Introduction to C++

CS 16: Solving Problems with Computers I
Lecture #2

Ziad Matni
Dept. of Computer Science, UCSB

A Word About Registration for CS16

FOR THOSE OF YOU NOT YET REGISTERED:

- This class is currently **FULL**
- If you are on the waitlist, you will be added automatically as others drop the course
  - **THE WAITLIST WILL CLOSE ON FRIDAY AT 5 PM!**
  - **IF YOU'RE NOT REGISTERED BY THEN, YOU'RE NOT IN THE CLASS!**
- **If you are not on the waitlist, you will not get into this class**

Administrative

- **You must register on Piazza**
  - https://piazza.com/ucsb/spring2017/cs16
  - You will not get my **class announcements** otherwise!
    - I'm not using GauchoSpace

- Remember: Lab1 is due on Tuesday AT NOON
  - Use the **submit.cs** service as shown in lab on Wed.

- Class webpage: https://ucsb-cs16-s17.github.io

Switching About In The Labs…

… is frowned upon ☹

- Please stick to the lab time that you have per your registration
  - The labs are pretty full and at capacity

IF YOU WANT TO SWITCH LAB SECTIONS,
YOU MUST:

1. **Find a person in the other lab to switch with you**
2. Get the OK from ***BOTH*** T.A.s

Outline

- Computer Software
- Introduction to C++
- Programming and Problem Solving

Computer Software

- All the data
- All the programs
- All the applications
- The operating system(s)

- What is firmware?

The Operating System

- 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 like memory
  - Allows us to communicate with the computer via I/O
  - Responds to user requests to run other programs

Some Common OS

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

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

Image from technologydatagroup.com

Algorithm vs. Program

- "Computer Science is about studying how to use algorithms to solve problems"
  - True or False?

- **Algorithm**
  - A sequence of precise instructions that leads to a solution

- **Program**
  - An algorithm expressed in a language the computer can understand

Instructions for Machines

- Computers are digital machines
  - Their basic parts operate on digital “switching” using a ***binary*** code
  - Everything is in “**1**”s and “**0**”s (called ***bits***)

Collections of bits:
1 nibble = 4 bits
1 Byte = 8 bits
1 Word = 32 or 64 bits
(depends on the CPU)

- For example, for a particular CPU, the sequence of 32 bits
  “***00101100101101110000110000011000***”
  could be an instruction to add 2 numbers together

Instructions for Machines

- Instructions get executed in the CPU in ***machine language*** (all of it in bits)
  - 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

Computer Languages: Low-Level Languages

- It’s helpful to program in something OTHER than 1s and 0s
- ***Low-level languages*** provide some (low) abstraction to the CPU instructions
  - Allow you to use **MNEMONICS**, not bits, to define instructions
  - e.g. “ADD X Y Z” (add 2 numbers)
    “LB A 0x813B” (get a byte of data from computer memory)

- This is often called ***assembly language***
- A program that “translates” A.L. into M.L. is called an
  assembler

Computer Languages: High-Level Languages

- It would be even MORE helpful to program in "natural language"
- ***High-level languages*** provide high abstraction to the CPU instructions
  - You can now write programs that very much look like ***algorithms***
- You don't need to spell CPU cycles out at all
  - e.g. 1 statement, like "***x = c*(a + b)***" is enough to get the job done
- A program that "translates" H.L.L. into A.L./M.L. is called a

