Principles of Computer Science I

Prof. Nadeem Abdul Hamid

CSC 120A - Fall 2004

Lecture Unit 1

Computer Science and Programming

- Computer Science is more than programming
  - The discipline is called *informatics* in many countries
  - Elements of both science and engineering
    - Scientists build to learn, engineers learn to build

      – Fred Brooks
  - Elements of mathematics, physics, cognitive science, music, art, and many other fields
- Computer Science is a young discipline
  - Fiftieth anniversary in 1997, but closer to forty years of research and development
  - First graduate program at CMU (then Carnegie Tech) in 1965
- To some programming is an art, to others a science

What is Computer Science?

What is it that distinguishes it from the separate subjects with which it is related? What is the linking thread which gathers these disparate branches into a single discipline? My answer to these questions is simple --- *it is the art of programming a computer*. It is the art of designing efficient and elegant methods of getting a computer to solve problems, theoretical or practical, small or large, simple or complex.

C.A.R. (Tony) Hoare

CSC 120 - Course Mechanics

- Syllabus on Viking Web (*Handouts* section)
- Class Meetings
  - Lectures: Mon/Wed/Fri, 10–10:50AM, SCI 107
  - Labs: Thu, 2:45–4:45 PM, SCI 228
- Contact
  - Office phone: (706) 368-5632
  - Home phone: (706) 234-7211
  - Email: *nadeem@acm.org*
- Office Hours: SCI 354B
  - Mon 11-12:30, 2:30-4
  - Tue 9-11, 2:30-4
  - Wed 11-12:30
  - Thu 9-11
  - (or by appt)

CSC 120 – Keys to Success

- Start early; work steadily; don't fall behind.
- You can't cram, unlike most other courses.
- If you get stuck, take a break and then go back to think about it.
- Don't hesitate to contact the instructor if you have any problems.

CSC 120 – Materials & Resources

- Textbooks: (1) Nell Dale, et al., *Programming and Problem Solving with Java*. (2) Lab manual
- Online course website: Check regularly
  - Handouts, assignments, labs, forum
- Hardware/Software:
  - Computer lab (SCI 228)
  - Java SDK
  - BlueJ development environment
  - SciTE (Scintilla text editor)

CSC 120 – Assignments & Grading

- Participation (10%)
  - Attendance policy
- Assignments (40%)
  - Weekly/Bi-weekly
  - Written and programming
- Labs (15%)
  - Complete work and turn in lab book at the end of session
  - Submit completed programming problems via VikingWeb
- Exams (35%) *(tentative dates)*
  - First Exam (10%), Wednesday, September 15, 2004
  - Second Exam (10%), Friday, October 22, 2004
  - Final Exam (15%), Monday, December 6, 2004 (10:30–12:30)

CSC 120 – Policies

- Attendance
- Academic Integrity
- Late Work
- Disabilities

(See Syllabus for details)

Computer Science as a Discipline

Is it part of…

- Mathematics? (theory)
- Science? (experimentation)
- Engineering? (design)

Computer Science - Subareas

Architecture	hardware-software interface
Artificial Intelligence	thinking machines
Computational Geometry	theory of animation, 3-D models
Graphics	from Windows to Hollywood
Operating Systems	run the machine
Scientific Computing	weather, hearts
Software Engineering	peopleware
Theoretical CS	analyze algorithms, models

- Many other subdisciplines… *numerical and symbolic computation, bioinformatics, databases, and information retrieval, human-computer communication*

Themes and Concepts of CS

- Theory
  - properties of algorithms, how fast, how much memory
  - average case, worst case: sorting cards, words, exams
  - *provable* properties, in a mathematical sense
- Language
  - programming languages: Java, C, C++, C#, Perl, Fortran, Lisp, Scheme, Visual BASIC, ...
  - Assembly language, machine language,
  - Natural language such as English
- Architecture
  - Main memory, cache memory, disk, USB, SCSI, ...
  - pipeline, multi-processor

