cis1.5
introduction to computing using c++
(legal applications)
fall 2008
lecture # I.1
introduction

instructor:

- Arif Tuna Ozgelen, *ozgelen@sci.brooklyn.cuny.edu*

course web page:

- `http://www.sci.brooklyn.cuny.edu/~ozgelen/cis1.5`

office hours:

- every Tuesday after class btw 12.30 pm - 1.30 pm. room# 4412

introduction to the course

- about this course
  - introduction to computer programming using the C++ language
  - uses **legal applications** as a *context* (i.e., the basis for examples and some of the lab exercises)
- the following topics will be covered in 6 units:

(I) Data and Output
(II) Control Structures and Input
(III) Functions
(IV) Arrays and Strings
(V) Searching and Sorting
(VI) Simple Classes

course structure

- 6 **units**
- each unit has:
  - 1-3 **lectures**
  - 2-3 **labs**
  - 1 **assessment**
- the labs will be hands-on sessions using laptops in the classroom (214 NE)
- the assessments will be:
  - programming assignments
- your grade = 6 assessments (9% each) + attendance (6%) + two midterms (20%) + one file exam (20%)

how to learn a programming language.

- YOU are responsible for your own learning!!!
- I will point you in the right direction...
- but YOU must PRACTICE, PRACTICE, PRACTICE...
- and PRACTICE some more!!!
- if you don't understand, then ASK for help!

today's topics

(1) computers: definition

(2) history of computing

(3) computer components

(4) programming

(5) C++ as a programming language

(6) our first C++ program

what is a computer?

- a computer is a machine that manipulates data according to a list of instructions.
- modern day computers contain *integrated circuits* making them powerful and small in size compared to their predecessors.
- most common form of computer in use today is the embedded computer, designed for a specific task. Usually programmable in limited ways. e.g. MP3 players, digital cameras, industrial robots, traffic lights, factory controllers.
- personal computers, however are extremely versatile due to their ability to store and execute a list of instructions called *programs*.
- a theoretical statement of versatility: Any computer with minimum capability is, in principle, capable of performing same tasks other computers can perform (Church-Turing thesis).

a little bit of history: early days.

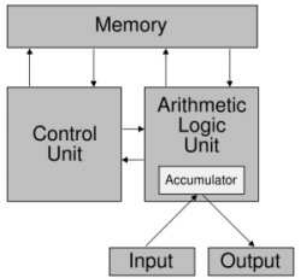
- originally, the term computer referred to a person who performed numerical calculations with a mechanical device such as an abacus or slide rule.
- modern history of computing begins with two important concepts: automated calculation and programmability.
- early examples of programmable mechanical devices:
  - Hero of Alexandria's mechanical theater (circa 10-70 AD) which performed a 10 minute play, operated by complex system of ropes and drums.
  - Joseph Marie Jacquard's textile loom which automatically weaves patterns on a punched paper card as a template (1801).
- first recognizable computer in concept and design: Charles Babbage's *Analytical Engine* in 1837. a fully programmable mechanical computer which was never built due to limited finances and Babbage's frequent minor changes on the design.
- Herman Hollerith's tabulating machines (1890), automated data processing on a large-scale. manufactured by Computing Tabulating Recording Corporation which later became IBM.

- tabulating machines used punch(ed) cards (IBM or Hollerith cards), a piece of stiff paper that contains a digital information represented by the presence of absence of holes in predefined positions. They were in use widely in 19th century textile looms and later in 20th century for data storage.
- technologies such as punch cards, boolean algebra, vacuum tube and teleprinters proved useful as practical computers begun to appear.

a little bit of history: first half of 20th century

- most of the scientific computing during the first half of 20th century, was done in analog computers which lacked versatility and accuracy.
- Claude Shannon's invention of most of digital electronics in 1937, led to modern computers.
- Konrad Zuse's electromechanical "Z machines" is the world's first operational computer (1941).
- U.S. Army's Ballistic Research Laboratory ENIAC (1946), which is referred as the world's first electronic computer, had a very inflexible architecture, which required rewiring to change its programming.
- recognizing its major flaws, a group of ENIAC developers came up with a far more flexible and elegant architecture known as *stored program architecture* or *von Neumann architecture*, formally described by John von Neumann in 1945.
- nearly all modern computers are built on some form of von Neumann architecture, which is the only trait most modern computers have in common.

von Neumann architecture



a little bit of history: modern computers

- during 1960's, vacuum tubes, which were then used in computers for amplifying, switching or creating an electrical signal, were replaced by transistor based electronics. Computers became faster, cheaper, more reliable, smaller and easier to build.
- during 1970's invention of integrated circuits and microprocessors (chip's), further decreased the cost and the size and increased reliability and speed of computers.
- in 1980's computers became sufficiently cheap and small to replace simple mechanical controls in appliances, also allowed wide usage in homes as personal computers (PC's).

computer components.

