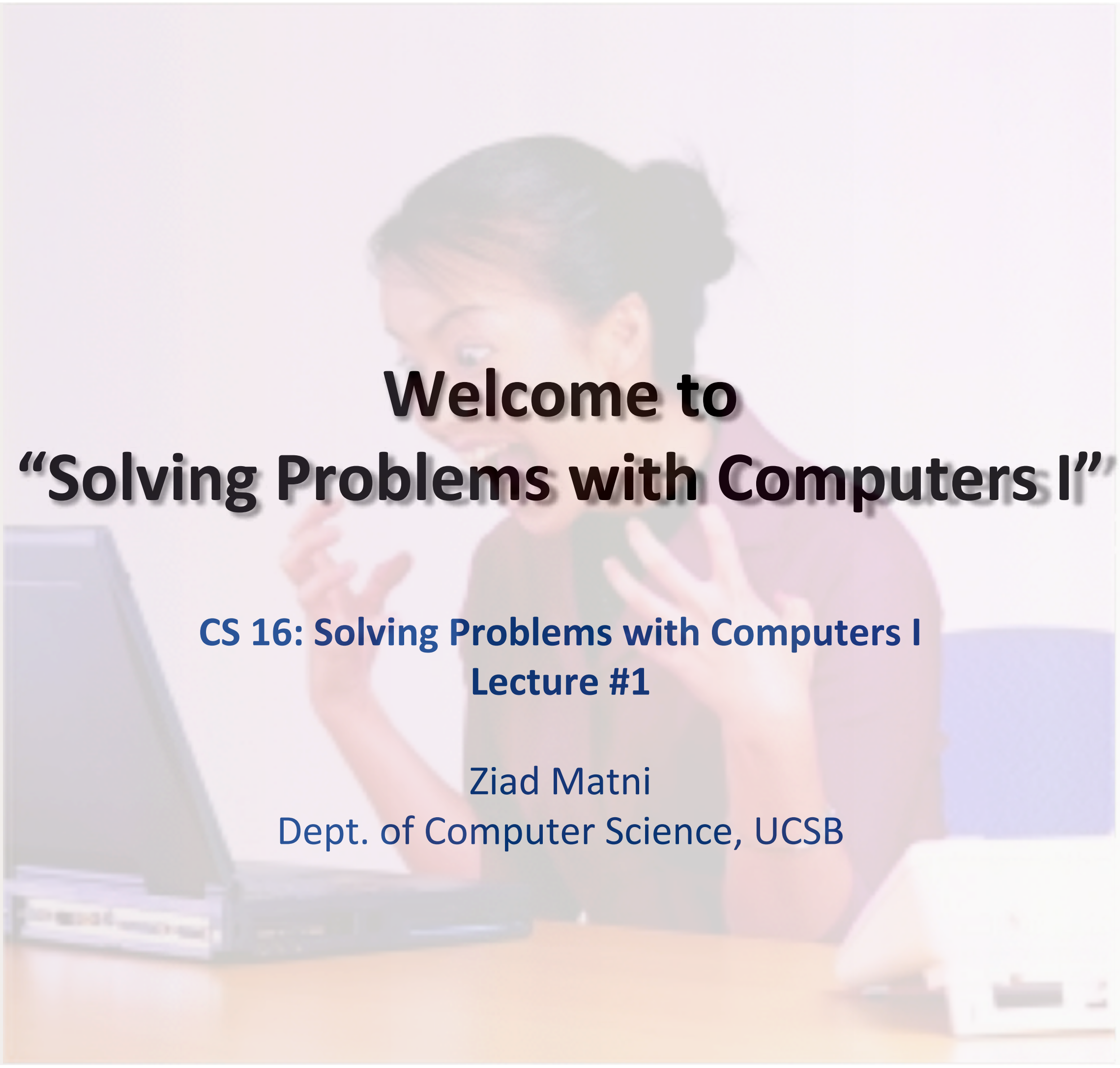


Welcome to “Solving Problems with Computers I”

CS 16: Solving Problems with Computers I
Lecture #1

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 and there's a LONG waitlist**
- If you are on the waitlist, you will be added automatically as others drop the course
- If you are not on the waitlist, you will not get into this class

