ziad Matni
Dept. of Computer Science, UCSB

Announcements

- **Homework #1 due today**
  - Please take out any staples or paper clips

- **No more switching lab times**
  - Labs at 9am, 10am, 11am are **FULL**
  - Other labs have some space left

Lecture Outline

- Computer Systems --- A review from last week
- Programming and Problem Solving
- Introduction to C++

REVIEW

Defining Computer

A device **that can be instructed** to carry out an **arbitrary** set of **arithmetic or logical operations** automatically

REVIEW

Computer Software

- The collection of programs used by a computer, and includes:
  - Applications
  - Translators (compilers)
  - System Managers (drivers, other OS components)

REVIEW

5 Main Components to Computers

- Inputs
- Outputs
- Processor
- Main memory
  - Usually inside the computer, volatile
- Secondary memory
  - More permanent memory for mass storage of data

REVIEW

Computer Memory

- Usually organized in two parts:
  - Address
    - Where can I find my data?
  - Data (payload)
    - What is my data?

- The smallest representation of the data
  - A binary *bit* ("0"s and "1"s)
  - A common collection of bits is a byte (8 bits = 1 byte)

REVIEW

What is the Most Basic Form of Computer Language?

- Binary *a.k.a* Base-2

- Expressing data AND instructions in either "1" or "0"
  - So,

"01010101 01000011 01010011 01000010 00100001 00100001"

could mean an *instruction* to "calculate 2 + 3"

Or it could mean a *number* (856783663333)

Or it could mean a *string of 6 characters* ("UCSB!!")

So… like… what process stuff in a computer?

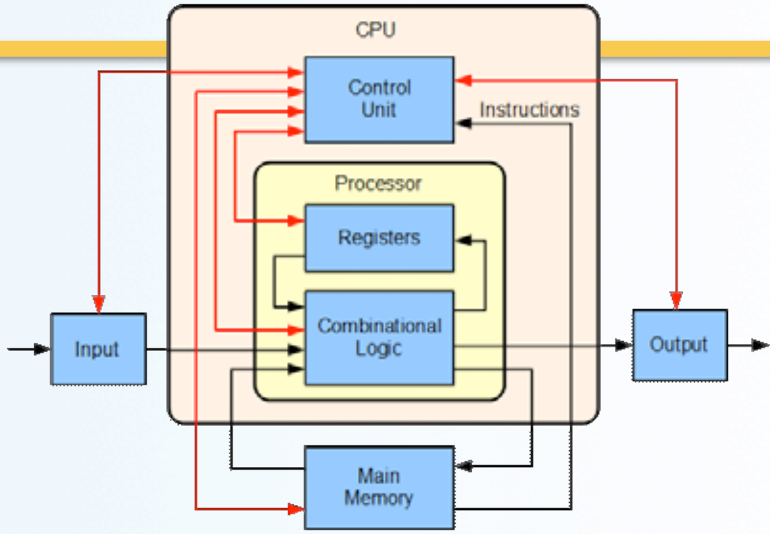
- The Central Processing Unit (CPU)
  - Executes program instructions

- Typical capabilities of CPU include:
  - Add
  - Subtract
  - Multiply
  - Divide
  - Move data from location to location

You can do just about anything with a computer with just these simple instructions

Parts of the CPU

- The CPU is made up of 2 main parts:
  - The Arithmetic Logic Unit (ALU)
  - The Control Unit (CU)



- The ALU does the calculations in binary using "registers" (small RAM) and logic circuits

- The CU handles breaking down instructions into control codes for the ALU and memory

Image from wikimedia.org

Microprocessor

A fully functional CPU with its local memory, all contained within one IC

Image from wikimedia.org

The Operating System

- Is it software?
  - Yes!
- Is it a program?
  - In a general sense, yes!
    (or more precisely, a bunch of programs acting in concert)
- What does it do?
  - Allocates the computer's resources
  - Allows us to communicate with the computer
  - Responds to user requests to run other programs

Some Common OS

MacOS ***Linux*** ***MS Windows***

Apple iOS ***Google Android*** ***Ubuntu***

Image from technologydatagroup.com

The CPU's Fetch-Execute Cycle

- **Fetch** the next instruction
- **Decode** the instruction
- **Get data** if needed
- **Execute** the instruction
- ***Why is it a cycle???***

This is what happens inside a computer interacting with a program at the "lowest" level