- computer = hardware + software
- a computer is organized into *logical units*:
  - input
  - output
  - memory
  - arithmetic and logic (ALU)
  - control unit
  - secondary storage

what is a program?

- a *computer program* is a set of instructions that tells the computer what to do
- a *computer programmer* is a person who writes those instructions
- there are many different *programming languages* that one can use to write computer programs—
  in this class, we will learn **C++**
- C++ is called a *high-level language* because:
  — it is kind of like English (no, really!)
  — well, it is more like English than the *low-level machine language* that the computer understands
- a *compiler* will translate a program from a high-level language into low-level machine language
- usually real world programs are large in size and created by many programmers therefore it is very unlikely for a program to be error or "bug" free.
- programming is generally is a cycle of editing the program file, called "source code", and fixing errors or "debugging".

Compilers and IDEs

- compilers are system dependent programs which translate source code into machine code. there are many different compilers for C++. you can use compilers separately, but in this class we are going to depend on an IDE to compile our programs.
- *integrated development environments (IDE)* are programs that provide facilities for software development to programmers, such as an editor, debugging, versioning, performance tools, a compiler etc.
- in class, we'll use a free, open source IDE called "CodeBlocks"
- some of the other IDE's you can use and also being used by other cis1.5 sections are "Dev C++" and "Eclipse"

getting started.

- programming is like solving puzzles
- think differently
- the world is now made up of
  objects
  and
  actions
- computer follows commands
  commands = series of instructions
- you will learn how to *command* a computer
  command = program = write instructions

computer instructions.

- set of instructions = *program*
- types of instructions:
  - machine language
  - assembly language
  - high-level language (e.g., C, C++, Java)
- program is *compiled* into machine language and then *executed (ran)*
- *executing (running) program = job = process = task*

machine language.

- lowest level
  – numeric
- computer is comprised of zillions of *switches* or *relays*
  – switches = ON or OFF
  – relays = OPEN or CLOSED
- hardware position is abstracted into software as 1's and 0's
- 1's and 0's $\Rightarrow$ *base 2*, or *binary*

assembly language.

- medium level, but still pretty low; i.e., hard to read and understand
- "English" words and abbreviations
- examples:
  LOAD
  ADD
  SHIFT
  STORE

high-level languages.

- examples: C, BASIC, FORTRAN, Pascal, C++, Java, LISP, Scheme
- even more like "English"
- high-level languages are
  1. *compiled* into machine language or *object code*
  2. *linked* into *executable code*
  3. *executed* or *ran* as programs

language examples.

- machine language:
  +1300042774
  +1400593419
  +1200274027
- assembly language:
  LOAD BASEPAY
  ADD OVERPAY
  STORE GROSSPAY
- high-level language:
  grossPay = basePay + overTimePay;

C++.

- C++ is an *object-oriented* language: it is structured around *objects* and *methods*, where a method is an action or something you do with the object
- C++ programs are divided into entities called *classes*
- some C++ classes are *native*
  but you can also write classes yourself
- C++ programs run as *applications*

our first c++ program.

"hello world"

- typical first program in any language
- output only (no input)

the application source code.

file name = hello.cpp

```
/*-------------------------------------------------------
   hello.cpp

   This class demonstrates output from a C++ application.
-------------------------------------------------------*/
#include <iostream>
using namespace std;

int main() {
  cout << "this is my c++ world\n";
  cout << "hello from inside of it!\n";
}
```


to do.

- get a copy of the textbook:
  Problem Solving with C++, J. Jones and K. Harrow, Addison Wesley
- ... and start to read chapter 1
- check out the class web page:
  `http://www.sci.brooklyn.cuny.edu/~ozgelen/cis1.5`

about you.

- please take out a piece of paper and write down...
  1. your name
  2. your class and major OR if you are a non-matriculating student, categorize yourself
  3. your background in computers, if any
- ...and give it to me before you leave