Instructor and TAs

- Instructor: Tevfik Bultan <bultan@cs.ucsb.edu>
- Teaching Assistants:
  - Yuanhun Yao <yao@cs.ucsb.edu>
  - Ankita Singh <ankitasingh.2309@gmail.com>
  - Yi Yang <elaineyang127@gmail.com>
  - Jordan Hughes <jordanhughes@cs.ucsb.edu>
  - Plane Janthong <abhabongse@cs.ucsb.edu>
  - Suraj Rajesh <suraj_rajesh@cs.ucsb.edu>
  - Roman Kazarin <rkazarin@umail.ucsb.edu>
Class and Lab Times

• Class Times:
  – Tuesday/Thursday 12:30-1:45 Location: GIRV 1004

• Lab Times:
  – Wednesday 9:00-9:50 Location: PHELP 3525
  – Wednesday 10:00-10:50 Location: PHELP 3525
  – Wednesday 11:00-11:50 Location: PHELP 3525
  – Wednesday 12:00-12:50 Location: PHELP 3525
  – Wednesday 1:00-1:50 Location: PHELP 3525
  – Wednesday 2:00-2:50 Location: PHELP 3525

Prerequisites

Important assumption:
• Students who plan to take this course should be able to program in some high-level language, so already familiar with fundamental concepts:
  – Data types, data storage (memory), variables, constants, assignment, arithmetic (including integer arithmetic)
  – Logical operations, selection and iteration structures
  – Modular programming (functions/methods)
  – Array basics

Prerequisites:
• At least one of the following:
  – CMPSC 8
  – Math 3A
  – AP Computer Science (in High School)
  – "Intro to CS" course at another college
Instructor and TA Office Hours

- Instructor’s office hours are at HFH 2123
- Thursday 10:00-12:00

- The TA office hours are Trailer 936 Room 104

TA Office hours:
- Monday 10:00-12:00, Jordan Hughes
- Monday 1:00-3:00, Plane Janthong
- Monday 3:00-5:00, Roman Kazarin
- Tuesday 2:00-4:00, Yuanshun Yao
- Thursday 4:00-6:00, Suraj Rajesh
- Friday 10:00-12:00, Yi Yang
- Friday 1:00-3:00, Ankita Singh
Course Work

• 10 Programming Assignments:
  – Due Mondays Midnight
  – You can turn in programming projects up to 24 hours late for a 10% penalty. They will not be accepted if they are more than 24 hours late.
  – 10% of programming assignment grade will be lab attendance
  – You have to submit programming assignments via http://submit.cs.ucsb.edu
• 4 Homework Assignments:
  – No late HW is accepted
  – Deliver to the HW Box in HFH 2108
• 2 Midterm Exams:
  – April 26, Tuesday, in class
  – May 19, Thursday, in class
• Final Exam:
  – June 6, Monday at 12:00

Grade Distribution

• Programming Assignments: 40% (4% lab attendance, 36% programming assignment grade)
• Homework Assignments: 15%
• Midterm Exam 1: 15%
• Midterm Exam 2: 15%
• Final Exam: 15%
Announcements, Class discussion list

• All announcements and assignments will be on the class webpage, it is your responsibility to monitor it:
  http://www.cs.ucsb.edu/~cs16/

• Piazza Page:
  https://piazza.com/ucsb/winter2016/cs16
  – Join Piazza for class related discussion
  – Please post your questions to Piazza instead of directly e-mailing to the Instructor or the TAs
  – We will continuously monitor the Piazza and answer the questions for everyone to see

Check the class web site, the Piazza page, and your email once a day

Course Topics

• Catalog Specification

  Fundamental building blocks for solving problems using computers. Topics include basic computer organization and programming constructs: memory CPU, binary arithmetic, variables, expressions, statements, conditionals, iteration, functions, parameters, recursion, primitive and composite data types, and basic operating system and debugging tools.
Course Topics


Our goal is to cover chapters 1-10 and 13
- Introduction (Chapter 1)
- C++ Basics (Chapter 2), Control Flow (Chapter 3)
- Functions (Chapter 4, Chapter 5)
- I/O Streams (Chapter 6)
- Arrays, Multi-Dimensional Arrays (Chapter 7)
- Strings and Vectors (Chapter 8)
- Pointers and Dynamic Arrays (Chapter 9)
- Structures (Chapter 10.1)
- Pointers and Linked Lists (Chapter 13)

What is a computer? Some History

Babbage’s difference engine, 1853
What is a computer? Some History

Colossus, 1943
ENIAC (Electronic Numerical Integrator and Computer), 1945

What is a computer? Some History

IBM mainframes of 1960s and 1970s
What is a computer? Some History

IBM PC and Apple Macintosh from 1980s

What is a computer? Nowadays

smartphones

tables

datacenters

connected via internet

laptops

desktops
What is a computer? Nowadays

Cars also have computers in them
Planes also have computers in them

Possible future: Almost everything will have a computer in it connected to the network

What is a computer?

