Universitat de Lleida

DEGREE CURRICULUM

INTRODUCTION TO PROGRAMMING I

Coordination: ARGELICH ROMA, JOSEP

Academic year 2020-21

INTRODUCTION TO PROGRAMMING I 2020-21

Subject's general information

<table>
<tr><td><b>Subject name</b></td><td>INTRODUCTION TO PROGRAMMING I</td></tr>
<tr><td><b>Code</b></td><td>102000</td></tr>
<tr><td><b>Semester</b></td><td>1st Q(SEMESTER) CONTINUED EVALUATION</td></tr>
<tr><td><b>Typology</b></td><td>
<table>
<tr><th>Degree</th><th>Course</th><th>Character</th><th>Modality</th></tr>
<tr><td>Double bachelor's degree: Degree in Computer Engineering and Degree in Business Administration and Management</td><td>1</td><td>COMMON</td><td>Attendance-based</td></tr>
<tr><td>Bachelor's Degree in Computer Engineering</td><td>1</td><td>COMMON</td><td>Attendance-based</td></tr>
</table>
</td></tr>
<tr><td><b>Course number of credits (ECTS)</b></td><td>6</td></tr>
<tr><td><b>Type of activity, credits, and groups</b></td><td>
<table>
<tr><th>Activity type</th><th>PRALAB</th><th>TEORIA</th></tr>
<tr><th>Number of credits</th><td>3</td><td>3</td></tr>
<tr><th>Number of groups</th><td>6</td><td>1</td></tr>
</table>
</td></tr>
<tr><td><b>Coordination</b></td><td>ARGELICH ROMA, JOSEP</td></tr>
<tr><td><b>Department</b></td><td>COMPUTER SCIENCE AND INDUSTRIAL ENGINEERING</td></tr>
<tr><td><b>Teaching load distribution between lectures and independent student work</b></td><td>GEI:
6 ECTS = 25x6 = 150 horas de trabajo
40% --> 60 horas presenciales
60% --> 90 horas de trabajo autónomo del estudiante</td></tr>
<tr><td><b>Important information on data processing</b></td><td>Consult this link for more information.</td></tr>
<tr><td><b>Language</b></td><td>Catalan</td></tr>
</table>

Teaching staff	E-mail addresses	Credits taught by teacher	Office and hour of attention
ALSINET BERNADÓ, MARIA TERESA	teresa.alsinet@udl.cat	3	
ARGELICH ROMA, JOSEP	josep.argelich@udl.cat	6	
GIBERT LLAURADÓ, DANIEL	daniel.gibert@udl.cat	3	
PLANES CID, JORDI	jordi.planes@udl.cat	9	

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.

This subject is scheduled in the fall semester of the 1st year.

The knowledge and competencies adquired in this subjects 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 to solve sequential treatment problems and then, to implement this algorithms with a programming language. Specifically, the programming language used for this pourpouse is ANSI C/C++ and 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

Strategic Competences of the UdL:

- EPS1. Capacity to solve problems and prepare and defence arguments inside the area of studies.

- EPS5. Capacity of abstraction and of critical, logical and mathematical thinking.
- EPS9. Capacity for unidisciplinary and multidisciplinary teamwork.
- EPS12. To be motivated for the quality and steady improvement.

Specific competences in the degree in Computer Engineering:

- GII-FB3. Capacity to understand and master the basic concepts of discreet mathematics, logical, algorithmic and computational complexity, and its application to solve engineering problems.
- GII-FB4. Basic knowledge of the use and programming of computers, operating systems, databases and computer programs with applications in engineering.
- GII-FB5. Knowledge of the structure, organisation, operation and interconnection of the computer systems, the basics of programming, and its application to solve engineering problems.
- GII-CRI7. Knowledge, design and efficient use of the types and data structure more suitable for solving a problem.
- GII-CRI9. Capacity to know, comprise and evaluate the structure and architecture of computers, as well as the basic components that conform them.