Image from agorolabs on slideshare.com

Your Instructor

Your instructor: **Ziad Matni** *(zee-ahd mat-knee)*

Email: ***zmatni@cs.ucsb.edu***

(please put **CS16** at the start of the subject header)

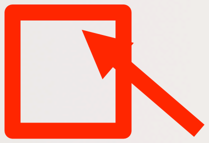
My office hours: Wednesdays **10:00 AM – 12:00 PM**, at **SMSS 4409**

(or by appointment)

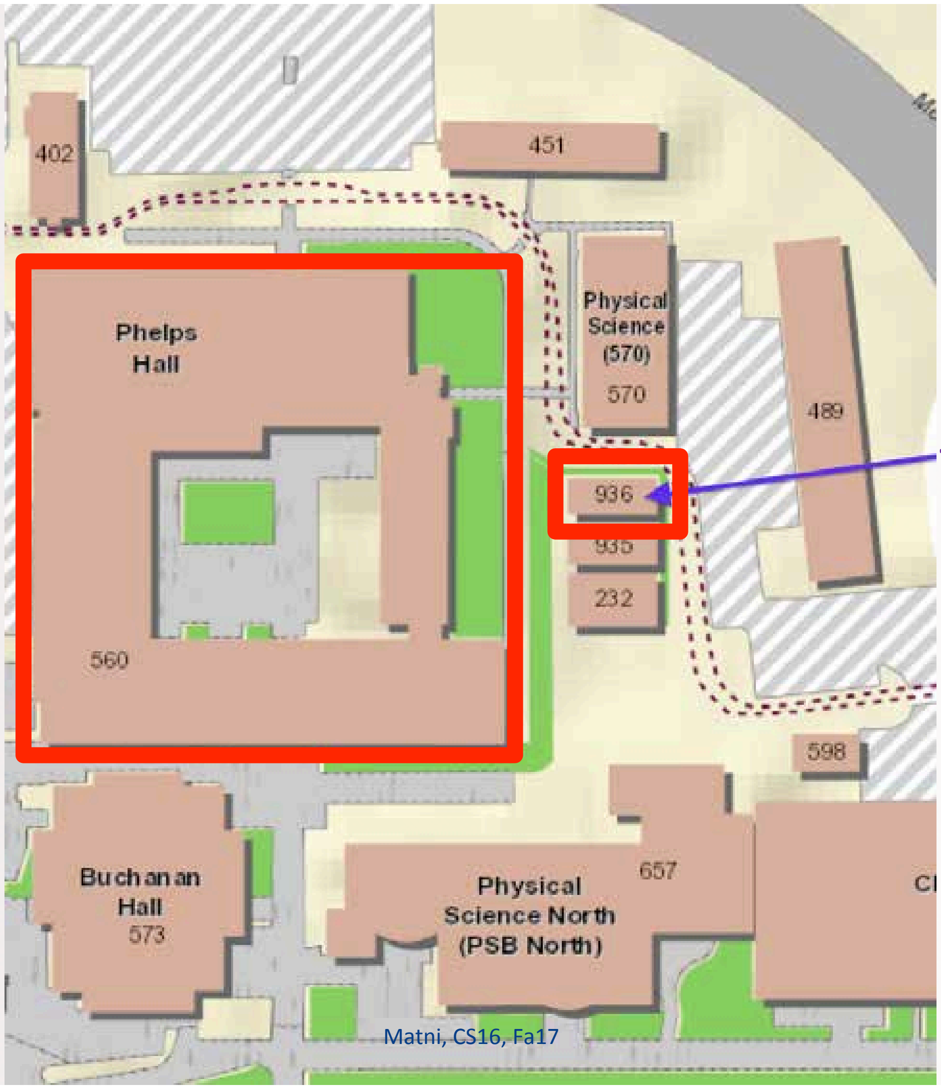
Your TAs and Graders

All labs will take place in **PHELPS 3525**
All TA office hours will take place in **TRAILER 936**