  compiler

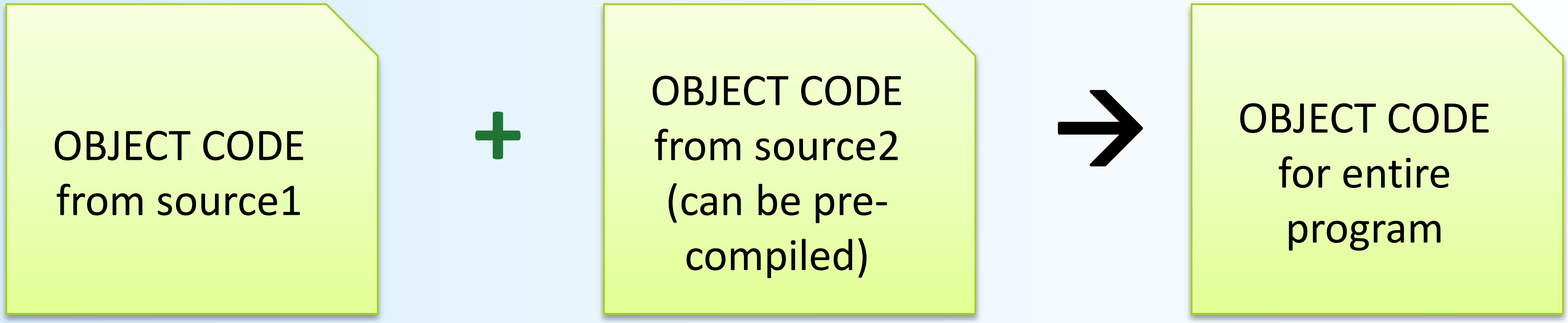
Compilers

- Language-specific
  - Compiler for Python will not work for C++, etc…
- Linux/UNIX OS have different built-in compilers
  - e.g. ***g++*** for C++, ***clang*** for C, etc…
- Source code
  - The original program in a high level language (text file)
- Object code
  - The translated version in machine language (binary file)

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



OBJECT CODE from source1

+

OBJECT CODE from source2 (can be pre-compiled)

→

OBJECT CODE for entire program

Introduction to the C++ Language

Invention of C++

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

- Overcame several shortcomings of its predecessor (**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
- When **designing a program**,
  one designs each object and their particular algorithms

