Introduction to C++

Giuseppe Lipari
http://retis.sssup.it/~lipari

Scuola Superiore Sant'Anna – Pisa

April 27, 2009

Course content

This course will cover advanced programming concepts with C/C++ language. It is divided in three parts:

- Part 1:
  - Intro to programming and software engineering
  - Recollection of C++ basics
  - Classes, Inheritance
  - Templates
  - The Standard Templates Library
- Part 2: Patterns
  - Basic Patterns
  - Singleton, Composite
  - Abstract factory, Prototype, Builder
  - Visitor, Adaptor, Observer
- Parte 3: Advanced Libraries and tools
  - Overview of the Boost Library
  - Unit testing with CPPUNIT
  - Autotools
  - Metaprogramming techniques

Books and material

- Books on C++
  - Stroustup (Language reference)
  - Bruce Eckel (Basic of C++)
  - Exceptional C++ (Tips and Tricks)
  - Modern C++ programming
- Books on patterns
  - Design Patterns
  - Extreme Programming
- Web resources
  - STL reference
  - Boost library
  - Guru of the week
- Slides
  - Part of the slides are courtesy of Alex Liu (Associate Professor at Michigan State University MSU) `alexliu@cs.msu.edu`

Exam

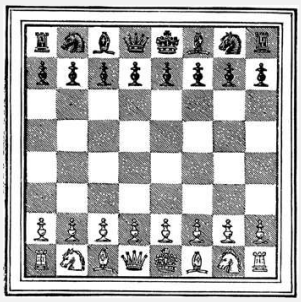
- The class can bedivided in groups of 2-3 students each
- During the course, I will give some assignments. The assignment can be done (at least partially) during the lab lectures.
- The assignments may be fragments (i.e. modules, classes) where I will ask you to apply some of the techniques studied during the course
- At the end, each group will produce a part of a program, and we will try to integrate everything to realize one single program.
- Grading:
  - 50% will be on the intermediate assignments
  - 40% on the final project
  - 10% will be on the integration

How to become a good software designer

- How to become a software design master?
  - Engineering approach
  - Lot of experience
- Learning to develop good software is similar to learning to play good chess

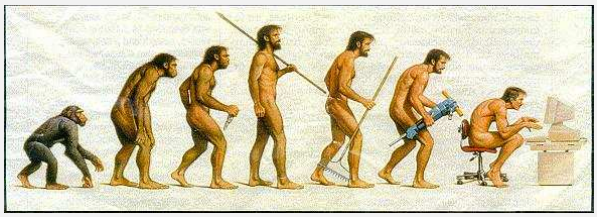
How to become a chess master

- First, learn the rules
  - e.g., names of pieces, legal movements, captures, board geometry, etc.
- Second, learn the principles
  - e.g., relative value of certain pieces, power of a threat, etc.
  - But principles are abstract. How to apply them in practice?
- Third, learn the patterns by studying games of other masters
  - These games have certain patterns that must be understood, memorized, and applied repeatedly until they become second nature.



To become a software design master

- First, learn the rules
  - e.g., programming languages, data structures, etc.
- Second, learn the principles
  - e.g., software engineering principles such as separation of concerns, etc.
  - But principles are abstract. How to apply them in practice?
- Third, learn the patterns by studying designs of other masters
  - These designs have certain patterns that must be understood, memorized, and applied repeatedly until they become second nature.



Organization of the course

- Review of C++ syntax (learn the rules)
- C++ programming techniques and Software Tools (learn the principles)
- Object Oriented Design Patterns (learn the patterns)