TA NAME	**LAB SECTION**	**OFFICE HOURS**
Xiyou Zhou	Mon. 8 am	Mon. 2 - 4 PM
Zhiyu “Zoey” Chen	Mon. 9 am	Thu. 10 AM – 12 PM
Yan Tang	Mon. 10 am	Fri. 2 - 4 PM
Muqsit Nawaz	Mon. 11 am	Tue. 4 - 6 PM
Shiyu Ji	Mon. 12 pm	Thu. 4 - 6 PM
Fatih Bakir	Mon. 1 pm	Tue. 4 - 6 PM
GRADER NAME		
Shane Masuda		Thu. 11 AM – 1PM
Siyi “Sydney” Ma		



YOU ARE HERE
(somewhere)



TRAILER 936

TRAILER 936

You!

With a show of hands, tell me… how many of you…

A. Are Freshmen? Sophomores? Juniors? Seniors?
B. Are CS majors? Other?
C. Have programmed before? What language?
D. Have programmed before “just for fun”?
E. Have programmed before “for work or school”?
F. Have used a Linux or UNIX system before?

This Class

- An **intermediate** (not a beginner's) class in computer science
  - You WILL need to have taken a beginner's class somewhere
- Covers the **basic building blocks for solving problems** using computers, in general, and using **C++ programming** specifically
- Enables you to go on to take other exciting classes in programming!!!!! OMG!!!

Why Are We Using C++ in this Course?

- ***C++ is one of the most widely used and in-demand computer programming languages***
  - For a list of commercial applications written in C++, see http://www.stroustrup.com/applications.html
- If you can learn C++, you can more easily learn (or even teach yourself) other popular P.L.s
  - Like Python, Java, PHP, Ruby, etc…
- It looks great on your resume!
  - Actually, it's a must on any "decent" CS major's resume…

How Is This Class Taught?

- Every class has a lecture based on the readings:

  YOU MUST DO THE READINGS BEFORE CLASS!!!

- You will be in a lab on Mondays:

  YOU MUST READ YOUR LAB ASSIGNMENT BEFORE YOU GO TO LAB!!!

- You have to do a lot of homework and lab assignments

BECAUSE PRACTICE MAKES PERFECT!!!
(and also, it's actually fun)

There's ~~actual~~ **A LOT OF** "work" to do…

• 8-9 Homeworks	once a week, all solo
• 8-9 Lab Assignments	once a week, some solo/some group
• 2 Midterm Exams	
• 1 Final Exam	

… and a partridge in a pear tree…

Resources?

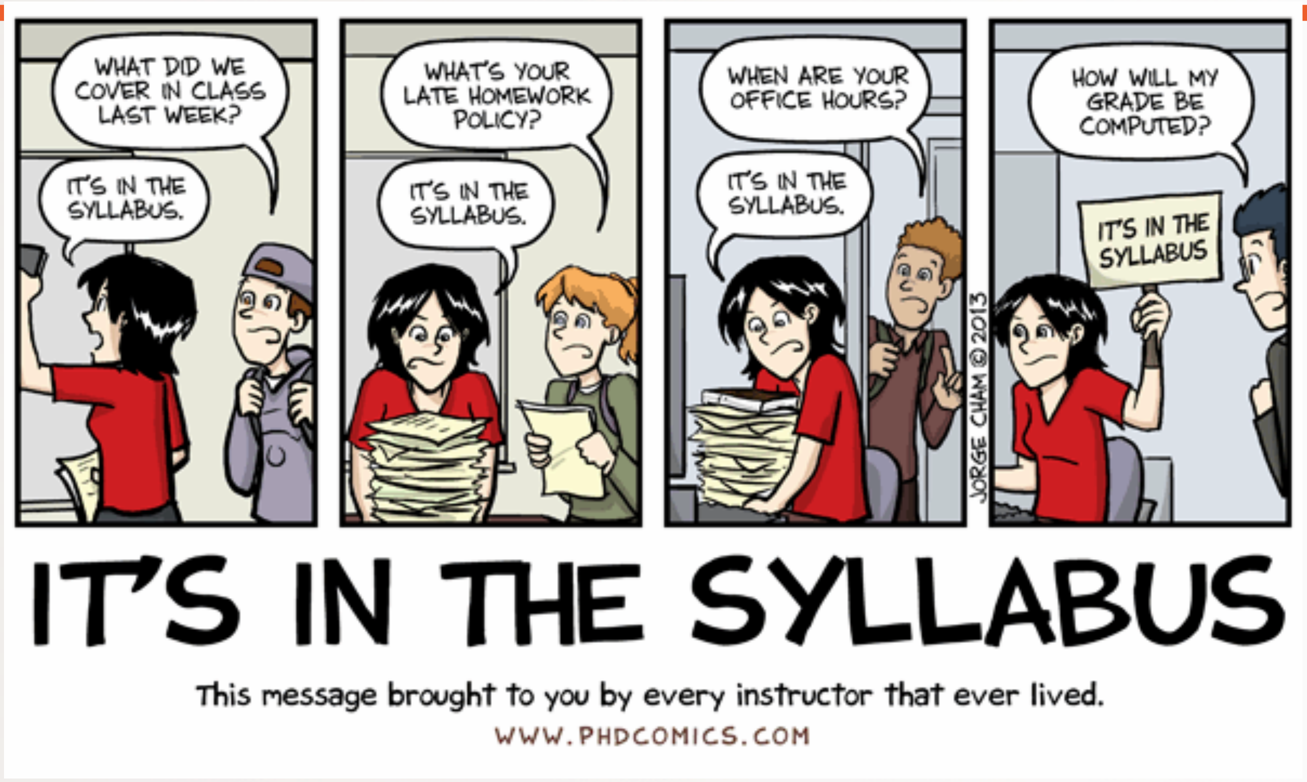
Class webpage:

