CSEN102 – Introduction to Computer Science

Topics:

Administrative Stuff

The Definition of Computer Science

Informal Definition of Algorithms

Prof. Dr. Slim Abdennadher

25.10.2007

Overview

What is CSEN102?

- Introduction to Computer Science principles
- A course with no prerequisites: no background in computer science needed

What isn't CSEN102?

- A Computer literacy course
- A programming course

Why should you learn CSEN102?

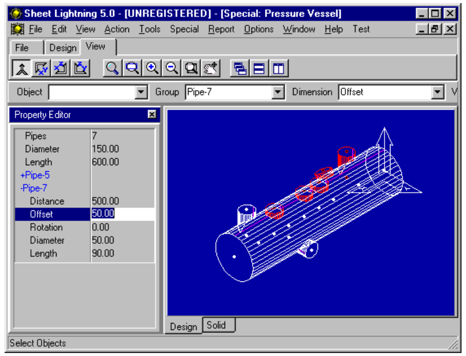
- To use computers for problem solving
- Acquire new skills that will allow you to create useful and customized computer-based applications
- Improve your problem solving skills (clarity, precision, logic, . . .)
- It is in the curriculum
- Acquire a useful vocabulary that will impress others in geeky conversations

Course Objectives

At the end of this course, you should be able to:

- **Demonstrate skills in problem solving**
  1. Express Problem solutions in the form of algorithms using pseudo-code
  2. Implement simple algorithms using Java
  3. Analyze algorithms in term of efficiency
- **Identify basic concepts in data representation and manipulation**
- **Build simple computer circuits using Boolean Logic**
- **Relate the concepts gained to understand the Von Neumann architecture**
- **Identify basic issues related to the software systems**

Applications

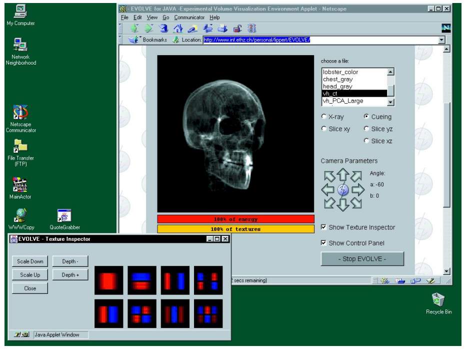

Sheet Lightning 5.0 - [UNREGISTERED] - [Special: Pressure Vessel]

Object: | Group: Pipe-7 | Dimension: Offset

Property Editor

Pipes	7
Diameter	150.00
Length	600.00
+Pipe-5	
-Pipe-7	
Distance	500.00
Offset	50.00
Rotation	0.00
Diameter	50.00
Length	90.00

Select Objects


EVOLVE for JAVA -Experimental Volume Visualization Environment Applet - Netscape

Location: http://www.inf.ethz.ch/personal/lippert/EVOLVE/

choose a file:
lobster_color
chest_gray
head_gray
vh_ct
vh_PCA_Large

○ X-ray ● Cueing
○ Slice xy ○ Slice yz
○ Slice xz

Camera Parameters
Angle:
a: -60
b: 0

100% of energy
100% of textures

☑ Show Texture Inspector
☑ Show Control Panel

EVOLVE - Texture Inspector

- **Engineering:**
  - Simulation models of the physical world
  - Signal and image processing
  - Graphical tools (CAD)
- **Medicine**
- **Business**
- **Art, Movies, Music**
- **Space Exploration**

Organization of the Course

- **Lectures:** 2 hours per week
- **Tutorials and Labs:** You have to attend!
  - Tutorials: 2 hours per week
  - Labs: 2 hours per week
- **Evaluation:** Grade is based on
  - a midterm and a final exam
  - 4 in-class quizzes
  - Assignments and lab test

Grading

Overall weighting for your grade

- **10%** for assignments
- **10%** for Lab test
- **15%** for quizzes
- **25%** for mid-term exam
- **40%** for final exam

Survival Guide

Tell me and I will forget;
show me and I may remember;
involve me and I will understand