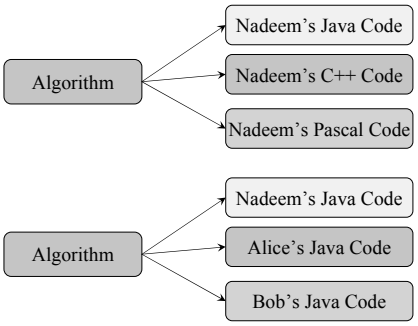
Algorithms: A Cornerstone of CS

- Step-by-step process that solves a problem
  - more precise than a recipe
  - eventually *stops* with an answer
  - general process rather than specific to a computer or to a programming language
- Searching: for phone number of G. Samsa, or for the person whose number is 489-6569
- Sorting: zip codes, hand of cards, exams
  - Why do we sort? What are good algorithms for sorting?
    - It depends
      - Number of items sorted, kind of items, number of processors, ??
  - Do we need a detailed sorting algorithm to play cards?

Sorting Experiment

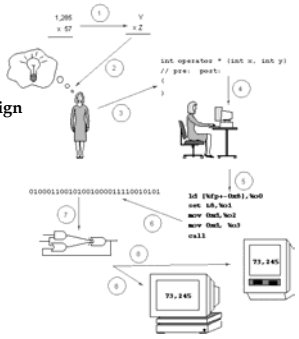
- Groups of four people are given a bag containing strips of paper
  - on each piece of paper is an 8-15 letter English word
  - create a sorted list of all the words in the bag
  - there are 100 words in a bag
- What issues arise in developing an algorithm for this sort?
  - ...
- Can you write a description of an algorithm for others to follow?
  - Do you need a 1-800 support line for your algorithm?
  - Are you confident your algorithm works?

Algorithms to Programs



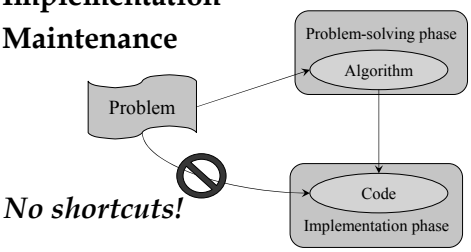
Writing a Program

- Specify the problem
  - remove ambiguities
  - identify constraints
- Develop algorithms, design classes, design software architecture
- Implement program
  - revisit design
  - test, code, debug
  - revisit design
- Documentation, testing, maintenance of program
- From ideas to electrons



Software Life Cycle

1. Problem-Solving and Design
2. Implementation
3. Maintenance



No shortcuts!

Problem-Solving Phase

- ANALYZE the problem and SPECIFY what the solution must do
- Design a GENERAL SOLUTION (ALGORITHM) to solve the problem
- VERIFY that your solution really solves the problem

Sample Problem

- A programmer needs an algorithm to determine an employee's weekly wages
- How would the calculations be done by hand?

One Employee's Wages

- During one week an employee works 52 hours at the hourly pay rate $24.75
- How much is the employee's wages?
- Assume a 40.0 hour normal work week
- Assume an overtime pay rate factor of 1.5

40 x $ 24.75	=	$ 990.00
12 x 1.5 x $ 24.75	=	$ 445.50
		$ 1435.50

Generalized Solution

If hours is over 40.0, then

wages

= (40.0 * payRate) + (hours - 40.0) * 1.5 *payRate

RECALL EXAMPLE

(40 x $ 24.75) + (12 x 1.5 x $ 24.75) = $1435.50

otherwise,

wages = hours * payRate

Employee's Weekly Wages

Objects: Employee, pay rate, hours worked, wages

Algorithm:

1. Get the employee's hourly pay rate
2. Get the hours worked this week
3. Calculate this week's regular wages
4. Calculate this week's overtime wages (if any)
5. Add the regular wages to overtime wages (if any) to determine total wages for the week

Terminology

- Object
  - An entity in the problem statement
    - Can be *abstract* or *concrete*
  - A collection of data values and associated operations
- Algorithm
  - instructions for solving a problem in a finite amount of time using a finite amount of data
- Program
  - an algorithm written for a computer that defines classes of objects and orchestrates their interactions to solve a problem
  - objects work together to create an application (or program) that solves a problem