https://ucsb-cs16-f17.github.io

Piazza discussions/Q&A:

https://piazza.com/ucsb/fall2017/cs16

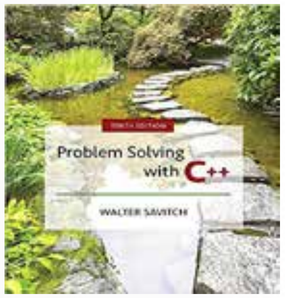
Just in Case…


WHAT DID WE COVER IN CLASS LAST WEEK?
IT'S IN THE SYLLABUS.
WHAT'S YOUR LATE HOMEWORK POLICY?
IT'S IN THE SYLLABUS.
WHEN ARE YOUR OFFICE HOURS?
IT'S IN THE SYLLABUS.
HOW WILL MY GRADE BE COMPUTED?
IT'S IN THE SYLLABUS
JORGE CHAM © 2013
IT'S IN THE SYLLABUS
This message brought to you by every instructor that ever lived.
WWW.PHDCOMICS.COM

So… let's take a look at that syllabus…

Full electronic version found at:
http://cs.ucsb.edu/~zmatni/syllabi/CS16F17_syllabus.pdf
PLEASE READ IT!!!!

Required textbook: **Problem Solving with C++ (10th Edition)** by Walter Savitch



Lectures: Uses the readings, but also adds it own components.
All assignments are based on what is said in lecture.
Slides are placed on website afterwards.

Homework: Once a week. Placed on website on Thursday. Due next Thursday IN CLASS.
YOU MUST PRINT OUT AND WRITE (CLEARLY) ON THE HOMEWORK SHEET! USE STAPLES!
Bring it to class (**no emails**).
Late policy: After due date, you get 20 pts off for "late" (< 24 hrs) or ZERO if after 24 hrs.

So… let's take a look at that syllabus…

Full electronic version found at:
http://cs.ucsb.edu/~zmatni/syllabi/CS16F17_syllabus.pdf
PLEASE READ IT!!!!

Labs: Once a week. Description sheet is placed on website on weekend. Lab is on Monday.
YOU MUST USE **submit.cs** TO TURN IN ALL LABS. PLEASE FOLLOW INSTRUCTIONS ON SHEET!
Must be turned in before Friday AT NOON.
Late policy: After due date, you get 20 pts off for "late" (< 24 hrs) or ZERO if after 24 hrs.

Most labs are solo-work, but some must be done in **pairs** (i.e. groups of 2 ppl).

