WELCOME TO CIS 1.5

Introduction to Computing using C++

Biomedical applications

- *Topics:*
  - Introduction to the course
  - What is a computer programming language?
  - What is biomedical computing?
- *Instructor:*
  - Prof Simon Parsons
  - *parsons@sci.brooklyn.cuny.edu*
- *Course web page:*
  - `http://www.sci.brooklyn.cuny.edu/~parsons/15-spring-2007`

Introduction to the course

- About this course
  - Introduction to computer programming using the C++ language
  - Uses *biomedical computing* 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 (4428 N)
- The assessments will be:
  - Programming assignments
- Your grade = 6 assessments (10% each) + 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!

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

Which compiler?

- There are lots of C++ compilers and programming environments
- In class, we'll use a free, open source *integrated development environment (IDE)* called "Code::Blocks" (we'll discuss this more in class next time)
- With an IDE, you can *edit* your computer program's "source files" and then compile the source files into an *executable application*; and finally you can run the application
- You can use a different IDE if you want to... (we'll talk about this more later)
- Some of the other CIS1.5 sections are using "Dev C++" and "Eclipse"

Getting started

- Programming is like solving puzzles
- Think differently
- The world is now made up of:
  - *objects*
  - *actions*
- Today's introductory topics:
  - Computer basics
  - Our first program

Computer commands

- Computer follows commands
  - *commands = series of instructions*
- You will learn how to *command* a computer
  - *command = program = write instructions*
- You understand the commands, but does the computer?
  - that's a question of cognition.
    → Artificial Intelligence, Cognitive Science

Computer components

- Computer = hardware + software
- A computer is organized into *logical units*:
  - input
  - output
  - memory
  - arithmetic and logic (ALU)
  - central processing (CPU)
  - secondary storage

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 *transistors* — *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, 30jan07/parsons

   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!
- Start to read chapter 1
- Check out the class web page:
  `http://www.sci.brooklyn.cuny.edu/~parsons/15-spring-2007`

About me

- Undergrad: University of Cambridge, Engineering, class of 1988
- Grad school: University of London, PhD 1993
- Previous teaching:
  - Queen Mary & Westfield College, London, UK.
  - University of Liverpool, UK.
  - Universidad Politechnica de Catalunya, Barcelona, Spain.
  - Universidad Nacional del Sur, Bahia Blanca, Argentina.
  - Columbia University.
- research interests:
  - Robotics;
  - Software agents and multi-agent systems; and
  - Rational action.

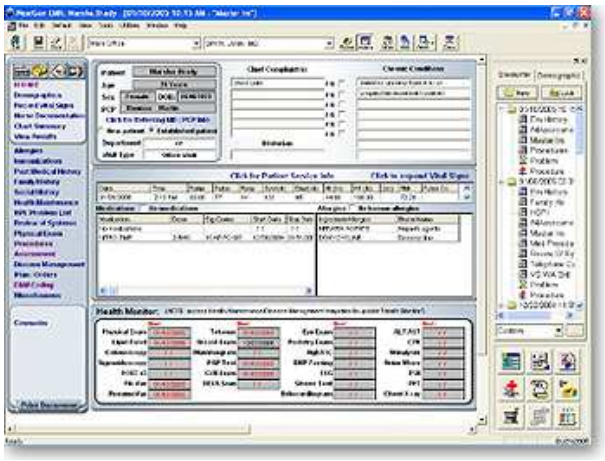
About you.

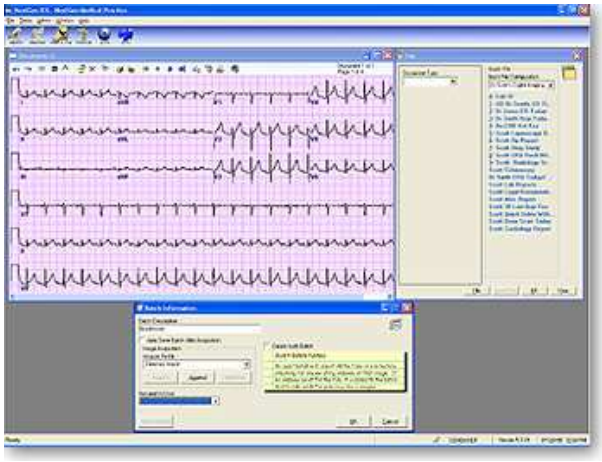
- Please take out a piece of paper and write down...
  1. Your name
  2. Your class and major OR if you are a non-matriculating student, categorize yourself
  3. Your background in computers, if any
  4. Why you are taking this course
  5. What you hope to get out of this course
  6. One sentence about one wonderful thing you did over the break
- ...and give it to me.