Implementation Phase: Programming

- Programming language
  - a language with strict grammatical rules, symbols, and special words used to construct a computer program
- Code
  - the product of translating an algorithm into a programming language
  - instructions for a computer that are written in a programming language

Implementation Phase: Testing

- Executing (running) your program on the computer, to see if it produces correct results
- If it does not, check the algorithm and/or code to find the error and fix it
- Finding known errors is called debugging

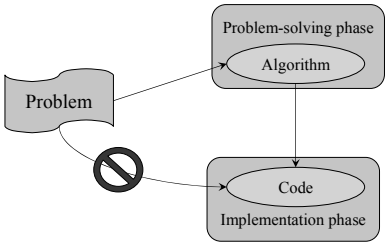
Maintenance Phase

- *Use* and *modify* the program to meet changing requirements or correct errors that show up in using it
- Maintenance begins when your program is put into use and accounts for the *majority of effort* on most programs
- Better design (problem-solving)
  == Easier maintenance

Software Life-Cycle (review)

- Problem-Solving and Design
  - Analysis and Specification
  - Design General Solution (Algorithm)
  - Verify
- Implementation
  - Concrete Solution (Code)
  - Test
- Maintenance
  - Use
  - Maintain

No Shortcuts!

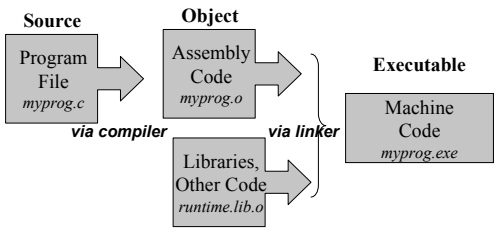


Programming Languages: High and Low

- Machine languages
  - Directly used by the computer
  - Binary-coded instructions
  - Not portable– only runs on one type of computer
- Assembly languages
  - Instruction mnemonics (ADD = 100101)
  - Still machine dependent
  - Translated to machine code by an assembler
- High-level languages
  - Portable (mostly)
    - Many are standardized by ISO/ANSI
  - Instructions are similar to natural language
  - Translated to assembly/machine code by a compiler
  - Examples: Java, C, Fortran, Pascal, Lisp, Scheme, …

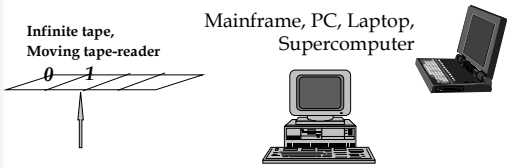
Compilation and Linking

- From high-level code to machine code



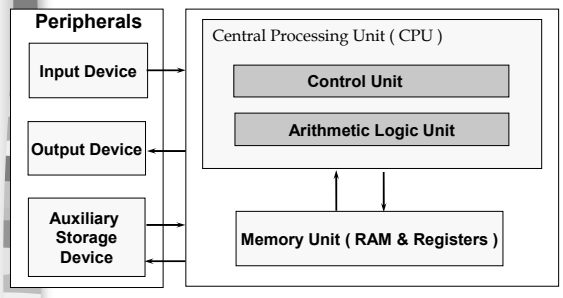
What is a Computer?

- Turing machine
  - Invented 1936 by Alan Turing as a theoretical model



- A computer is a computer, is a computer
  - All have the same "power" (Church-Turing thesis)

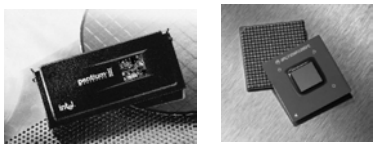
Basic Computer Components



Central Processing Unit (CPU)

- Heart and brain of the computer
- Two primary components
  - Arithmetic/Logic Unit (ALU) performs arithmetic and logical operations
  - Control Unit controls the order in which instructions in the program are executed