Exams: 2 Midterms + 1 Final (cumulative).
No re-dos, no early takes, no late takes (except for extreme cases), no make-ups.

Laptop Policy: Can only use it to take notes, do class-related work.
Cell Phone Policy: Put that thing back where it came from or so help me… 😡

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, <u>YOU MUST</u>:

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

What YOU have to do *before MONDAY*

YOU HAVE A LAB on MONDAY!!!

- Log into **Piazza** and have a look around
  - Sign up for this class' page. Go to: https://piazza.com/ucsb/fall2017/cs16
- Go to the **class main website** and have a look around
  - Go to: https://ucsb-cs16-f17.github.io/
- Read the lab assignment (**lab01**) *before* you go into your lab: BE PREPARED

What YOU have to do *before Next Tuesday*

YOU HAVE ANOTHER LECTURE ON TUESDAY!!!

- **Do the required reading!!! (Chapter 1 and start Chapter 2)**
- **On the class main website:**
  1. Click on your first homework assignment (**h01**)
     - Best to click on the PDF link
  2. Print it DOUBLE SIDED
  3. Did you print it DOUBLE SIDED??????
     - NO??!?!?!?! GO BACK TO STEP 2!!!!
  4. Do the homework in pen or pencil and DO IT NEATLY!!!
  5. Bring the hardcopy of the homework (nice and stapled, please) **to class with you on Thursday and hand it in**

A Refresher Lecture on Computers

What is this “Computer” you speak of?

… and how can it help me “solve problems”???

Let’s define a “computer”

- Computer (n.): a computing device
- A device **that can be instructed** to carry out an **arbitrary** set of **arithmetic or logical operations** automatically

Computers = Computing Devices

Compute

(v) To make sense of ; to **calculate** or reckon

- What was the first computing tool ever?

COMPUTING TOOLS!

Invented around when humans fell out of the trees

Abacus → *Invented in China about 5000 years ago*
Mechanical computer → *Invented in France about 400 years ago*
Programmable computer → *Invented in UK about 150 years ago*
Electronic programmable computer → *Invented in UK/US about 70 years ago*

Abstraction

(n) A **mental model** that *removes complex details* *a very important concept in programming*

Do you need to know this?

🡐 🡐 🡐

To know how to do this?

