SYDE 121
Section 1 - Introduction

- Today:
  - 1.1 Computing Basics
  - 1.2 Computing History
  - 1.3 Computer Systems
  - 1.4 Algorithms
- Readings:
  - Ch. 1 Savitch (course text)
  - SD121 Style Guide (on website) (Note: you will have to review this document as we progress through the course)

1.1 Computing Basics

- What is a computer??
  - device used to process data and/or information

data: raw inputs

information: transformation of data into something meaningful

1.1 Computing Basics (cont.)

- Why do engineers learn about computers?
  - essential tool for an engineer
  - industry demands
  - perform computational tasks
  - life-skill: add to 3 R's
  - develop logical problem solving skills

1.1 Computing Basics (cont.)

- What are computers used for??
  - computations: Matlab
  - graphics: CAD, image analysis
  - business and management: spreadsheets, scheduling
  - communication: email, cellular telephones
  - automation: robotics, HVAC
  - information technology (IT): internet, GIS
  - education

1.1 Computing Basics (cont.)

- Advantages of using a computer?
  - performs calculations faster and more reliably than a human
  - once algorithm works, can be applied over and over
  - no fatigue
  - do not have to pay the computer a salary!
  - cheaper to simulate design first and then test

1.1 Computing Basics (cont.)

- Disadvantages of using a computer?
  - special cases must be accounted for
  - cannot "think" ... yet!
  - frustrating/intimidating especially for first time users
  - difficulties with qualitative reasoning/decision making
  - expense (initial outlay, upkeep)
  - usually quite difficult if not impossible to simulate the real world perfectly

1.2 Computer History Bits

- earliest computer? abacus ~2600 B.C.
- automatic mechanical calculator (Pascal, 1642)
- 1621: first slide rule
- Charles Babbage (1791 - 1871): father of modern computing
- Augusta Ada (1815 - 1853): iterative structures
- Alan Turing (1937): hypothetical machine
- Claude E. Shannon: boolean logic and switching circuits
- John Atanasoff (1939) built the ABC (Atanasoff Berry Computer) - first computer to use vacuum tubes

1.2 Computer History Bits (cont.)

- Collusus: machine used in WWII to break enemy codes
- Howard Aiken (1939) began work on a program controlled computer, the Mark I
- during Mark II development, dead moth caused relay to fail; the first computer "bug"!!!
- ENIAC (Electronic Numerical Integrator and Calculator) (1946); first large-scale computer (18,000 vacuum tubes)
- 1947: transistor
- UNIVAC (1951): first commercial computer
- 1954: development of Fortran; first high level programming language

1.2 Computer History Bits (cont.)

- 1959: first integrated circuit
- 1962: Paul Baran of RAND develops the idea of distributed, packet-switching networks
- 1963: PDP-1, first microcomputer
- 1964: IBM SYSTEM/360
- 1969: ARPANET goes online
- 1973: Basic aspects of Internet created
- 1977: advent of first completely assembled personal computers (Apple Computer, Radio Shack, and Commodore)
  - e.g. Apple II sold for $1,200, 16k of RAM, no monitor

1.2 Computer History Bits (cont.)

- 1981: first IBM PC appears
- 1982: TCP/IP (Transmission Control Protocol and Internet Protocol) is established as the standard for ARPANET
- 1984: Apple Macintosh debuts; GUI; 8-MHz, 32-bit Motorola 68000 CPU; built-in 9" B/W screen
- 1984: MIDI standards established
- 1984: CD-ROM introduced
- 1985: Microsoft Windows 1.0 ships
- 1985: C++ issued (Bell Labs)

1.2 Computer History Bits (cont.)

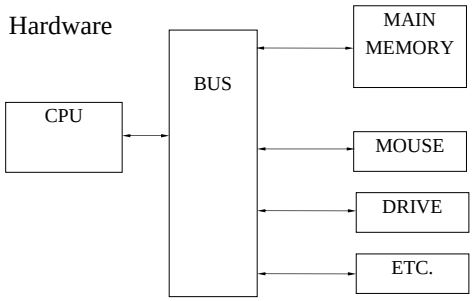
- 1988: worm released on Internet
- 1989: Number of network hosts breaks 100,000.
- 1989: Tim Berners-Lee proposes WWW to CERN
- 1989: IRC implemented
- 1991: First webserver released
- 1993: Mosaic (webbrowser) released
- 1995: Java released
- 1996: 1 of 3 U.S. homes has a PC