What is biomedical computing?

- Computer processing of medical data.
  - Support patient care.
  - Telesurgery.
  - Develop new therapies.
- Computer processing of biological data
  - Sequence analysis
  - Systems biology
- Biology-inspired computation.
  - New forms of computation

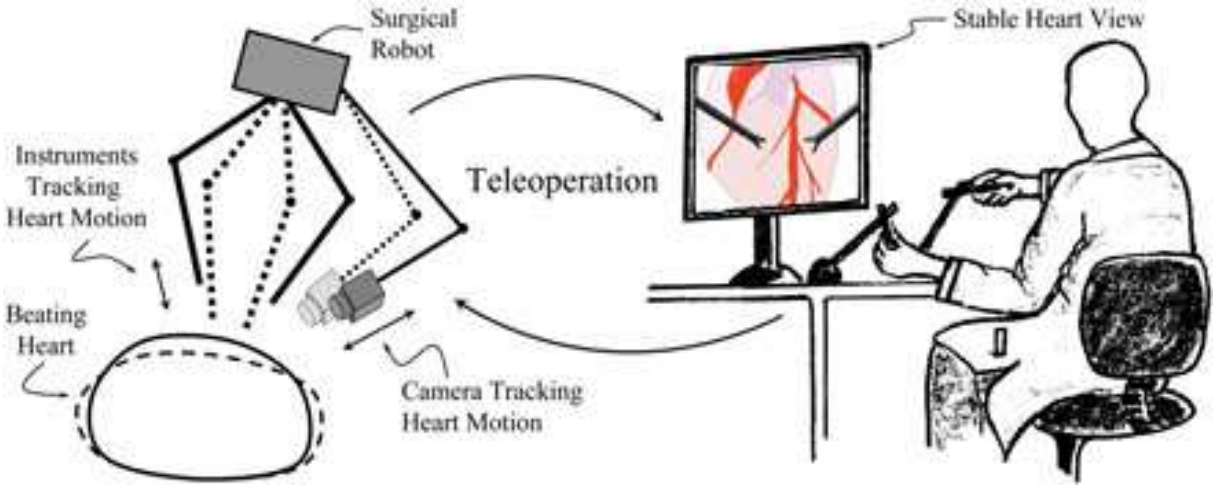
What is biomedical computing?





- Support patient care.
- Electronic medical records (EMR).

What is biomedical computing?



- Telesurgery.

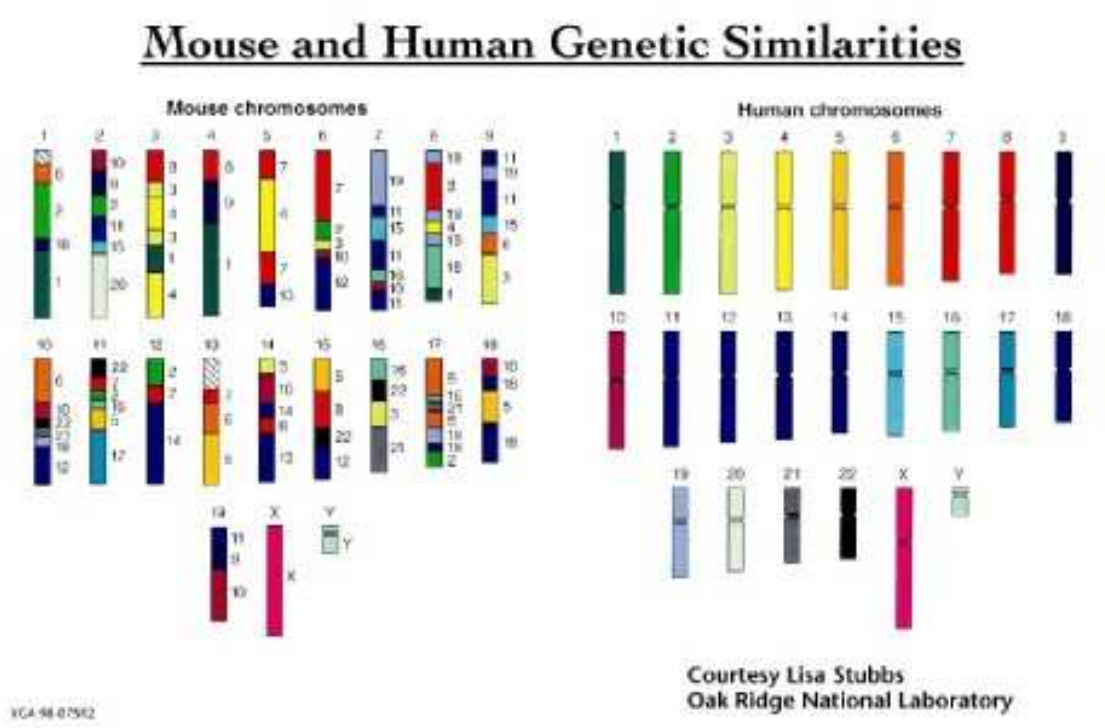
What is biomedical computing?