- A computer is a **programmable device**

- A **program** is a set of instructions that tell computer how to behave
  - So, a computer is something whose behavior is modifiable by providing a program

- After giving a computer a program, we can ask the computer to follow the instructions of the program
  - This is called **executing** or **running** the program

- When we run the program on a computer, the computer behaves exactly as instructed by the program
Why call it computer?

- A computation is producing an output from a set of given inputs based on a given program
- So a computer is the thing that does computation
- But, if you had a device that was not programmable and only performed a single computation, then we would not call it a computer
- So, “programmable computing device” would have been a more accurate name than “computer”

Organization of a Computer

- We said that a program is a set of instructions, at its core the computer has a processor that executes instructions
  - Processor is the thing we are instructing what to do by writing a program
- During computation a computer may have some intermediate results that it needs to store
  - It needs a memory to remember these intermediate results
Organization of a Computer

- A computer needs to interact with the outside world
- A computer needs way to **receive inputs**
  - keyboard, mouse, touch-screen, microphone, camera, etc.
- A computer needs ways to **report outputs**
  - screen, printer, speakers, etc.

Organization of a Computer

- Five main components
  - Input devices
    - Allows communication to the computer
  - Output devices
    - Allows communication to the user
  - Processor (CPU)
  - Main memory
    - Memory locations containing the running program
  - Secondary memory
    - Permanent record of data often on a disk
### Computer Memory

- Main memory stores instructions and data while a program is running

- Main Memory
  - Long list of memory locations
    - Each contains zeros and ones
    - Can change during program execution
  - Binary Digit or Bit
    - A digit that can only be zero or one
  - Byte
    - Each memory location has eight bits
  - Address
    - Number that identifies a memory location

### Larger Data Items

- Some data is too large for a single byte
  - Most integers and real numbers are too large

  - Address refers to the first byte

  - Next few consecutive bytes can store the additional bits for larger data
Memory

- Memory is divided into numbered locations called **bytes**
- The number that identifies a memory location is called its **address**
- A group of consecutive bytes can be used as the location for a data item, such as a number or letter. The address of the first byte is called the address of the data item.

### Data or Code?

- ‘A’ may look like 01000001
- 65 may look like 01000001
- An instruction may look like 01000001

- How does the computer know the meaning of 01000001?  
  - Interpretation depends on the current instruction

- Programmers rarely need to be concerned with this problem.  
  - Reason as if memory locations contain letters and numbers rather than zeroes and ones
Secondary Memory

• Secondary memory
  – Stores instructions and data between sessions
  – A file stores data or instructions in secondary memory

• A computer might have any of these types of secondary memory media
  – Hard disk
    • Fast
    • Fixed in the computer and not normally removed
  – Flash drive
    • Slower than hard disks
    • Easily shared with other computers
  – Compact disk
    • Slower than hard disks
    • Easily shared with other computers
    • Can be read only or re-writable

Memory Access

• Random Access
  – Usually called RAM
    • Computer can directly access any memory location

• Sequential Access
  – Data is generally found by searching through other items first
    • More common in secondary memory
The Processor

- Typically called the CPU
  - Central Processing Unit
  - Follows program instructions
  - Typical capabilities of CPU include:
    - add
    - subtract
    - multiply
    - divide
    - move data from location to location

- Computer processor can be divided to specialized units
  - Arithmetic Logic Unit (ALU) that only specializes in arithmetic calculations

- Computer can have co-processors that do specializes computations
  - Graphics coprocessor

Hardware vs. Software

- The physical parts of the computer is called the hardware

- The programs that run on the computer are called the software

- Programs, code, software have similar meanings

- The person who writes programs or software is a programmer or a software developer
## Computer Software

- A computer program is...
  - A set of instructions for a computer to follow

- Computer software is …
  - The collection of programs used by a computer

## Computer Software

- The operating system
  - Allows us to communicate with the computer
  - Is a program
  - Allocates the computer’s resources
  - Responds to user requests to run other programs

- Common operating systems include…
  - UNIX, Linux, Windows, MacOS
Computer Input

- Computer input consists of
  - A program
  - Some data

How do we program?