Computer Languages and the F-E Cycle

- Instructions get executed in the CPU in machine language (i.e. all in "1"s and "0"s)
  - Even the *smallest* of instructions, like "add 2 to 3 then multiply by 4", need *multiple* cycles of the CPU to get executed fully
  - But THAT'S OK! Because, typically, CPUs can run *many millions* of instructions per second
- In *low-level languages*, you will need to spell those cycles out
  - Most programmers nowadays do not bother with this approach
- In *high-level languages*, you won't
  - E.g. 1 statement, like "*x = c*(a + b)*" is enough to get the job done

"high level" vs. "low level" Programming

- High Level computer languages, like C++, are A LOT simpler to use!
- Uses syntax that "resembles" human language
- Easy to read and understand:

  x = c(a + b)   vs.   101000111010111*

- But, still… the CPU *NEEDS* machine language to do what it's supposed to do!
- So *SOMETHING* has to "translate" high level code into machine language…

Compilers

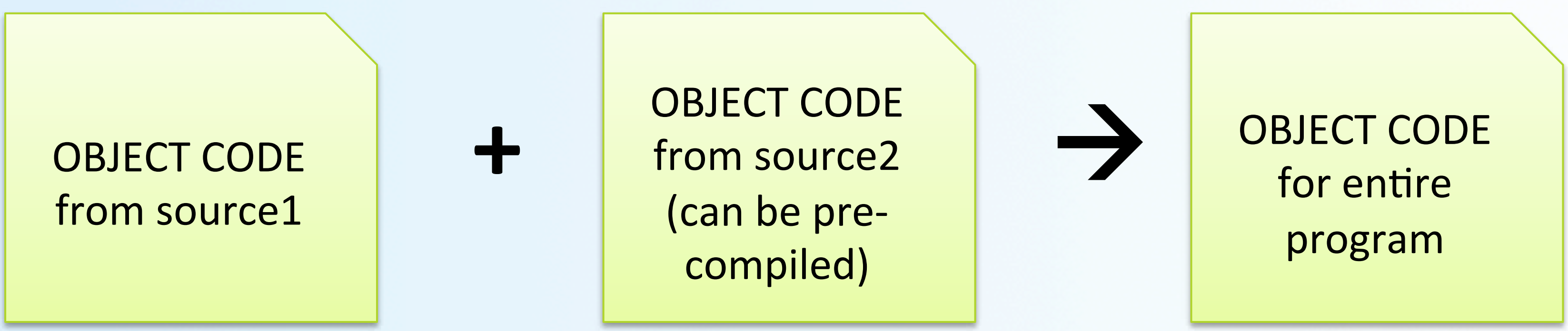
- *SOMETHING* has to "translate" high level code into machine language…
  - A program called a Compiler
  - Compilers are "language-specific"
  - In Linux/UNIX, there are several kinds like "g++" or "clang"

- Source code
  - The original program in a high level language
- Object code
  - The translated version in machine language

Linkers

- Some programs we use are already compiled
  - Their object code is available for us to use and combine with our own object code

- A Linker combines object codes



Algorithm vs. Program

- **Algorithm**
  - A sequence of precise instructions that leads to a solution
- **Program**
  - An algorithm expressed in a language the computer can understand

Some Historical Background…

The First Modern Computing Devices

Images from Wikimedia.org

B. Pascal (1623 – 1662)

"Pascaline" : a calculating machine (1652)

- Blaise Pascal
  - Mechanical device that could add, subtract, divide & multiply using gears
- Joseph Jacquard
  - Jacquard's Loom, used punched cards to describe patterns

J. Jacquard (1752 – 1834)

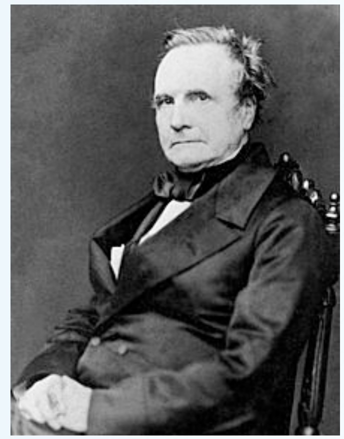
Jacquard Loom (invented 1801)