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. Iterative design of programs

2.1 Sequential Access

- Algorithmic schemes for sequence processing
- Algorithmic schemes for search 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 transfer mechanisms

3.3 Descendant 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 taught in the laboratory.

Large Groups: 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 Groups: Laboratory (3 ECTS)

- Tutorials and personalized follow-up for practice groups. The teacher provides a collection of problems. Solutions are developped along the semester.
- Using compilers and editing tools.
- Continous 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

Sem	Descripción	Actividad Presencial GG	Actividad Presencial GM	Trabajo autónomo
1	Presentation Introduction	Introduction to the course. Introduction: Processes, algorithms and programs	Using a programming environment	To solve programming exercises
2	Basic algorithmic structures	Unit 1. Constants, variables, basic types and valid expressions	Programming Exercises	To solve programming exercises
3	Basic algorithmic structures	Unit 1. Assignment, sequential composition and alternative composition	Programming Exercises	To solve programming exercises
4	Basic algorithmic structures	Unit 1. The iterative composition	Programming Exercises	To solve programming exercises
5	Iterative design of programs	Unit 2. Sequential Access	Practice 1: Overview of the first mandatory practice	To implement Practice 1 in groups
6	Iterative design of programs	Unit 2. Search in sequences	Programming Exercises Support for Practice 1	To solve programming exercises To implement Practice 1 in groups
7	Iterative design of programs	Unit 2. Direct access. Tables	Programming Exercises Support for Practice 1	To solve programming exercises To implement Practice 1 in groups

8	Iterative design of programs	Unit 2. Programming Exercises with tables: treatment and search.	Programming Exercises Support for Practice 1	To solve programming exercises To implement Practice 1 in groups
9		1st Assessment	Delivery Practice 1	To study To implement Practice 1 in groups
10	Iterative design of programs	Unit 2. Multidimensional tables	Classic sorting algorithms	To solve programming exercises
11	Non-basic data processing	Unit 3. Procedures and Functions	Programming Exercises	To solve programming exercises
12	Non-basic data processing	Unit 3. Parameter transfer mechanisms	Practice 2: Overview of the second mandatory practice	To implement Practice 2 in groups
13	Non-basic data processing	Unit 3. Descendant design of algorithms	Programming Exercises Support for Practice 2	To solve programming exercises To implement Practice 2 in groups
14	Non-basic data processing	Unit 3. Programming exercises: Descendant design of algorithms	Programming Exercises Support for Practice 2	To solve exercises To implement Practice 2 in groups
15	Non-basic data processing	Unit 3. Programming exercises: Descendant design of algorithms	Programming Exercises Support for Practice 2	To solve programming exercises To implement Practice 2 in groups
16		2nd Assessment		To study To implement Practice 2 in groups
17		2nd Assessment	Delivery Practice 2	To study To implement Practice 2 in groups
18				
19		Improvement of the 2nd assessment	Improvement of the 2nd practice	To study To implement Practice 2 in groups

Evaluation

Evaluation activities

Acronym	Evaluation activities	Weighing	Minimum grade required	Team work	Mandatory	Improvement
P1	*Assessment 1*	25%	4	NO	YES	YES
P2	*Assessment 2*	35,00%	4	NO	YES	YES
PRA1	*Practice 1*	15,00%	4	Maximum 2 people	YES	YES
PRA2	*Practice 2*	25,00%	4	Maximum 2 people	YES	YES
To pass the subject is necessary to obtain the minimum grade of 4 in all written tests and practices. In addition, the final grade must be > = 5.						
Final grade = 0,25*P1 + 0,35*P2 + 0,15*PRA1 + 0,25*PRA2						

Remarks:

- If the grade obtained in the written test P2 is > = 4, then this grade acts as improvement of the first written test P1, the weight of which is 25%.
- If the grade obtained in the written test P2 is < 4, then the student can choose to improve 60% representing the written tests. The improvement assessment is a single written test and 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 PRA2 is > = 4, then this grade acts as improvement/recovery of the first practice PRA1, the weight of which is 15%.
- If the second practice grade PRA2 is < 4, the practice may be recovered in the recovery period.

Bibliography

Basic References:

- J. Castro, F. Cucker, X. Messeguer, A. Rubio, L. Solano and B.Valles. *Curs de Programació*. McGraw-Hill, 1992.
- G. Brassard and P. Bratley. *Fundamentosde 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. *ComoProgramar 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.