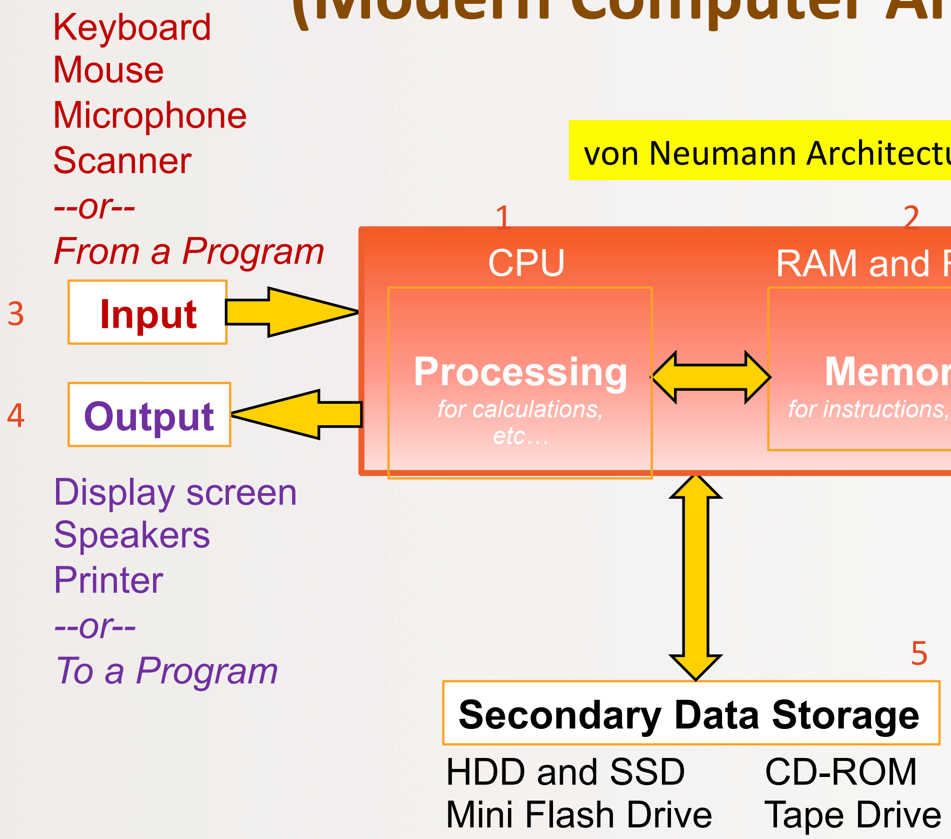
🡓 🡓 🡓

Images from jblearning.com

Computer Systems

- **Hardware**
  - The physical
    - CPU and Memory ICs
    - Printed circuit boards
    - Plastic housing, cables, etc…
- **Software**
  - The instructions and the data
    - Programs and applications
    - Operating systems

A Map of Computer Components (Modern Computer Architecture)



CPU = Central Processing Unit
RAM = Random-Access Memory
ROM = Read-Only Memory
HDD = Hard Disk Drive
SSD = Solid State Drive
OS = Operating System

5 Main Components to Computers

1. Processor
2. Main memory
   – Usually inside the computer, volatile
3. Inputs
4. Outputs
5. Secondary memory
   – More permanent memory for mass storage of data

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)
  - **Can one store *any* type of information building- block (like a number, or a letter) in 1 byte?**

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* (856,783,663,333)

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

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

Algorithm vs. Program

- "Computer Science is about studying how to use ____________ to solve problems"

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

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

High-Level Computer Languages

- A computer language that closely mimics "natural language"
  - As opposed to just being 0s and 1s (that's "machine language")
- ***High-level languages*** provide high abstraction to the CPU Instructions
  - Your programs very much look like ***algorithms***
- A program that "translates" a High Level Language into Low Level Language (like machine language) is called a ***compiler***
  - Why are compilers necessary???
  - ***Because CPUs ONLY understand their instructions in Machine Language***

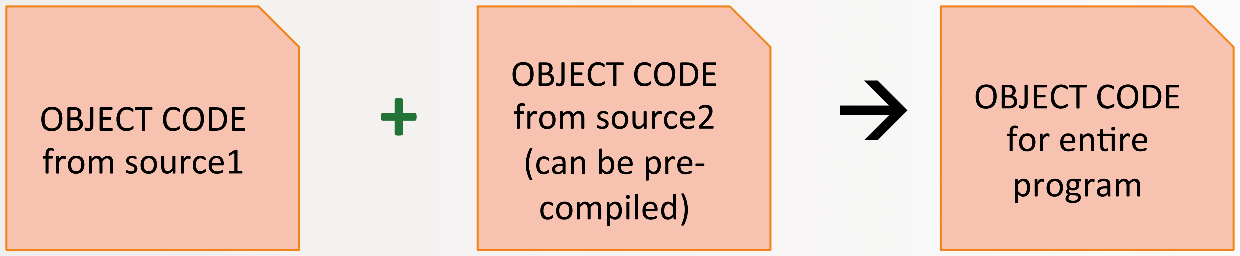
Compilers

- 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
  - We would just want to combine it 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 considered 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  }
```

Note the use of tabbed spaces

```
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
// means the following line is a comment

Program Style

- The **layout** of a program is designed mainly to make it **readable** by humans

- Compilers accept almost any patterns of line breaks and indentations!
  - So layout *conventions* are there not for the machine, but for the human

- 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 […]
  - They are not interchangeable!

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
    - More on this in a later class

- `main` functions end with a "`return 0;`" statement

YOUR TO-DOs

- ❑ Sign up on Piazza
- ❑ Go to the class website
- ❑ Read Lab1 THIS WEEKEND
- ❑ Do Lab1 MONDAY
- ❑ Do HW1 and hand it in on Thursday in class

- ❑ Solve world hunger
- ❑ Reverse global warming

</LECTURE>