Computing Devices for General Purposes

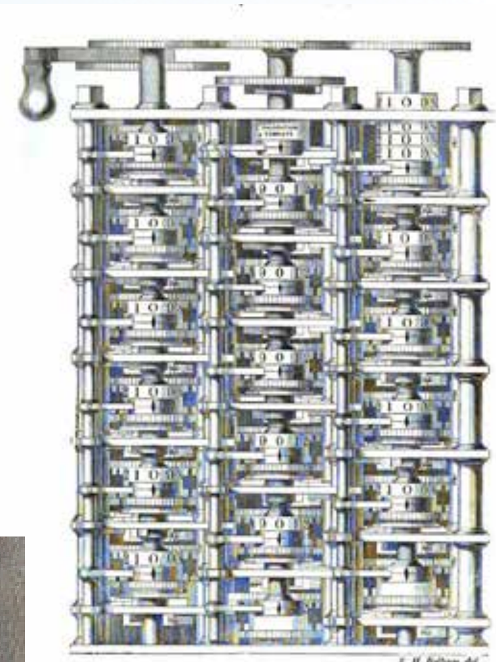
Images from Wikimedia.org

- **Charles Babbage**
  - *Analytical Engine* could calculate polynomial functions and differentials
  - Calculated results, but also *stored intermediate findings* (i.e. precursor to computer memory)
  - "Father of Computer Engineering"

- **Ada Byron Lovelace**
  - Worked with Babbage and foresaw computers doing much more than calculating numbers
  - Loops and Conditional Branching
  - "Mother of Computer Programming"



C. Babbage (1791 – 1871)



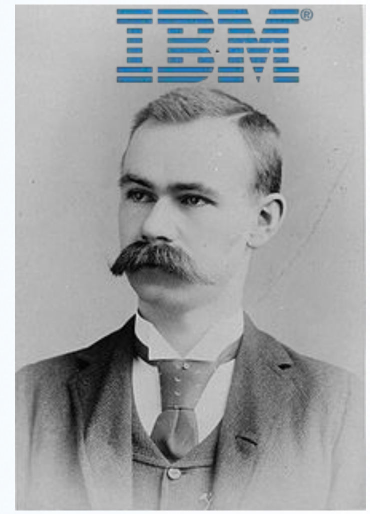
Part of Babbage's Analytical Engine

A. Byron Lovelace (1815 – 1852)

Punched Card Data Processors

Images from Wikimedia.org

- **Herman Hollerith**
  - Developed a "mechanical tabulator" in the early 1900s and used it very successfully to do the census for the US government
  - His Tabulating Machine Company (with 3 others) became **International Business Machines Corp. (*IBM*)** in 1911



H. Hollerith (1860 – 1929)

IBM punched card "Accounting Machines", pictured in 1936.

But these were all single-purpose calculating machines

The Modern Digital Computer

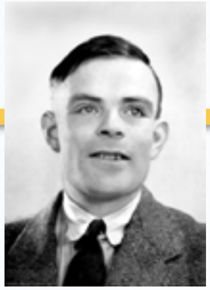
Alan Turing

A. Turing (1912 – 1954)

- Theorized the possibility of computing machines capable of performing *any* conceivable mathematical computation as long as this was representable as an *algorithm*
  - Called "*Turing Machines*" (1936)
  - Lead the effort to create a machine to successfully decipher the German "Enigma Code" during World War II

- Algorithm
  - A step-by-step set of operations to be performed to process something
  - First described in 825 AD by Al-Khawarizmi, a Persian mathematician

Turing's Legacy



- Turing Machine : An abstract model
  - Calculating machine that can "read" in symbols on a medium and "writes" out results on another, based on a "table" of instructions
  - What we call "computers" today owe a lot to this concept
- The *Turing Test* : Asks "Can Machines Think?"

  - A test to see if a machine can exhibit intelligent behavior like a human
  - Example: CAPTCHA
  - Completely Automated Public Turing test to tell Computers and Humans Apart
- The Turing Award
  - Called the "Nobel Prize" for computing
  - For contributions of lasting and major technical importance to the computer field
  - https://en.wikipedia.org/wiki/Turing_Award

Computers Since the Mid-20th Century

- ENIAC (1946) and UNIVAC (1951)
  - The 1st general purpose computers (private use and commercial use, respectively)
  - ENIAC developed by the US Army ; had a role in the development of the H-Bomb
  - UNIVAC developed by a private corporation and sold to other companies
  - Enormous machines – took up entire floors of a building

- Commercialization of the microprocessor (1960s) and personal computers (1970s and 1980s)
  - Made the hardware a lot smaller and cheaper
  - Apple I and II, Macintosh (Apple), PC (IBM)
  - Lots of software companies to run the hardware (Microsoft's DOS, Windows)

The Individual Computer Gives Way to the Network

- Invention of computer networking protocols, like *Ethernet* and *TCP/IP* (1980s)
  - Bob Metcalfe
  - Vint Cerf
- Deployment of ARPANET (later NSFNET) (1970s and 1980s)
  - Mostly just for university research use and the military
- The transition of NSFNET into the Internet (1990s)

Name These Contemporary Computer Titans

Steve Jobs & Steve Wozniak, founded Apple

Bill Gates, co-founded Microsoft

Tim Berners-Lee, invented hypertext/WWW

Vint Cerf, co-invented TCP/IP

Larry Page & Sergey Brin, invented/founded Google

Problem Solving

Problem Solving

How do you solve problems?