Keep up with the course material

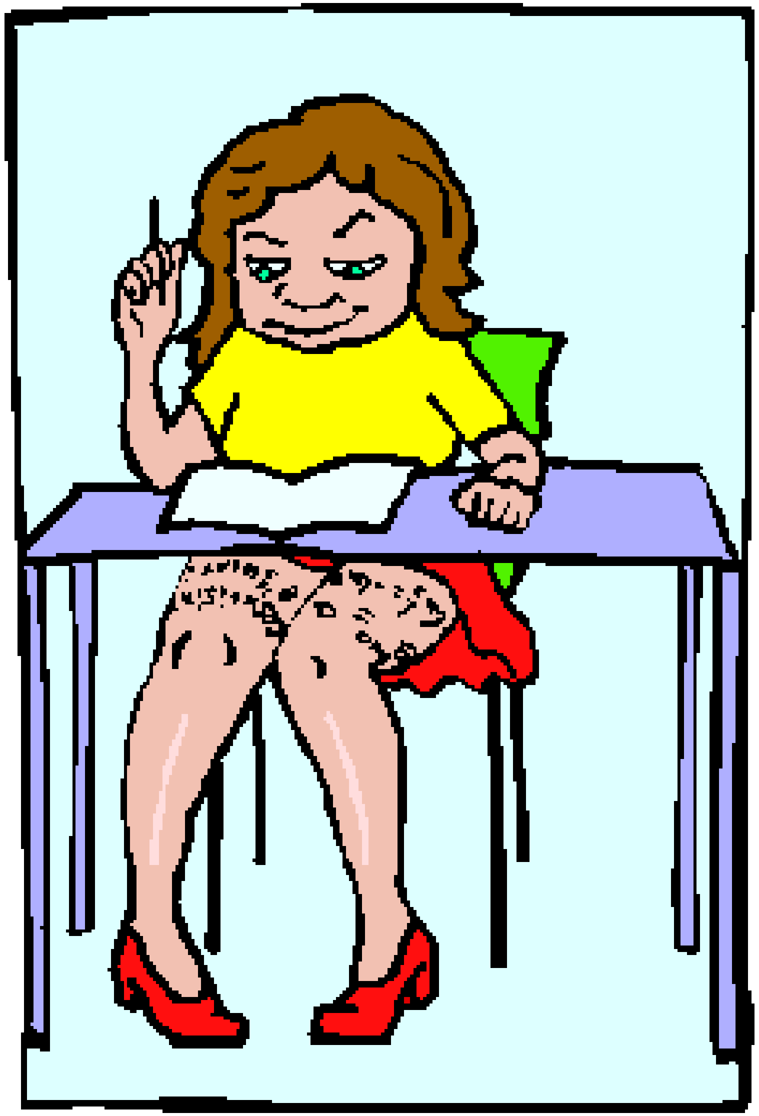
- Attend lectures and tutorials
- Participate in the discussions (be active)
- Solve the assignments and understand the model answers provided

Visit home page regularly for announcements and supplemental material

`http://www.cs.guc.edu.eg/`

Survival Guide

Do not copy !!!



Survival Guide

Keep in touch

- Email
- Office Hours

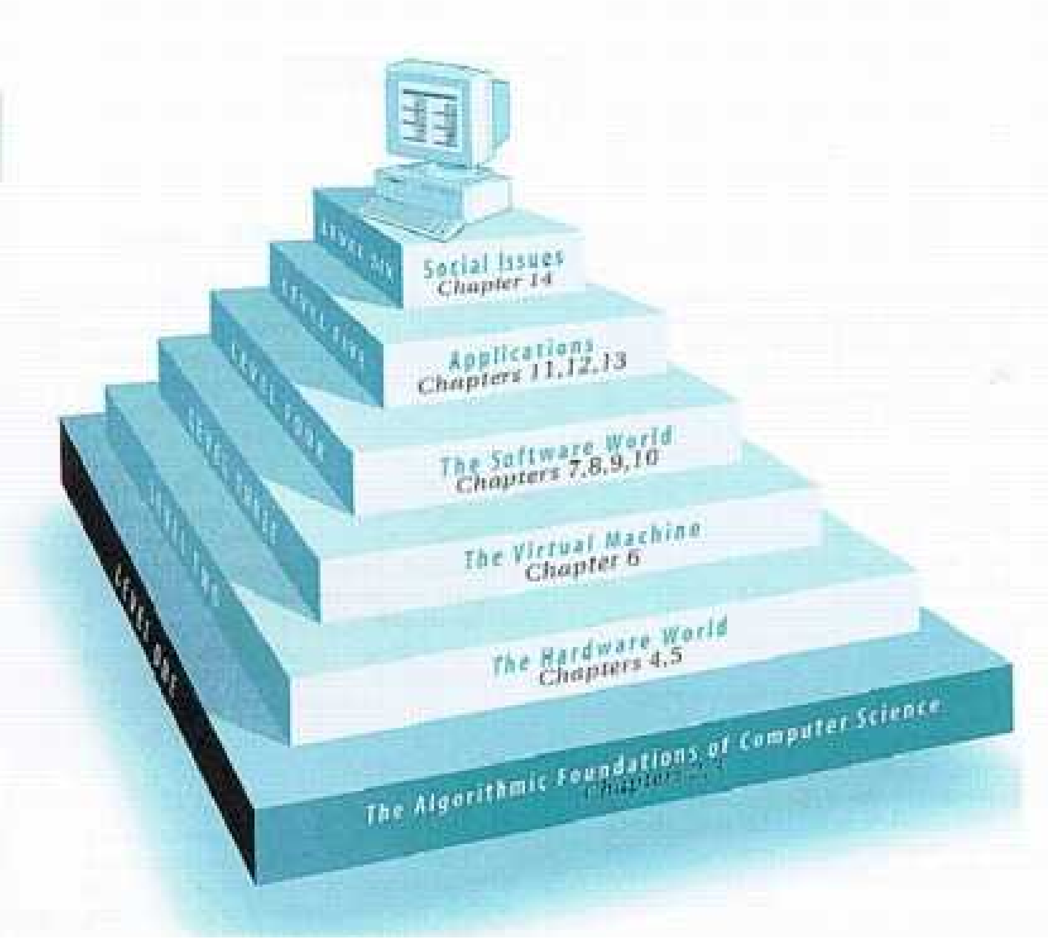
Ask for help when you need it

- Professor
- TAs

Structure of this Course

We will follow the pyramid of steps from Schneider and Gersting's Textbook

1. Algorithmic Foundations
2. Hardware World
3. Virtual Machines
4. Software World
5. Applications



End of Administrative Stuff

Complains??

