Course guide
340368 - FOPR-I1O23 - Fundamentals of Programming

Last modified: 11/03/2024

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: 2023 **ECTS Credits:** 7.5 **Languages:** Catalan

LECTURER

Coordinating lecturer:	BERNARDINO CASAS FERNÁNDEZ
Others:	BERNARDINO CASAS FERNÁNDEZ

PRIOR SKILLS

Basic knowledge of mathematics for the required level at the university entrance exam.

DEGREE COMPETENCES TO WHICH THE SUBJECT CONTRIBUTES

Specific:
3. CEFB3. Ability to understand and to have a good command of discrete, logical, algorithmically mathematics and computing complexity and its application to automatical treatment of information by means of computational systems and its application to solve engineering problems.
4. CEFB4. Basic knowledge of use and computer programming, as well as of operating systems, data base and generally informatic programs with engineering applications.
5. CEFC6. Basic knowledge and application of algorithmic processes, informatic techniques to design solutions of problems, analyzing if proposed algorisms are apt and complex.
6. CEFC7. Knowledge, design and efficient use of data types and structures the most appropriate to resolve problems.

Transversal:
1. SELF-DIRECTED LEARNING. Detecting gaps in one's knowledge and overcoming them through critical self-appraisal. Choosing the best path for broadening one's knowledge.
2. TEAMWORK. Being able to work as a team player, either as a member or as a leader. Contributing to projects pragmatically and responsibly, by reaching commitments in accordance to the resources that are available.

TEACHING METHODOLOGY

The course consists of:
- 2 hours per week of theory class in the classroom (large group) where the teacher presents contents,
- 3 hours per week in classroom lab (small group) where it's done practical work and evaluable activities are proposed and performed.

LEARNING OBJECTIVES OF THE SUBJECT

1. To understand the programme building process and how to use the tools required: console, editor and compiler.
2. To know the syntax and the semantics expressions and the basic instructions of the imperative programming languages (C ++).
3. Being proficient in using functions and actions in programming.
4. To understand the function and parameter concepts.
5. To know in depth the tables and to identify where their use is appropriate.
6. To be able to contrast solutions regarding the use of time and memory resources and to choose the most appropriate in simple cases.
7. To understand the patterns of treat-all and search algorithms patterns.
8. Choosing an appropriate scheme solution.
9. To understand the recursion concept. To be able to propose recursive solutions to simple problems.
10. To comprehend in depth the binary search, insertion sort, selection sort, mergesort and quicksort.
11. To know in depth other fundamental algorithms: Hörner, fast product, etc.
12. To be able to write programs readable, elegant and efficient.

STUDY LOAD

Type	Hours	Percentage
Self study	112,5	60.00
Hours large group	30,0	16.00
Hours small group	45,0	24.00

Total learning time: 187.5 h

CONTENTS

Introduction

Description:
Programming examples
Algorithms, programming languages and computer programs
Steps in the design of a program

Full-or-part-time: 6h
Theory classes: 2h
Laboratory classes: 3h
Self study : 1h

Variables and statements

Description:
Variables, data types and expressions
Statements:
- Assignment
- Input / Output
- Conditional statement

Full-or-part-time: 6h
Theory classes: 2h
Laboratory classes: 3h
Self study : 1h

Loops

Description:
While statement
For statement

Full-or-part-time: 6h
Theory classes: 2h
Laboratory classes: 3h
Self study : 1h

Data types and visibility

Description:
Data types
Type conversion
Visibility

Full-or-part-time: 6h
Theory classes: 2h
Laboratory classes: 3h
Self study : 1h

Subprograms: procedures and functions

Description:
Subprogram concept
Parameter passing
Functions
Procedures

Full-or-part-time: 7h
Theory classes: 2h
Laboratory classes: 3h
Self study : 2h

Algorithms on sequences. Invariants.

Description:
Algorithms on sequences:
- Treat-all algorithms
- Search algorithms
Reasoning about loops: invariants

Full-or-part-time: 7h
Theory classes: 2h
Laboratory classes: 3h
Self study : 2h

Recursion

Description:
Recursive design
Exemples

Full-or-part-time: 6h
Theory classes: 2h
Laboratory classes: 3h
Self study : 1h

Vectors

Description:
Vectors
Searching in vectors

Full-or-part-time: 6h
Theory classes: 2h
Laboratory classes: 3h
Self study : 1h

Vectors and strings

Description:
More vectors examples
Strings

Full-or-part-time: 6h
Theory classes: 2h
Practical classes: 3h
Self study : 1h

Multi-dimensional vectors

Description:
Matrices
N-dimensional vectors
Search in a matrix
Search in a sorted matrix
Matrix multiplication

Full-or-part-time: 8h
Theory classes: 2h
Laboratory classes: 4h
Self study : 2h

Structures and data structure design

Description:
Structures
Data structure design

Full-or-part-time: 6h
Theory classes: 2h
Laboratory classes: 3h
Self study : 1h

Sorting

Description:
Selection Sort
Insertion Sort
Bubble Sort
Merge Sort

Full-or-part-time: 8h
Theory classes: 2h
Laboratory classes: 4h
Self study : 2h

Numerical algorithms

Description:
Product of polynomials
Sum of polynomials
Sum of sparse vectors
Root of a continuous function

Full-or-part-time: 9h
Theory classes: 2h
Laboratory classes: 4h
Self study : 3h

Advanced examples

Description:
Sports tournament
Permutations
Sub-sequences summing n

Full-or-part-time: 7h
Theory classes: 2h
Laboratory classes: 3h
Self study : 2h

Conclusions

Description:
Why is programming hard?
Useful programs
Correct programs
Efficient programs
Programs are mathematical objects
Easy to understand, modify and extend
Programming has limits
Quotes

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

GRADING SYSTEM

QU = Grade from questionnaires, all with the same weight.
AC = Grade from activities, all with the same weight.
PR = Grade from programming task.
C1 = Grade from Exam 1.
C2 = Grade from Exam 2.
PV = Validation Test.
Nota Final = max (50%C2, 20%C1+30%C2) + 10%QU + 20%AC + 20%(PR*PV)
The reevaluation contains the C2 test.

EXAMINATION RULES.

The activities (AC), the controls (C1 and C2) and the validation test (PV) are face and individual.
The questionnaires (QU) are self-assessed and their delivery is electronic and individual.
The programming task (PR) is done in groups. Teachers could ask about the work presented by the students and consider their answers when qualifying. The validation test shall be carried out in conjunction with Control 2.