Understand the problem

Devise a plan

Carry out the plan

Look back and re-assess

Strategies

Ask questions!

- *What do I know about the problem?*
- *What is the information that I have to process in order the find the solution?*
- *What does the solution look like?*
- *What sort of special cases exist?*
- *How will I recognize that I have found the solution?*

Strategies

Ask questions! Never reinvent the wheel!

Similar problems come up again and again in different guises

A good programmer recognizes a task that has been solved before and plugs in the solution

However, a good programmer does not plagiarize…

Strategies

Divide and Conquer!

Break up a large problem into smaller units
and solve each smaller problem

- Applies the concept of abstraction
- The divide-and-conquer approach can be applied over and over again until each subtask is manageable

Computer Problem-Solving

Analysis and Specification Phase

- Analyze the problem
- Specify the details

Algorithm Development Phase

- Develop an algorithm
- Test your algorithm

Implementation Phase

- Code your algorithm
- Test your code

Maintenance Phase

- Use the program
- Maintain the program

Can you see a recurring theme?

Developing Software Products

- As a business product
  - Software is “made” (developed) to meet market needs
- Needs resources and **planning**
  - Software needs to be
    programmed, documented, tested, fixed/maintained
- There is a process to everything you need to do!
  - A complex task – a problem to solve – needs a plan, an algorithm

Systems Development Life Cycle (SDLC)

A structured approach to software development:

GOAL:

