DEGREE CURRICULUM

INTRODUCTION TO PROGRAMMING I

Coordination: MARTINEZ RODRIGUEZ, SANTIAGO

Academic year 2023-24

INTRODUCTION TO PROGRAMMING I 2023-24

Subject's general information

<table>
<tr><td>Subject name</td><td>INTRODUCTION TO PROGRAMMING I</td></tr>
<tr><td>Code</td><td>105000</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 Computer Engineering</td><td>1</td><td>COMMON/CORE</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>PRALAB</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>MARTINEZ RODRIGUEZ, SANTIAGO</td></tr>
<tr><td>Department</td><td>COMPUTER ENGINEERING AND DIGITAL DESIGN</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>
<tr><td>Distribution of credits</td><td>Theory: 3
Practices: 3</td></tr>
</table>

INTRODUCTION TO PROGRAMMING I 2023-24

Teaching staff	E-mail addresses	Credits taught by teacher	Office and hour of attention
MARTINEZ RODRIGUEZ, SANTIAGO	santi.martinez@udl.cat	3	Arrange with the teacher. Optionally, by videoconference.
TOMAS GLEYAL, MARC	marc.tomas@udl.cat	6	Arrange with the teacher. Optionally, by videoconference.

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 student's learning results are knowing how to apply the techniques of analysis and design of algorithms to implement them in a high-level programming language.
Specifically, the chosen imperative language is ANSI C/C++ and the problems to be solved are mainly those related with sequences processing.

In particular, the student's learning results are as follows:

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

Competences

Cross-Disciplinary Competences

- **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 / Module of basic training

- **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.

Specific Competences / Module of common training in the computer branch

- **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.

INTRODUCTION TO PROGRAMMING I 2023-24

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 credits)

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

Medium Group: Practices (3 credits)

- 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 practices.

Autonomous work:

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

Week	Description	Large Group Activity	Medium Group Activity	Autonomous Work
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 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 Midterm Exam	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 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 Midterm Exam	Delivery of Practice 2.	Study. Implement Practice 2 in groups.
17		2nd Midterm Exam		Study.
18				Study.
19		Improvement Exam		Study.

Evaluation

The **continuous evaluation** of the subject is based on 3 blocks:

- **Practice Block** (25%): It consists of two activities: Practice 1 and Practice 2. They cannot be improved. A minimum grade is not required.
- **Theory block 1** (25%): It consists of one activity: 1st Midterm Exam. It can be improved with the theory block 2. A minimum grade is not required. Date of the exam: the date of the realization of the 1st Midterm Exam, defined by the EPS.
- **Theory block 2** (50%): It consists of one activity: 2nd Midterm Exam. It can be improved. A minimum grade is not required. Date

of the exam: the date of the realization of the 2nd Midterm Exam, defined by the EPS.

Improvement of Theory Block 2: It consists of performing a 2nd Midterm Exam again. A minimum grade is not required. Date of the exam: the date of the realization of the Improvement Exam, defined by the EPS. The realization of Improvement of Theory Block 2 does not condition the maximum grade achieved in the subject.

Evaluation activities

Acronym	Evaluation Activity	Weight	Minimum Score	Group	Compulsory	Recoverable
EP1	1st Midterm Exam	25%	No	No	No	Yes
EP2	2nd Midterm Exam	50%	No	No	No	Yes
PR1	Practice 1	10%	No	Yes (≤ 2)	No	No
PR2	Practice 2	15%	No	Yes (≤ 2)	No	No
To pass the subject the final score must be ≥ 5.						
Final Score = 0.25 · EP1 + 0.5 · EP2 + 0.1 · PR1 + 0.15 · PR2						

Remarks:

- If the grade obtained in the midterm exam EP2 is > = 5, then this grade may act as an improvement of the first midterm exam EP1.
- The student can choose to improve the midterm exam EP2. The improvement exam is a single written exam. The mark obtained will replace the mark of the two midterm exams of the course.

Alternative evaluation (students who waive continuous evaluation):

Students who have the approval to be evaluated by alternative evaluation (see requirements and procedure in the evaluation regulations) will have to do the following activities.

- **Practice 1** (10%): It cannot be improved. A minimum grade is not required. Delivery date: the date of the realization of the 1st Midterm Exam, defined by the EPS.
- **Practice 2** (15%): It cannot be improved. A minimum grade is not required. Delivery date: the date of the realization of the 2nd Midterm Exam, defined by the EPS.
- **Midterm Exam 2** (75%): It can be improved. A minimum grade is not required. Date of the exam: the date of the realization of the 2nd Midterm Exam, defined by the EPS.
- **Improvement of Midterm Exam 2** (75%): A minimum grade is not required. Date of the exam: the date of the realization of the Improvement Exam, defined by the EPS. The realization of Improvement of Midterm Exam 2 does not condition the maximum grade achieved in the subject.

Bibliography

Algorithms

- Teresa Alsinet, Josep Argelich, Sergi Vila: Programació I. Notes del curs. Eines; Edicions i Publicacions de la Universitat de Lleida.
- Jorge Castro, Felipe Cucker, Xavier Messeguer, Albert Rubio, Lluis Solano, Borja Valles: Curs de Programació. McGraw-Hill, 1992.
- Gilles Brassard, Paul Bratley: Fundamentos de Algoritmia. Prentice Hall, 1997.
- Luis Joyanes: Fundamentos de Programación. Algoritmos, Estructuras de Datos y Objetos. McGraw-Hill, 2003.

ANSI C and C++

- Harvey M. Deitel, Paul J. Deitel: Cómo Programar en C/C++. Prentice-Hall, segunda edición, 2002.
- Bjarne Stroustrup: Programming: Principles and Practice Using C++. Addison Wesley, 2008.
- Luis Joyanes: Programación en C++. McGraw-Hill, 2006.