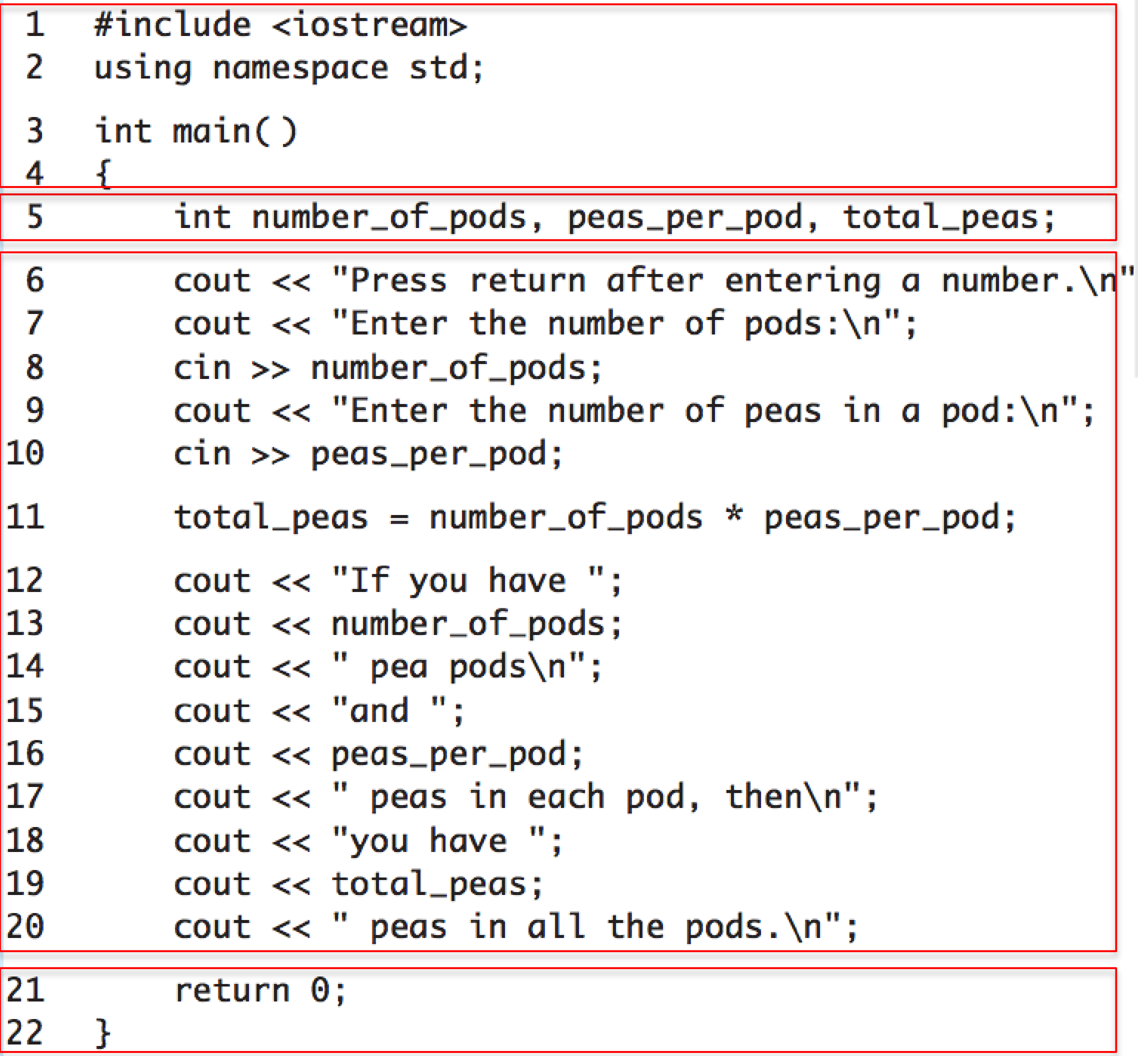
A Sample C++ Program

A simple C++ program begins this way:

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

And ends this way

```
 return 0;
}
```



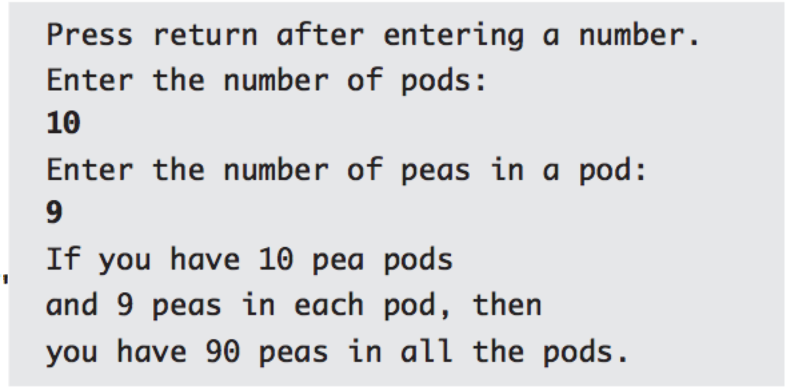
```
1   #include <iostream>
2   using namespace std;

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***

cout **and** ***cin*** **are objects defined in the library** ***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 been established for example:
  1. Place opening brace ‘{‘ and closing brace ‘}’ on a line by themselves
  2. Indent statements (i.e. use tabbed spaces)
  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***
- Use curly-brackets { … } to encapsulate groups of statements that belong together
  - Parentheses (…) have a different use in C++
  - As do square brackets […]

Some C++ Rules and Conventions

- ***Include directives*** (like `#include <iostream>`) always placed in beginning of the program before any code
  - Tells the compiler ***where to find*** information about objects used in the program
- `using namespace std;`
  - A statement that tells the compiler to use names of objects in `iostream` in a “standard” way
- `main` functions end with a “`return 0;`” statement

YOUR TO-DOs

- ❑ **Sign up on Piazza if you haven't yet**
- ❑ **Read** Chapter 2 (sections 2.1, 2.2, 2.3)
- ❑ Do **Homework 1** (due next TUESDAY 4/11)
- ❑ Finish up **Lab 1** and submit by TUESDAY 4/11 AT **NOON**
- ❑ I'll put up **Lab 2** online by Monday/Tuesday:
  give it a look when it's on there to prepare for Wed.

- ❑ Eat at least half of the vegetables on your plate.

</LECTURE>