1.3 Computer Systems

- What are the basic components of a computer?
  - Hardware: physical components of the computer such as the monitor, keyboard, printer, computer chips, etc.
  - Software: programs that run on the computer (eg. word processing, games)

1.3 Computer Systems (cont.)

- Hardware

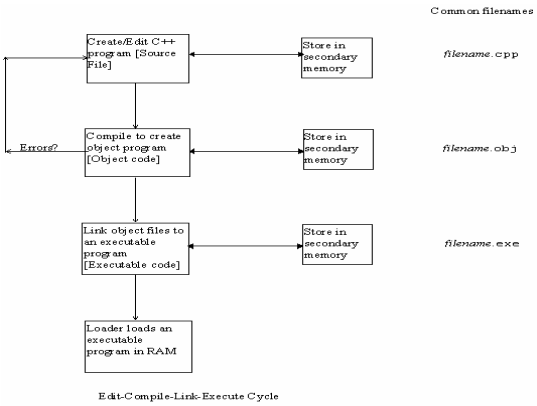


1.3 Computer Systems (cont.)

- Information representation
  - Memory: 1 bit: 0 or 1; 8 bits: 1 byte; usually use megabytes (Mb) (10e6 bytes) or gigabytes (Gb) (10e9 bytes)
  - Text: ASCII; each character represented by different bit pattern
  - Instructions: machine language

1.3 Computer Systems (cont.)

- Compilers
  - source code (typed by human) (*.cpp) is converted using a *compiler* into the *object* code
  - *linker* links multiple object codes into *executable* program (*.exe)



Edit-Compile-Link-Execute Cycle

1.3 Computer Systems (cont.)

- PCs: generally used at home or business
- Workstations: networking capability; originally the realm of UNIX O/S, now Windows NT as well
- Mainframe: larger (and more expensive!) computer designed for many users

1.3 Computer Systems (cont.)

- What is a computer program?
  - set of instructions that a computer follows to process raw data into meaningful information
- What is a programmer?
  - person who writes computer programs
- What is a software engineer?
  - person who designs, maintains, implements, manages, etc. computer programs

1.3 Computer Systems (cont.)

- Programming languages:
  - low level: primitive eg. assembly language
  - high level : more readable statements eg. Java, C, C++, C#, Pascal, BASIC, FORTRAN
- Assembly language statement may read as:

  ADD X Y Z
- Comparable high level language statement:

  Z = X + Y

1.4 Algorithms and Object Oriented Design

- What is an algorithm?
  - a set of deterministic steps that
  - converts input information to output information
- An algorithm is:
  - unambiguous, and
  - executable, and
  - terminates

Five-Step Problem Solving Methodology

I. Problem Analysis
II. Design
III. Coding and debugging
IV. Integration
V. Testing and Validation

I. Problem Analysis

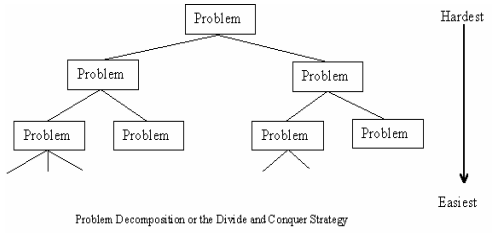
Problem Statement: Understand what problem needs to be solved and prepare a *concise* problem statement.

Input/Output: Analyze and write down what are the necessary inputs and the outputs.

II. Design

- Problem Decomposition:
  - Top-down: divide problem iteratively into more manageable subproblems until solvable by hand
  - Object/class design: organize data and functions into objects/classes
- Algorithm Design:
  - study simple hand example
  - algorithm design
  - prepare pseudocode

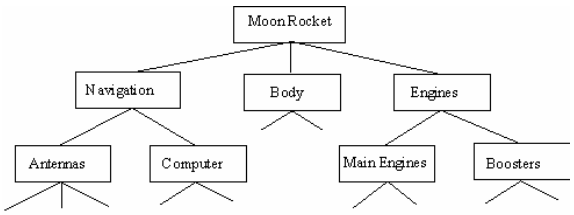
Top-Down Design Process



Problem Decomposition or the Divide and Conquer Strategy

Divide the problem into easier and easier subproblems

Top-Down Design Example



Decomposition of a Moon Rocket

Design Continued....

- Hand Example:
  - Solve by hand an example. If not possible go back and decompose the problem further!
- Algorithm Design:
  - The hand example should now help you make an algorithm (steps to follow to solve the problem) which can be applied to all problems of that kind!