- Computer scientists developed programming languages to help software developers in writing programs
  - Low level languages: assembly languages, machine languages
  - High level programming languages
### High-level Languages

- High level programming languages are developed to make it easier for humans to write programs.
- Common high level programming languages include: FORTRAN, C, Lisp, C++, Java, PHP, Python, JavaScript, etc.
- These high level languages
  - Resemble human languages
  - Are designed to be easy to read and write
  - Use more complicated instructions than the CPU can follow
  - Must be translated to zeros and ones for the CPU to execute a program
- High level programming languages contain **libraries** that are programs that handle common tasks (such as printing to screen) which make writing programs easier.

### Low-level Languages

- Machine languages
  - These are languages of the microprocessor, it tells microprocessor what instruction to execute, written as 0s and 1s
- Assembly languages
  - These languages are very close the machine language, but instead of 0s and 1s it is written typically using ASCII characters to be more readable for humans. An assembler translates it to the machine language.
Low-level Languages

- An assembly language command such as

  \[\text{ADD } X \ Y \ Z\]

  might mean add the values found at \(x\) and \(y\) in memory, and store the result in location \(z\).

- Assembly language must be translated to machine language (zeros and ones)

  \[0110 \ 1001 \ 1010 \ 1011\]

- The CPU can follow machine language

Compilers

- Translate high-level language to machine language

  - 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
  - For example: Input and output routines

- A Linker combines
  - The object code for the programs we write
  - The object code for the pre-compiled routines
  - The machine language program the CPU can run

Compiling a program

- In this class we are going to learn a high level programming language called C++
- We will translate C++ programs to the object code using a C++ compiler
- The compiled program then will be linked with the libraries, and then it can be executed on a computer
Algorithms

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

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

An Example Algorithm

An Algorithm

Algorithm that determines how many times a name occurs in a list of names:

1. Get the list of names.
2. Get the name being checked.
3. Set a counter to zero.
4. Do the following for each name on the list:
   - Compare the name on the list to the name being checked,
     and if the names are the same, then add one to the counter.
5. Announce that the answer is the number indicated by the counter.
Program Design

• Programming is a creative process
  – No complete set of rules for creating a program

• Program Design Process
  – Problem Solving Phase
    • Result is an algorithm that solves the problem
  – Implementation Phase
    • Result is the algorithm translated into a programming language

Problem Solving Phase

• Be certain the task is completely specified (requirements analysis)
  – What is the input?
  – What information is in the output?
  – How is the output organized?

• Develop the algorithm before implementation (software design)
  – Experience shows this saves time in getting your program to run.
  – Test the algorithm for correctness
Implementation Phase

- Translate the algorithm into a programming language
  - Easier as you gain experience with the language

- Compile the source code
  - Locates errors in using the programming language

- Testing: Run the program on sample data
  - Verify correctness of results

- Testing results may require modification of the algorithm and program

Software Development Process
## Software Life Cycle

- Developing software takes a lot of effort.
- One can identify several phases in software development
  - **Requirements analysis**: Figuring out precisely the task that the software we plan to develop is supposed to do
  - **Software design**: Coming up with a high level solution to how to do that task in a program
  - **Software implementation**: Translating the high level design into a program written in a programming language
  - **Testing and debugging**: Checking the program to identify and eliminated errors
  - **Maintenance**: Improvements and modifications to the program after it is released
- All these phases together are called the **software life-cycle** or the **software process**

## Object Oriented Programming

- Abbreviated OOP
- Used for many modern programs
- Program is viewed as interacting objects
  - Each object contains algorithms to describe its behavior
  - Program design phase involves designing objects and their algorithms
<table>
<thead>
<tr>
<th>OOP Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Encapsulation</td>
</tr>
<tr>
<td>– Information hiding</td>
</tr>
<tr>
<td>– Objects contain their own data and algorithms</td>
</tr>
<tr>
<td>• Inheritance</td>
</tr>
<tr>
<td>– Writing reusable code</td>
</tr>
<tr>
<td>– Objects can inherit characteristics from other objects</td>
</tr>
<tr>
<td>• Polymorphism</td>
</tr>
<tr>
<td>– A single name can have multiple meanings depending on its context</td>
</tr>
</tbody>
</table>