What is Computer Science?

What Computer Science Isn't?

Or, more accurately, what it is much more than:

Computer Science is **NOT** restricted to

- **the study of computers:** computers are tools used in the field
- **the study of how to write computer programs:** Programming is an important part of computer science, but primarily as a a tool to implement ideas.
- **the study of the uses and applications of computers and software:** this only provides competency in tool usage

So, what is Computer Science?

What is Computer Science then?

Computer Science is the **study of algorithms**, including

- **Their formal and mathematical properties**
- **Their hardware realizations**
- **Their linguistic realizations**
- **Their applications**

This definition may seem a little puzzling, until we learn a bit more about algorithms.

OK, but What is an Algorithm?

Consider the following problem: We want to wash our hair twice.

Algorithm:

Step 1: Wet your hair

Step 2: Lather your hair

Step 3: Rinse your hair

Step 4: Lather your hair

Step 5: Rinse your hair

Step 6: Stop, you have finished shampooing your hair

Informally:

An **algorithm** is a step by step method for solving a problem

Algorithms

- Algorithms are not necessarily limited to simple tasks.
- We use **algorithms** all the time in our daily life, for example:
  - Cooking recipes
  - Directions how to get to places
  - Performing mathematical tasks such as:
    - Calculate the students' GPA
    - Calculate the interests for invested money in a bank
    - . . . .

An Algorithm for Calculating the area of a square

```
Step 1. Get the value of Side

Step 2. Area = Side * Side

Step 3. Print the value of Area
```

Why is this important?

If we can specify an algorithm to solve a problem, then we can **automate** its solution.

- A **computing agent** is an entity capable of performing the steps described in the algorithm, that is, execute the algorithm
- Could be a:
  - Person
  - Robot
  - Computer
- In our case, typically a computer.

Why use a computer?

.

- **Computers are fast:** they can perform operations without errors at speed unattainable by human beings
- **They can store very large amount of information:** Human beings have a difficulty managing and keeping track of a large number of objects
- **They are not task specific:** they can be programmed to perform different tasks. Most other tools can do only one thing.
- **Their tasks can be automated:** computers are excellent at performing the same task over and over again on similar pieces of data (ex: preparing payment bills for every mobile phone user)

Definition of Computer Science

Computer Science is the **study of algorithms**, including

- **Their formal and mathematical properties**
  - **How to design algorithms to solve. a wide range of problems**
  - **How to determine whether problems are (efficiently) computable**
  - **Studying the behavior of algorithms**
- **Their hardware realizations**
  - **Designing and building computer systems**
- **Their linguistic realizations**
  - **Designing programming languages and translating algorithms so they can be executed by the hardware**
- **Their applications**
  - **Identifying important problems for computers**
  - **Designing software to solve these problems**

Is any Step-by-Step Procedure an Algorithm?

- Instructions how to use a shampoo bottle
  - Step1: Wet hair
  - Step2: Lather
  - Step3: Rinse
  - Step4: Repeat
- Make the crust
- Write out the exact decimal value of π
- Make a list of all positive integers

What is an Algorithm?

Formal Definition:

An algorithm is a **well-ordered** *collection of* **unambiguous** *and* **effectively computable operations** *that, when executed,* **produces a result** *and* **halts in a finite amount of time***.*

- An algorithm is **well-ordered**: each step of the algorithm is executed in the order in which it is written, or else the order is clearly stated.
- An algorithm is **unambiguous**: The algorithm must be clearly stated, in terms that the computing agent (e.g computer) understands
- An algorithm is **effectively computable**: It must be possible for the computing agent to perform the operation and produce a result
- An algorithm must **halt in a finite amount of time**: must even if it would take centuries to finish

Algorithm: Historic Roots

Named after the Persian mathematician **Muhammad Ibn Musa Al-Khwarismi**

- 780-850 in Khwarism (today Khiva), Usbekistan
- developed a strategy for calculating heritage proportions for rich Arabians with four woman using algebraic methods
- His name was turned into **Algorism** and that evolved **Algorithm**

Algorithm: Historic Occurrence

- The oldest known algorithm is probably **Euclid's Algorithm** to determine the greatest common divisor (GCD) of two integers (circa 365-275 B.C)
- **Method**: To find the GCD of two numbers, repeatedly replace the larger by subtracting the smaller from it until the two numbers are equal.
  - Only Subtraction and comparison operations are needed.
- **Example:** GCD of 132 and 168

132	168
132	36
96	36
60	36
24	36
24	12
12	12

So the GCD of 132 and 168 is 12.