Pentium (left) and *PowerPC G3* chips



Memory

- An ordered sequence of storage cells, each capable of holding a piece of data
  - "Data" is a sequence of bits (on/off)
  - 8 bits = 1 byte (more numbers later on)
- Each memory cell has a distinct address
- The information held can be input data, computed values, or program instructions

Peripheral Devices

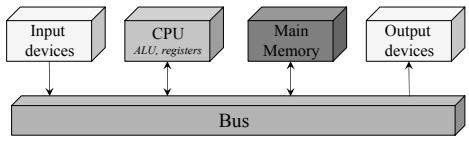
- Input Devices
  - keyboard
  - mouse
- Output Devices
  - printer
  - video display
  - LCD screen
- Auxiliary Storage
  - disk drive
  - CD-ROM drive
  - DVD-ROM drive

Summary: von Neuman Architecture

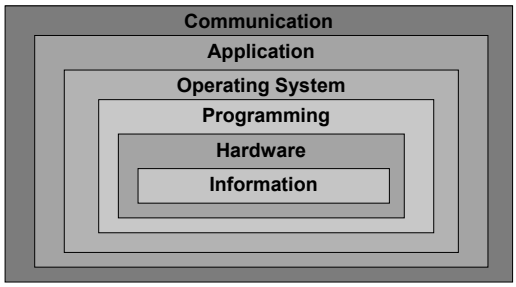
- John von Neumann (1940s)
- Still the basis of most computers today
- Data and instructions are logically the same and can be stored in the same place
- Separate information processing and storage
  - Memory holds both data and instructions
  - ALU performs arithmetic and logic operations
  - Input unit moves data from outside to inside
  - Output unit moves results from inside to outside
  - Control unit acts as stage manager to ensure all other components work in concert

Fetch-Execute Cycle

- Fetch the next instruction
- Decode the instruction into control signals
- Get data if needed (from memory)
- Execute the instruction



Layers of a Computing System



What makes programming fun?

What delights may its practitioner expect as a reward?

- **First is the sheer joy of making things**
- **Second is the pleasure of making things that are useful**
- **Third is the fascination of fashioning complex puzzle-like objects of interlocking moving parts**
- **Fourth is the joy of always learning**
- **Finally, there is the delight of working in such a tractable medium. The programmer, like the poet, works only slightly removed from pure thought-stuff.**

Fred Brooks

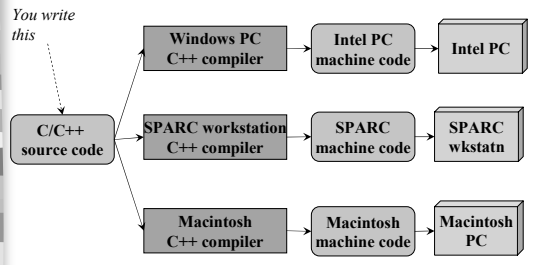
Distinctions

- **Hardware**
  - **Collection of physical components**
- **Software**
  - **Collection of programs providing instructions**
- **Information**
  - **any knowledge that can be communicated**
  - **any sensory perception that "inwardly forms" us**
- **Data**
  - **information that has been put into a form that is usable by the computer – transformed into quantifiable measurements**

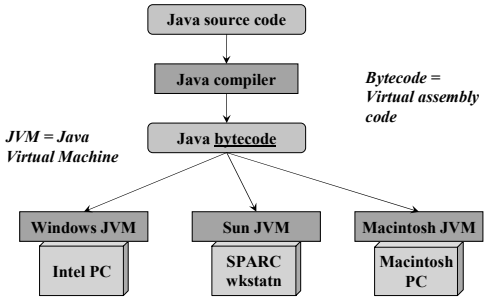
Recap

- **The discipline of Computer Science**
  - **Themes and concepts, subareas**
- **Software Life Cycle**
  - **Problem-Solving and Design**
  - **Implementation and Testing**
  - **Maintenance**
- **Hardware and Software**
  - **Computer components, fetch-execute cycle**
  - **Programming languages, high and low, compilation**
- **Coming up next…**
  - **Introduction to Java**
  - **Problem-Solving Techniques**
  - **Binary representation of data**

Compilation Framework (review)



Java Runtime Framework



JVM Interpreter

- JVM interprets Java *bytecode* on-the-fly
  - Takes one virtual assembly instruction and translates it to appropriate machine instruction(s) and immediately runs it (them).
- Achieves high-portability
  - Only compile source code once
  - Runs on any machine with a JVM interpreter
- Running an interpreter is like compiling and running one instruction at a time
  - Portability is a pro; can you think of any cons?

