Course guide
340380 - PROP-I4O23 - Programming Project

Last modified: 16/05/2024

Unit in charge: Vilanova i la Geltrú School of Engineering
Teaching unit: 723 - CS - Department of Computer Science.

Degree: BACHELOR'S DEGREE IN INFORMATICS ENGINEERING (Syllabus 2018). (Compulsory subject).

Academic year: 2024 **ECTS Credits:** 6.0 **Languages:** Catalan

LECTURER

Coordinating lecturer: Orellana Bech, Bernat

Others: Orellana Bech, Bernat

PRIOR SKILLS

Knowledge of programming and data structures:

- Ability to solve algorithmic problems of medium difficulty from a clear specification, and implement solutions in an imperative programming language.

- Knowledge of basic mechanisms for structuring programs (modularization, encapsulation, abstract data types, classes) and ability to apply them to problems small-sized (a few modules)

- Knowledge of the elements of object oriented programming (classes, objects, mechanisms for implementation).

- Familiarity with object-oriented imperative language.

- Ability to use data structures and programming in this language.

- Ability to use language in this book.

- Mastery of basic strategies for finding and correcting errors in simple modules.

DEGREE COMPETENCES TO WHICH THE SUBJECT CONTRIBUTES

Specific:
1. CEC01. Ability to have a thorough understanding of the fundamental principles and models of computation, ability to apply the principles to interpret, select, evaluate, model, and create new concepts, theories, applications and advance the technological development related to computing.
2. CECO2. Ability to understand theoretical basics of programming languages and techniques of lexical, syntactic and semantic associates processing, and apply them to create, design and process languages.
3. CEC03. Ability to assess the computational complexity of a problem, to know algorithmic strategies that may lead to its resolution and to recommend, develop and implement the one which guarantees the best performance according to established requirements.
4. CECO4. Ability to learn basics, paradigms and techniques of intelligent systems and analyze, design and build systems, services and computing applications that use these techniques in any scope.

TEACHING METHODOLOGY

In the course we work is algorithmic programming techniques through lectures and laboratory classes. In the laboratory classes we look at object oriented programming in a practice, developing programming activities to establish these techniques and finally developing a project of average size for which students must develop the techniques learned in lectures and combine them with object-oriented programming techniques that have been in the laboratory classes.

LEARNING OBJECTIVES OF THE SUBJECT

Learning techniques to identify the complexity of a problem and apply the appropriate resolution strategy.
Estrucutra learning of graph to represent combinatorial problems.
Learning the different algorithmic strategies for solving computational problems.
Learn advanced concepts of OOP.

STUDY LOAD

Type	Hours	Percentage
Hours small group	30,0	20.00
Hours large group	30,0	20.00
Self study	90,0	60.00

Total learning time: 150 h

CONTENTS

Graphs

Description:
Review of graph's theory. Graph representation for computing: adjacency matrix, adjacency lists. Classical graph algorithms. graph traversal.

Specific objectives:
(ENG) Aprenentatge de la importància dels grafs per representar problemes on hi ha relacions entre elements.
Aprenentatge de diferents estructures de representació pels grafs i la eficiència espacial i temporal de les mateixes
Disseny d'algorisme de connectivitat entre nodes del graf: Camins, parts connexes d'un graf, cicles, etc.

Full-or-part-time: 1h
Theory classes: 1h

Algorithmic costs

Description:
Measurement of temporal and spatial cost. Hierarchy of problems according to their complexity. Treatable and intractable problems. NP problems: known cases. Reduction technique: identification of NP problems.

Specific objectives:
(ENG) Adquirir i el concepte de cost temporal i espaial d'un problema. Aprenentatge de la jerarquia de complexitat. Capacitat de discernir un problema tractable d'un d'intractable. Aprenentage de la tècnica de reducció.

Full-or-part-time: 3h
Theory classes: 3h

Combinatorial Algorithms

Description:
Application of generative algorithms. Search for any solution or for the optimal solution. Description of the search space. Generative search scheme.Domain and problem definition languages (STRIPS, PDDL). Blind search. Heuristic functions. Informed search. Game search algorithms: Minimax, alpha-beta, Monte Carlo.

Full-or-part-time: 26h
Theory classes: 14h
Laboratory classes: 4h
Self study : 8h

Greedy algorithms

Description:
Scheme of the greedy algorithms. Greedy criterion: the need for optimality demonstration. Example cases.

Specific objectives:
(ENG) Aprenentatge de l'estategia dels algorísmes voraços i identificació de problemes que admeten aquestes solucions.

Full-or-part-time: 12h
Theory classes: 4h
Self study : 8h

Dynamic Programming

Description:
Inefficient recursive programs. Memorization. Steps of dynamic programming. Recursive equations.

