FACULTY OF **ENGINEERING**

DEGREE COURSE: **COMPUTER AND CONTROL ENGINEERING BS**

SUBJECT: ALGORITHMS AND DATA STRUCTURES

LECTURER: LUIGI SARTI

E-mail: luigi.sarti@uniecampus.it

OBJECTIVES

The course is aimed at:

1) improving students' skills in understanding, evaluating, selecting and developing algorithms using state-of-the-art programming languages and tools

CONTENTS

Introduction to the course
Computer Science Foundations

- Introduction to the analysis of algorithms
  - *Complexity asymptotic analysis*
  - *O, Ω, Θ formalisms*
  - *Recurrence relation as an analytic method*
- Algoritmic strategies
  - *Brute force*
  - *Greedy techniques*
  - *Backtracking*
  - *Binary tree representations*
- Computing algorithms
  - Simple numeric algorithms
  - Sequential and binary search
  - Sorting
  - Hash tables and collision handling strategies
  - Binary trees as abstract data
  - Depth-first and breadth-first visits
  - Search trees
  - Stacks and queues
  - Shortest paths algorithms (Dijkstra, Floyd)
  - Minimum spanning tree (Prim e Kruskal)
  - Transitive closures (Floyd)
  - Coding and decoding algorithms
  - Data compression algorithms (Ziv-Lempel)
- Automata, grammars and languages
  - *Alphabets, strings and languages*
  - *Finite state machines and regular expressions*

Context independent grammars
Stack-based automata
Context independent grammars
LL(1) grammars and top-down parsing

Programming

Programming paradigms
Functional programming
Object-oriented programming
Encapsulation and information-hiding
Methods to separate interface and implementation
Classes, subclasses and inheritance
High level languages syntax and semantics: variables, types, expressions and assignment
I/O operations
Selective and iterative control structures
Functional abstraction and parameter passing

Data structures
Built-in types
Arrays
Records
Data representation in memory
Static, automatic and dynamic allocation
Memory management at run-time
Pointers and references
Linked structures
Implementation of stacks, queues, hash tables, graphs, trees
Data persistence: files, streams, file system

Recursion
Mathematical recursive functions

Program translation
Interpreters and compilers
Translation steps (lexical analysis, parsing, code generation, optimization)
Virtual machines
Intermediate languages

Declarations, types, abstraction
Types as sets of values vs. sets of operations
Binding, visibility, accessibility e lifetime of values
Type-checking
Iterators
Parameter types

Object oriented programming
Polymorphism
Class hierarchies
Collections
Design patterns

Event programming
Event handling
Concurrency
Exception handling

Functional programming
Overview of functional languages

Recursion on lists, natural numbers, trees and recursively defined data
Closures
Lazy programming

Overview of emergent programming languages

Scala
Clojure
Erlang

Final exercise

XML documents parsing

Conclusions

LEARNING OUTCOMES

At the end of the course, students will:

- be aware of methodological and operational issues related to the development and selection of computing algorithms;
- be able to solve problem using state-of-the-art methods, techniques, formalisms, languages and tools;
- be aware of the potential and the limits of primary programming paradigms and languages;
- improve their basic cognitive means for lifelong self-training.

ASSESSMENT

Written exam: multiple choice and open questions

RECOMMENDED TEXTBOOKS

Eckel, Bruce (2006). Thinking in Java - 4th ed., Pearson Education, ISBN 0-13-187248-6.
http://www.mindview.net