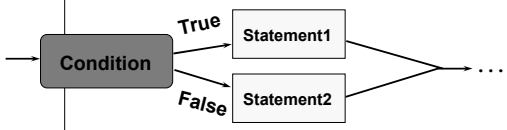
Java Programming: Basic Control Structures

- Sequence
  - series of statements that execute one after another
- Selection (branch)
  - executes different statements depending on certain conditions
- Loop (repetition)
  - repeats statements while certain conditions are met
- Subprogram
  - breaks the program into smaller units
- Asynchronous control *(won't cover this semester)*
  - handles events that originate outside our program, such as button clicks

Sequence

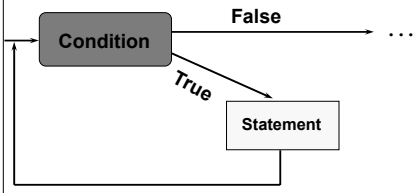
statement1 ; *statement2* ; *statement3* ; ...

Selection (branch)



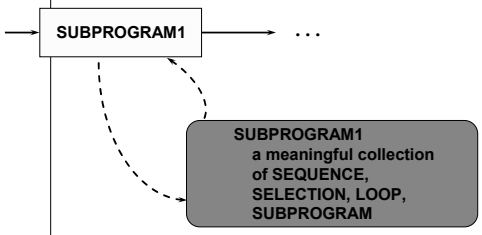
if *condition* **then** *statement1* **else** *statement2*

Loop (repetition)

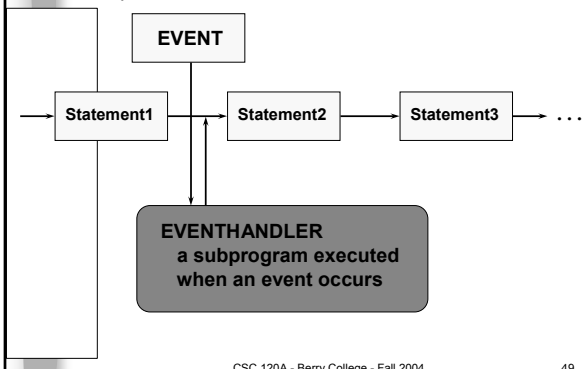


while *condition* **do** *statement* ; ...

Subprogram (function, method)



Asynchronous control



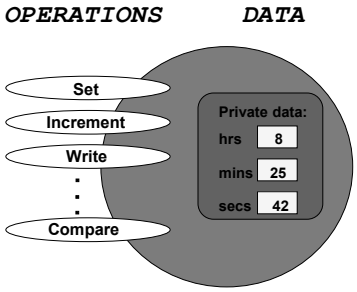
Object-Oriented Programming

- Data type
  - The specification in a programming language of how information is represented as data
  - Associated with a set of operations that can be performed on the data
- Object
  - Collection of data values and associated operations
- Class
  - A description for a set of objects

Classes

- A class in OO programming is a blueprint for creating new objects
- To instantiate a class means to create a new object based on the blueprint
- Classes can be organized into groups called packages. Packages make up a library.
  - Packages and libraries help share code to avoid "reinventing the wheel," among other things

An Object of Class "Time"



Java blueprint for "Time" objects

```
class Time {
  // data (fields)
  int hrs
  int mins;
  int secs;

  // operations (methods)
  public void set( ) { ... }

  public void increment () { ... }

  ...
}
```