Full-or-part-time: 6h
Theory classes: 6h

Agregations and divide and conquer

Description:
Divide and conquer scheme. Aggregation and elimination scheme.

Specific objectives:
(ENG) Aprenentatge de l'estategia d'agregació i de divideix i venç i identificació de problemes que admeten aquestes solucions.

Full-or-part-time: 2h
Theory classes: 2h

Object Oriented Language: Java

Description:
Hands-on learning of the Java language

Specific objectives:
Hands-on learning of the Java language

Full-or-part-time: 5h
Laboratory classes: 5h

Advanced Object Oriented Programming

Description:
Learning advanced features of object-oriented programming.

Specific objectives:
Learning advanced features of object-oriented programming.

Full-or-part-time: 3h
Laboratory classes: 3h

ACTIVITIES

Activity 1

Description:
Activities using preprogrammed classes and their extension, applied to games and searches. Javadoc usage. Development with inheritance and polymorphism.

Related competencies :
I_CECO2. CECO2. Ability to understand theoretical basics of programming languages and techniques of lexical, syntactic and semantic associates processing, and apply them to create, design and process languages.
I_CECO4. CECO4. Ability to learn basics, paradigms and techniques of intelligent systems and analyze, design and build systems, services and computing applications that use these techniques in any scope.

Full-or-part-time: 21h
Laboratory classes: 5h
Self study: 16h

Activity 2

Description:
Object-Oriented Programming of combinatorial algorithms for games.

Related competencies :
I_CECO1. CEC01. Ability to have a thorough understanding of the fundamental principles and models of computation, ability to apply the principles to interpret, select, evaluate, model, and create new concepts, theories, applications and advance the technological development related to computing.
I_CECO4. CECO4. Ability to learn basics, paradigms and techniques of intelligent systems and analyze, design and build systems, services and computing applications that use these techniques in any scope.
I_CECO2. CECO2. Ability to understand theoretical basics of programming languages and techniques of lexical, syntactic and semantic associates processing, and apply them to create, design and process languages.

Full-or-part-time: 23h
Laboratory classes: 5h
Self study: 18h

Project

Description:
Programming of a game AI, using algorithms based on MiniMax, alpha-beta pruning, Iterative deepening, transposition tables and heuristic calculations that include the use of dynamic and/or combinatorial and/or greedy programming algorithms. Use of UML modeling tools and Javadoc . Use of Git repository for team working.

Related competencies :
I_CECO4. CECO4. Ability to learn basics, paradigms and techniques of intelligent systems and analyze, design and build systems, services and computing applications that use these techniques in any scope.
I_CECO1. CEC01. Ability to have a thorough understanding of the fundamental principles and models of computation, ability to apply the principles to interpret, select, evaluate, model, and create new concepts, theories, applications and advance the technological development related to computing.
I_CECO3. CEC03. Ability to assess the computational complexity of a problem, to know algorithmic strategies that may lead to its resolution and to recommend, develop and implement the one which guarantees the best performance according to established requirements.

Full-or-part-time: 48h 40m
Laboratory classes: 8h 40m
Self study: 40h

GRADING SYSTEM

Theory Grade = max(0,5 Exam1 + 0,5 Exam2; second-chance examination)
Small projects grade = 0,5 Small project 1 + 0,5 small project 2

IF Theory Grade>= 3 then Final Grade = 0,5 Theory + 0,3 Big project + 0,2 Small projects
Else Final Grade = 0,7 Theory + 0,2 Big project + 0,1 Small projects

BIBLIOGRAPHY

Basic:
- Cormen, Thomas H. ; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford. Introduction to algorithms [on line]. 3rd. Cambridge: MIT Press, 2009 [Consultation: 15/02/2024]. Available on: https://search-ebscohost-com.recursos.biblioteca.upc.edu/login.aspx?direct=true&AuthType=ip,uid&db=nlebk&AN=2932690&site=ehost-live&ebv=EK&ppid=Page-__-1. ISBN 9780262033848.
- Levitin, Anany. Introduction to the design and analysis of algorithms. 3rd. Boston: Pearson Addison-Wesley, 2012. ISBN 9780273764113.
- Edmonds, Jeff. How to think about algorithms. 2nd ed. Cambridge: Cambridge University Press, 2024. ISBN 9781009302135.
- Skiena, Steven S. The Algorithm design manual. 3th ed. Cham: Springer, 2020. ISBN 9783030542559.
- Savitch, Walter. Java : an introduction to problem solving and programming. 8th ed. New York: Pearson, 2019. ISBN 9781292247472.
- Wu, C. Thomas. An Introduction to object-oriented programming with JAVA. 5th. Boston [etc.]: McGraw-Hill, 2009. ISBN 9780073523309.

RESOURCES

Other resources:
The following material is available on the digital campus:
- Class slides
- Code samples
- Solved exams from previous courses