- Surgical robots.

What is biomedical computing?

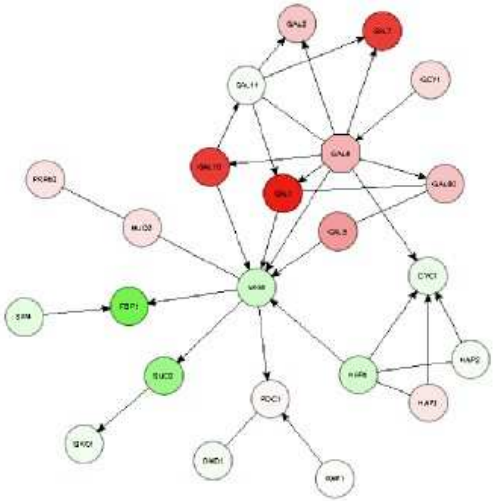
- Develop new therapies.
- Process EEG data.

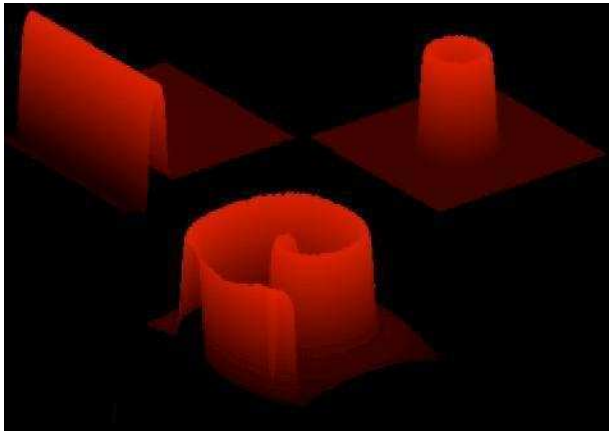
What is biomedical computing?



- Genome sequence analysis

What is biomedical computing?



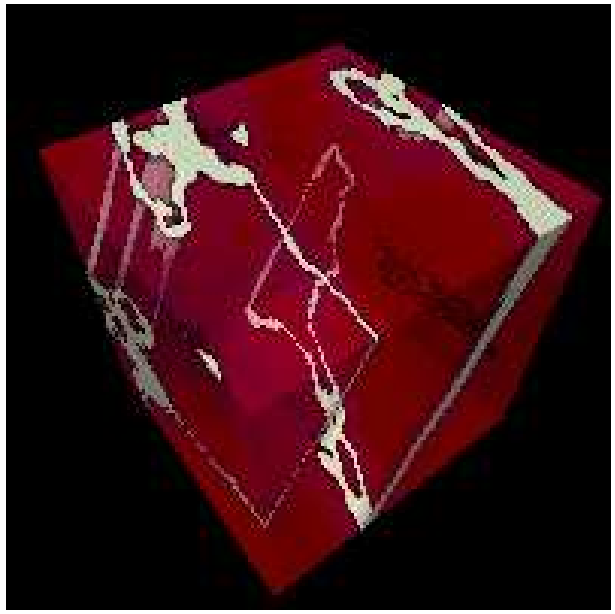


- Systems biology, simulation.

What is biomedical computing?

- Biology-inspired computation.
- Ant colony optimization.

What is biomedical computing?



- Genetic art.
- Kandid.

Summary

- This lecture has introduced the course.
- It has also talked about:
  - Basics of computer programming languages.
  - Described some of the context in which this course will be placed, that of biomedical computing.
- We will come back to the biomedical computing aspects later.