Problem-Solving Techniques

- Ask questions
  - about the data, the process, the output, error conditions
- Look for familiar things
  - certain situations arise again and again
- Solve by analogy
  - it may give you a place to start
- Use means-ends analysis
  - Determine the I/O and then work out the details

More Problem-Solving Techniques

- Divide and conquer
  - break up large problems into manageable units
- Building-block approach
  - can you solve small pieces of the problem?
- Merge solutions
  - instead of joining them end to end to avoid duplicate steps
- Overcome mental block
  - by rewriting the problem in your own words

Numbers, numbers, numbers

- Number: A unit of an abstract mathematical system subject to the laws of arithmetic
- Natural number
  - The number 0 and any number obtained by repeatedly adding 1 to it
- Negative number
  - A value less than 0, with a sign opposite to its positive counterpart
- Integer
  - A natural number, a negative of a natural number, or zero *(yes, that definition is redundant)*
- Rational number
  - An integer or the quotient of two integers (excluding division by 0)

How Many Ones in 943?

- 943 = 9 hundreds + 4 tens + 3 ones
  = 900 ones + 40 ones + 3 ones
  = $9 \times 10^2 + 4 \times 10^1 + 3 \times 10^0$ (in base ten)
- Base
  - The foundational value of a number system, which dictates the number of digits and the value of digit positions
- Positional notation
  - The position of each digit has a place value
  - The number is equal to the sum of the products of each digit by its place value

Bits 'n Bytes

- Electric circuit states correspond to on (1) or off (0)
- Computers represent all data by combinations of 0s and 1s
- Numbers are represented using a binary, or base-2, system.
- Base 2
  - Only two digits 0, 1
  - Bit: a single digit
  - Byte: a group of 8 bits (an 8 digit binary number)
  - Word: a group of 16 (short), 32, or 64 (long) bits
- Letters ('A', 'a', 'B', …) are represented using one or two bytes

Binary Numbers

Decimal	Binary
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100

Binary Place Values

1 1 1 0

$1x2^3 + 1x2^2 + 1x2^1 + 0x2^0 =$
$1x8 + 1x4 + 1x2 + 0x1 =$
$8 + 4 + 2 + 0 = 14_{(decimal)}$

Numbers on the Computer

- Have limited size (number of bits)
  - With four decimal digits, the biggest number we can write is 9999_{10} *(subscript indicates base-10)*
  - With four bits (binary digits), the biggest number we can write is $1111_2 = 15_{10}$
- Computers have a scheme for representing negative numbers
  - The left-most bit can be used to tell the sign, but it's not exactly just a sign bit
- Java thus works with numbers in a range of values
  - *e.g.* from $-32{,}768_{10}$ to $32{,}767_{10}$
  - If a number gets too big (or small) it "wraps around"
- Keep this in mind (it can be a source of errors)

Interesting Number Base Systems

- Base-2 – Ancient Chinese world view (yin/yang)
- Base-5 – "Hand" (5 fingers on a hand)
- Base-10 – "Decimal" (10 fingers)
- 12 as a grouping system (Europe, China)
- Base-20 – Mayan culture (20 digits, fingers and toes)
- Base-60 – Sumerian culture, used as grouping number by many other cultures (has many factors)

Predictions that didn't make it!

- "I think there is a world market for maybe five computers." – Thomas Watson, IBM chair, 1943
- "Where … the ENIAC is equipped with 18,000 vacuum tubes and weighs 30 tons, computers in the future may have only 1,000 vacuum tubes and weigh only 1.5 tons." – *Popular Mechanics*, 1949
- "Folks, the Mac platform is through—totally." – John C. Dvorak, *PC Magazine*, 1998
- "There is no reason anyone would want a computer in their home." – Ken Olsen, Digital Equipment Corp. president, chairman, and founder, 1977
- "I predict the Internet .. will go spectacularly supernova and in 1996 catastrophically collapse." – Bob Metcalfe, 3Com founder, 1995

L. Kappelman, "The Future is Ours," CACM 44:3 (2001), p46