III. Coding and Debugging

- Code
  - Code your design (usually a single module at a time) into the chosen language. In this course we will use C++.
- Debug
  - Debugging is a process of observing, identifying, and removing errors.

IV. Integration

- Integrate smaller modules to solve overall problem
- Complex with large scale programs with many programmers
- For simple problems this step can be combined with Step 3: Coding and debugging.

V. Testing and Validation

- Testing:
  - Test the program with various test cases and *verify* that the program is correct!
- Validation:
  - Go back to the problem analysis step and check that the problem you have solved is what is required of you!

Simple Example

- Problem: Given a student grade record, update the record by calculating the average of the grades.

```
81990021

90 78 55 67 75

-1
```

An Example of the Student Grade Record

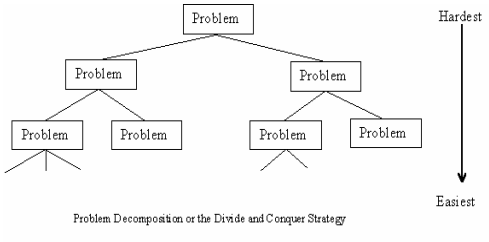
I. Problem Analysis

- Problem Statement: Calculate the average grade and write out the student_grade_record with the average.
- Input/Output Analysis:

Input: student_grade_record

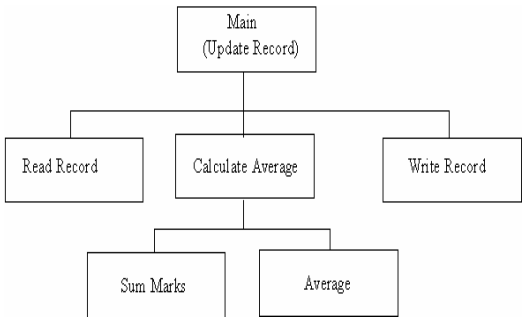
Output: student_grade_record with the calculated average.

II. Design: Top-Down Design Process



Problem Decomposition or the Divide and Conquer Strategy

Divide the problem into easier and easier subproblems



Top-Down Decomposition

Hand Example

Hand Example: The average grade can be found by the following two steps:

sum = 90 + 78 + 55 + 67 + 75 = 365

average = sum/5 = 365/5 = 73

Algorithm Design

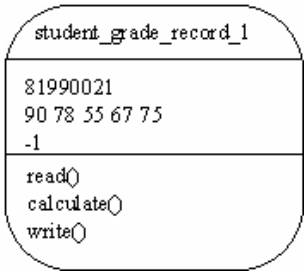
When the average has not yet been calculated

sum = grade1 + grade2 ... + gradeN

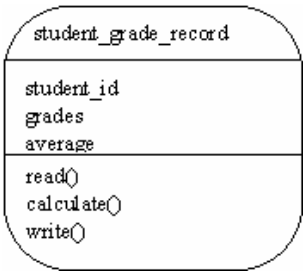
average = sum /N

Note: this pseudocode becomes more elaborate with larger programs

Object based Decomposition



Class Design

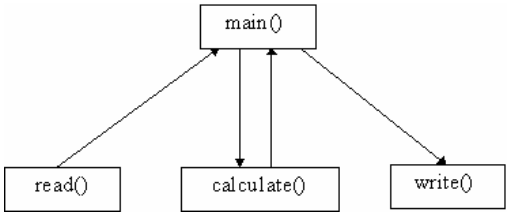


III. Coding

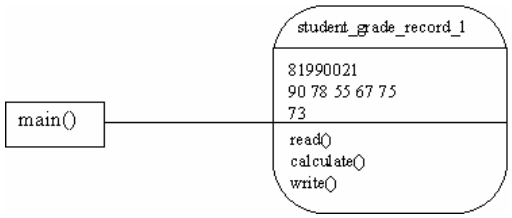
Solution 1:

```
//Main Program
.....
//Read student record
.....
//Calculate average
.....
.....
//Write student record
.....
```


Solution 2



Solution 3



Details on

4. Integration

and

5. Testing and Validation

are not appropriate for presentation at the moment, but will be revisited.

USE THIS ALWAYS!

Summary of problem solving methodology:

1. Problem Analysis:
   - · Problem Statement
   - · Input/output analysis
2. Design:
   - · Decomposition
   - · Class Design
   - · Hand Example
   - · Algorithm Design
3. Coding and Debugging:
4. Integration:
5. Testing and Validation: