DEGREE CURRICULUM

ALGORITHMS AND PROGRAMMING

Coordination: MARTÍNEZ RODRÍGUEZ, SANTIAGO

Academic year 2018-19

2018-19

Subject's general information

<table>
<tr><td>Subject name</td><td>ALGORITHMS AND PROGRAMMING</td></tr>
<tr><td>Code</td><td>102364</td></tr>
<tr><td>Semester</td><td>1st Q(SEMESTER) CONTINUED EVALUATION</td></tr>
<tr><td>Typology</td><td>
<table>
<tr><th>Degree</th><th>Course</th><th>Character</th><th>Modality</th></tr>
<tr><td>Bachelor's degree in Digital Interaction and Computing Techniques</td><td>1</td><td>COMMON</td><td>Attendance-based</td></tr>
</table>
</td></tr>
<tr><td>Course number of credits (ECTS)</td><td>6</td></tr>
<tr><td>Type of activity, credits, and groups</td><td>
<table>
<tr><th>Activity type</th><th>PRAULA</th><th>TEORIA</th></tr>
<tr><td>Number of credits</td><td>3</td><td>3</td></tr>
<tr><td>Number of groups</td><td>2</td><td>1</td></tr>
</table>
</td></tr>
<tr><td>Coordination</td><td>MARTÍNEZ RODRÍGUEZ, SANTIAGO</td></tr>
<tr><td>Department</td><td>COMPUTER SCIENCE AND INDUSTRIAL ENGINEERING</td></tr>
<tr><td>Teaching load distribution between lectures and independent student work</td><td>6 ECTS = 25x6 = 150 working hours:
40% -> 60 in-class hours,
60% -> 90 autonomous work hours.</td></tr>
<tr><td>Important information on data processing</td><td>Consult this link for more information.</td></tr>
<tr><td>Language</td><td>Catalan</td></tr>
</table>

Teaching staff	E-mail addresses	Credits taught by teacher	Office and hour of attention
MARTÍNEZ RODRÍGUEZ, SANTIAGO	santi@matematica.udl.cat	6	
ORELLANA BECH, BERNAT	bernat.orellana@udl.cat	3	

Subject's extra information

To address the subject is advisable to show interest in analyzing real problems and developing technological solutions to solve them. It is also advisable to show analytical skills, logical reasoning and critical capacity.

The knowledge and competencies acquired in this subject will be useful to follow other subjects with contents related with programming languages, data structure and algorithms.

Learning objectives

The learning objectives of the course are to design algorithms, and then to implement these algorithms with a programming language. Specifically, the programming language used for this purpose is ANSI C/C++ and the problems to be solved are mainly those related with sequences processing.

In particular, the main learning objectives are:

- To design and implement algorithmic structures to solve the different types of problems.
- To design and implement data structures to encode information.
- To design and implement iterative algorithms.
- To identify problem types and to apply appropriate algorithmic strategies.
- To design and implement algorithms to solve complex problems in a structured way.
- To design and implement solutions using the top-down design technique.
- To use a software development environment based on a high-level programming language.

Competences

Basic Competences

- **B01**. That students have demonstrated to possess and understand knowledge in an area of study that starts from the base of general secondary education, and is usually found at a level that, although supported by advanced textbooks, also includes some aspects that imply knowledge coming from the vanguard of his/her field of study.

Transversal Competences

- **CT3**. Acquire training in the use of new technologies and information and communication technologies.
- **CT5**. Acquire essential notions of scientific thought.

General Competences

- **CG2**. Design, develop, evaluate and guarantee the accessibility, ergonomics, usability and security of computer systems.
- **CG3**. Use adequate hardware and software platforms to develop and execute interactive digital applications.
- **CG5**. Know the basic subject areas and technologies needed to learn and develop new methods and technologies, and those that help to adapt to new situations.
- **CG7**. Solve problems through initiative, determination, independence and creativity.
- **CG8**. Capacity for abstraction and critical, logical and mathematical reasoning.

Specific Competences

- **CE2**. Capacity to understand and master the basic concepts of discrete mathematics, logics, algorithmic and computational complexity, and its application to solve computational problems.

- **CE3**. Basic knowledge of the use and programming of computers, operating systems and databases, and their use in the development of interactive applications.
- **CE4**. Capacity to know, understand and evaluate the structure and architecture of computers, as well as the basic components that conform them.
- **CE16**. Capacity to design and evaluate person-computer interfaces that guarantee the usability of systems, services and computer applications.
- **CE17**. Capacity to apply knowledge on design to propose and defend a design concept for an interactive system and use proper creative technologies to develop each project.
- **CE24**. Capacity to understand the human factors involved in any interactive process between humans and technology, as well as being able to adequately apply them in the design of interactive products and services, and their interfaces.

Subject contents

Introduction: Processes, algorithms and programs.

Unit 1. Basic algorithmic structures

1.1 Constants, variables, basic types and valid expressions
1.2 Assignment, sequential composition, alternative composition and iterative composition
1.3 Programming environment

Unit 2. Design of iterative programs

2.1 Sequential access

- Algorithmic schemes for sequence processing
- Algorithmic schemes for searching in sequences

2.2 Direct access. Tables

- Sequential tables
- Direct tables
- Multidimensional tables
- Classic sorting algorithms

Unit 3. Non-basic data processing

3.1 Procedures and functions
3.2 Parameter passing mechanisms
3.3 Top-down design of algorithms

Methodology

Each week students attend 2 hours with a Large group and 2 hours with a Medium Group. Medium Group sessions are practices.

Large Group: Theory and Problems (3 ECTS)

- Theory: classes supported with slides and/or notes.
- Part of practical application: always work with problems and programming exercises.

Medium Group: Practices (3 ECTS)

- Tutorials and personalized follow-up for practice groups. The teacher provides a **collection of problems**. Solutions are developed along the semester.
- Using compilers and editing tools.
- Continuous work driven by means of **two mandatory practices**.

Autonomous work:

- The practice will be completed with non-contact hours. In the Medium Group sessions the teacher supports mandatory practices which must be develop by the student throughout the course autonomously.
- It is recommended that students solve all problems from the collection problem, in order to practice and get feedback from the teacher.

Development plan

Week	Description	Large Group Activity	Medium Group Activity	Autonomous Work
1	Presentation Introduction	Introduction to the course. Introduction: processes, algorithms and programs	Using a programming environment.	Solve programming exercises.
2	Basic algorithmic structures	Unit 1: Constants, variables, basic types, valid expressions and standard input and output.	Programming exercises.	Solve programming exercises.
3	Basic algorithmic structures	Unit 1: Assignment, sequential composition and alternative composition.	Programming exercises.	Solve programming exercises.
4	Basic algorithmic structures	Unit 1: Iterative composition.	Programming exercises.	Solve programming exercises.
5	Design of iterative programs	Unit 2: Sequential access.	Practice 1: Overview of the first mandatory practice.	Implement Practice 1 in groups.
6	Design of iterative programs	Unit 2: Search in sequences.	Programming exercises. Support for Practice 1.	Solve programming exercises. Implement Practice 1 in groups.
7	Design of iterative programs	Unit 2: Direct access. Tables.	Programming exercises. Support for Practice 1.	Solve programming exercises. Implement Practice 1 in groups.
8	Design of iterative programs	Unit 2: Programming exercises with tables: treatment and search.	Programming exercises. Support for Practice 1.	Solve programming exercises. Implement Practice 1 in groups.
9		1st Assessment	Delivery of Practice 1.	Study. Implement Practice 1 in groups.
10	Design of iterative programs	Unit 2: Multidimensional tables.	Classic sorting algorithms.	Solve programming exercises.
11	Non-basic data processing	Unit 3: Procedures and Functions.	Programming exercises.	Solve programming exercises.
12	Non-basic data processing	Unit 3: Parameter passing mechanisms.	Practice 2: Overview of the second mandatory practice.	Implement Practice 2 in groups.
13	Non-basic data processing	Unit 3: Top-down design of algorithms.	Programming exercises. Support for Practice 2.	Solve programming exercises. Implement Practice 2 in groups.
14	Non-basic data processing	Unit 3: Programming exercises: top-down design of algorithms.	Programming exercises. Support for Practice 2.	Solve programming exercises. Implement Practice 2 in groups.
15	Non-basic data processing	Unit 3: Programming exercises: top-down design of algorithms.	Programming exercises. Support for Practice 2.	Solve programming exercises. Implement Practice 2 in groups.
16		2nd Assessment		Study. Implement Practice 2 in groups.
17		2nd Assessment	Delivery of Practice 2.	Study. Implement Practice 2 in groups.
18				
19		Improvement exam	Improvement of the 2nd practice.	Study. Implement Practice 2 in groups.

Evaluation

Evaluation activities

Acronym	Evaluation Activity	Weight	Minimum Grade	Team Work	Mandatory	Improvement
EP1	1st Assessment	25%	4	No	Yes	Yes
EP2	2nd Assessment	35%	4	No	Yes	Yes
PR1	Practice 1	15%	4	Yes (≤ 2)	Yes	Yes
PR2	Practice 2	25%	4	Yes (≤ 2)	Yes	Yes
To pass the subject it is necessary to obtain a minimum grade of 4 in all written tests and practices. In addition, the final grade must be ≥ 5.						
Final Grade = 0.25 · EP1 + 0.35 · EP2 + 0.15 · PR1 + 0.25 · PR2						

Remarks:

- If the grade obtained in the written test EP2 is ≥ 4, then this grade acts as improvement of the first written test EP1, the weight of which is 25%.
- If the grade obtained in the written test EP2 is < 4, then the student can choose to improve the 60% representing the written tests. The improvement assessment is a single written test and it is evaluated on 10 points. The new grade will represent 60% of the final grade. To pass the course this grade must be ≥ 4.
- If the second practice grade PR2 is ≥ 4, then this grade acts as improvement of the first practice PR1, the weight of which is 15%.
- If the second practice grade PR2 is < 4, the practice may be improved in the recovery period.

Bibliography

Algorithms

- T. Alsinet, J. Argelich and S. Vila. Programació I: Notes de curs. Eines, Edicions de la Universitat de Lleida, in press.
- J. Castro, F. Cucker, X. Messeguer, A. Rubio, L. Solano and B. Valles. Curs de Programació. McGraw-Hill, 1992.
- G. Brassard and P. Bratley. Fundamentos de Algoritmia. Prentice Hall, 1997.
- L. Joyanes. Fundamentos de Programación. Algoritmos, Estructuras de Datos y Objetos. McGraw-Hill, 2003.

ANSI C and C++

- H.M. Deitel and P.J. Deitel. Cómo Programar en C/C++. Prentice-Hall, segunda edición, 2002.
- B. Stroustrup. Programming -- Principles and Practice Using C++. Addison Wesley, 2008.
- L. Joyanes. Programación en C++. McGraw-Hill, 2006.