A software **development process** that leads to

a **high quality system** that

meets or exceeds customer expectations,

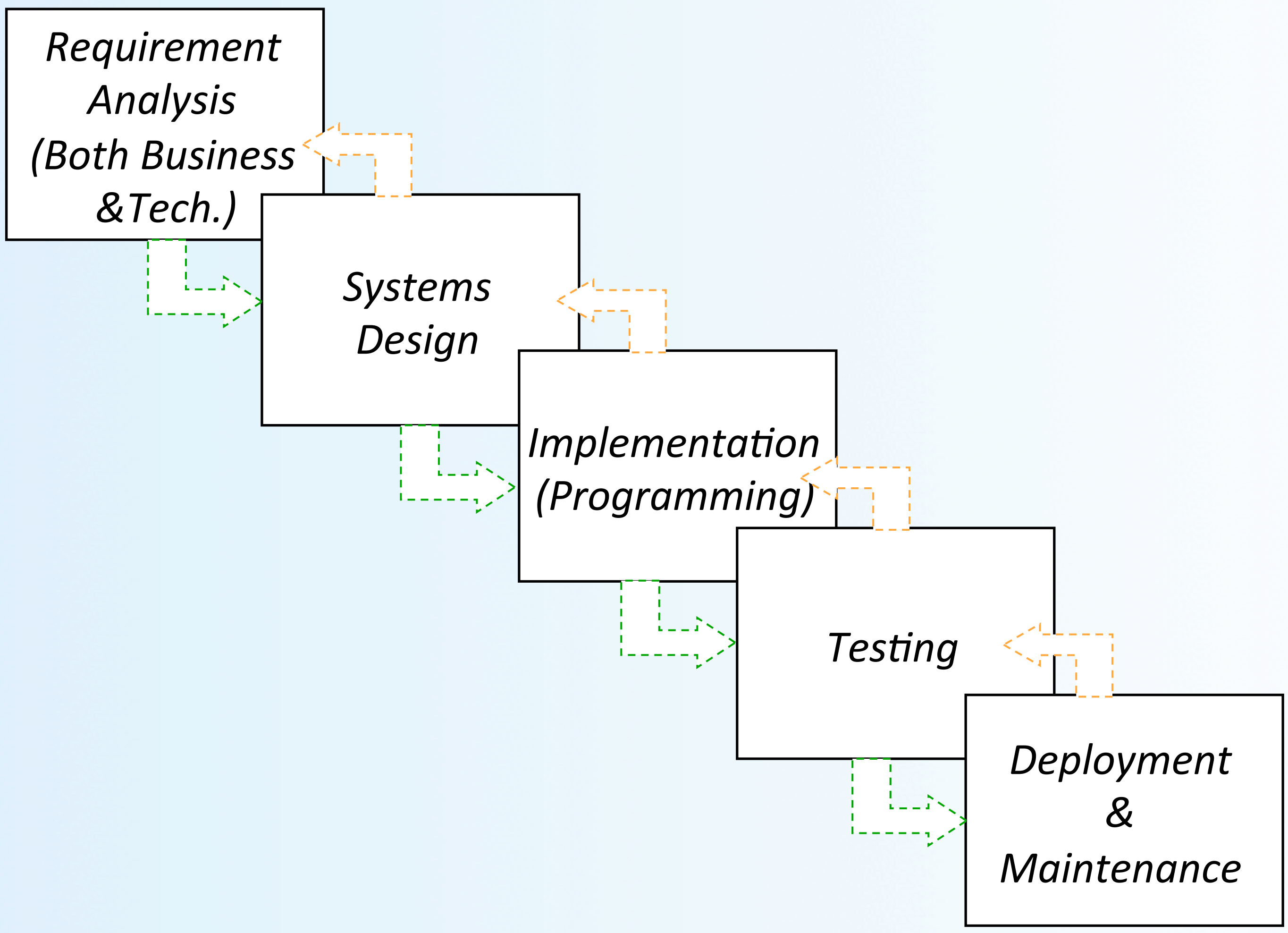
within **time and cost estimates**,

works **effectively** and **efficiently** in the current and

planned infrastructure,

and is **cheap** to maintain and **cost effective** to enhance.

Software Systems Development: Waterfall Model



Introduction to the C++ Language

A Little Historical Context…

- Derived from the C language
  - C was derived from the B language
  - B was derived from the BCPL language

- Why the ‘++’?
  - ++ is the increment operator in C++
  - Tongue-in-cheek naming…

Invention of C++

- C++ developed by Bjarne Stroustrup at AT&T Bell Labs in the 1980s.
  - Still maintains a webpage at http://www.stroustrup.com

- Overcame several shortcomings of C
- Incorporated object oriented programming
  - C++ is not a fully OOP language, though!!
- C remains a subset of C++

Object Oriented Programming (OOP)

- Used in most modern programs
- Program is viewed as made up of ***interacting objects***
- Each **object** contains algorithms to describe its behavior
- In the design phase, one designs objects and their algorithms

OOP Characteristics

- Encapsulation
  - Information hiding
  - Objects contain their own data and algorithms
- Inheritance
  - Writing reusable code
  - Objects can inherit characteristics from other objects
- Polymorphism
  - A single name can have multiple meanings depending on its context

A Sample C++ Program

A simple C++ program begins this way:

```
#include <iostream>
using namespace std;
int main()
{
```

And ends this way

```
 return 0;
}
```

```
1   #include <iostream>
2   using namespace std;
3   int main( )
4   {
5       int number_of_pods, peas_per_pod, total_peas;
6       cout << "Press return after entering a number.\n";
7       cout << "Enter the number of pods:\n";
8       cin >> number_of_pods;
9       cout << "Enter the number of peas in a pod:\n";
10      cin >> peas_per_pod;
11      total_peas = number_of_pods * peas_per_pod;
12      cout << "If you have ";
13      cout << number_of_pods;
14      cout << " pea pods\n";
15      cout << "and ";
16      cout << peas_per_pod;
17      cout << " peas in each pod, then\n";
18      cout << "you have ";
19      cout << total_peas;
20      cout << " peas in all the pods.\n";
21      return 0;
22  }
```

```
Press return after entering a number.
Enter the number of pods:
10
Enter the number of peas in a pod:
9
If you have 10 pea pods
and 9 peas in each pod, then
you have 90 peas in all the pods.
```

1-4:	Program start
5:	Variable declaration
6-20:	Statements
21-22:	Program end

cout << “some string or another” ; — *output stream statement*

cin >> some_variable; — *input stream statement*

stream is an entity where a program can either insert or extract characters

cout* and *cin* are objects defined in *iostream

Program Style

- The layout of a program is designed mainly to make it readable by humans
- Programs (i.e. compilers) accept almost any patterns of line breaks and indentations
- Conventions have established themselves, for example:
  1. Place opening brace '{' and closing brace '}' on a line by themselves
  2. Indent statements
  3. Use only one statement per line

Some C++ Rules and Conventions

- Variables are declared before they are used
  - Typically at the beginning of program
- Statements (not always lines) end with a semi-colon
- Include Directives (like `#include <iostream>`) placed in the beginning
  - Tell the compiler where to find information about items used in the program
- using namespace std;
  - Tells the compiler to use names in `iostream` in a "standard" way
- Main functions end with a return statement

TO DOs

- Readings
  - Chapter 2 of textbook
  - Only sections 2.1, 2.2, and 2.3
- Homework #2
  - Due on Thursday, 9/29
  - Submit in class
- Lab #1
  - Submit online via submit.cs **by FRIDAY at NOON!**

</LECTURE>