Universitat Autònoma de Barcelona

2019/2020

Programming Lab

Code: 102767
ECTS Credits: 6

Degree	Type	Year	Semester
2502441 Computer Engineering	OB	2	1

Contact

Name: Gemma Sánchez Albaladejo

Email: Gemma.Sanchez@uab.cat

Use of Languages

Principal working language: catalan (cat)

Some groups entirely in English: No

Some groups entirely in Catalan: Yes

Some groups entirely in Spanish: No

Teachers

Ernest Valveny Llobet

Aura Hernández Sabaté

Oriol Ramos Terrades

Prerequisites

The subject cannot have any official prerequisite by regulation. But students who have not completed and previously passed the subjects of Fonaments d'informàtica and Metodologia de la Programació have a high percentage of suspended. Therefore, it is strongly recommended that the student has satisfactorily completed and passed the
previous subjects of Fonaments d'informàtica, Metodologia de la Programació as well as Matemàtica Discreta Therefore, you are familiar with the basic and advanced structures of programming, Orientation objects and the concept of graph with the different methods of travel on them.

Objectives and Contextualisation

This subject is part of the Algorithmic and Information subject and should be seen as the logical continuation of the subject Metodologia de la Programació and the practical continuation of the Matemàtica Discreta. The basic objective is to deepen the notions of object-oriented programming introduced in the Programming Methodology and expand with other programming concepts and other more complex data structures, as well as efficient algorithms for traversing. The concept of a recursive algorithm with simple and more complex recursive algorithms such as those related to tree and graph paths will be introduced. In addition, search algorithms and efficient management will be introduced and deepen the concept of time and space cost of an algorithm. At the end of the course the student must be able to design and program solutions to complex problems in an optimal way.

In this way, the training objectives that are proposed for the subject are the following:

- Be able to analyze a complex problem, design an optimal solution, implement it, calculate its cost and test it.
- Understand and know how to use complex data structures such as trees, graphs etc. and use them correctly and in an efficient way to solve complex algorithmic problems.

- Understand and correctly apply the advanced principles of object-oriented programming: templates , abstract classes, virtual functions.
- To provide the student with the ability to design algorithms for the resolution of complex problems, seeing complex algorithms of travel and search in complex data structures. In addition to analyzing the temporal and spatial complexity of them in order to choose the solution that best suits the needs of each moment.
- Introduce the concept of recursion and its application to the path of complex recursive structures, as well as being able to analyze the complexity of recursive algorithms.
- Program in a real programming language and be able to debug your own programs.
- Develop the programs following a style norms tending to achieve quality programs. Within these style rules include those that facilitate the understanding of the code, such as the use of comments, the tabulation of the code, the use of appropriate names for variables and functions, etc. And the use of exceptions.

Competences

- Acquire thinking habits.
- Have the capacity to conceive, develop and maintain computer systems, services and applications employing the methods of software engineering as an instrument to ensure quality.
- Know and apply the basic algorithmic procedures of computer technologies to design solutions for problems and to analyse the adequacy and complexity of the algorithms proposed.
- Knowledge, design and efficient use of the most suitable data types and structures for problems solving.

Learning Outcomes

1. Apply program debugging, testing and correction strategies.
2. Develop a capacity for analysis, synthesis and prospection.
3. Develop programs using good programming style and documenting them properly.
4. Identify possible troubleshooting strategies by means of ones own concepts of the object-oriented programming paradigm.
5. Identify the computational complexity of an algorithm in terms of memory resources and run time.
6. Know and understand programming paradigms.
7. Select and apply the most appropriate combination of data structures and strategies to resolve a computing problem.

Content

0. Introduction
Objectives and presentation of the subject. Review of Programming Oriented to Objects and Dynamic Data Structures.

1. Object orientation
Advanced object paradigm. Templates Heritage, lessons abstract, virtual functions and polymorphism.

2. Non-linear data structures. hash

"Hashing" techniques. Hash Matrices and Hash lists. Hash Functions.

3. Recursion and Sorting algorithms

Introduction to recursive algorithms. Bubble method, QuickSort, mergesort. Recursivity Calculation complexity.

4. Non-linear data structures. Graphs
Representations and tours. BFS, DFS, Resolution of problems with graphs.

5. Non-linear data structures. Trees

Definition and representation of a tree. Paths of trees. Binary Heaps. Red-Black Tree.

6. Basic concepts of Python

Basic programming concepts in Python

Methodology

The teaching methodology of the subject starts from the principle that says *"program it's the only way to learn to program"* and, therefore, will be mainly focused on the practical work of the student. It is also based on making the most of the face time the student is with the teacher. In this way, descriptive theoretical concepts, easily reached by the student by viewing videos or reading articles, will be done autonomously (and guided) by the student. While the implementation of these concepts or the expansion of them will be done in class with the help of the teacher. The main objective of the subject is that the student knows how to solve a given problem, efficiently, using complex data structures, if necessary. For this reason, the learning will focus on accompanying the student in his task of solving problems from theoretical concepts previously studied autonomously. The C ++ programming language will be used mainly, and some programming concepts will be given in python.

The general methodology of the subject can be divided into three phases:

Preparation of the class: The objective of this phase is that the students can learn the concepts that will be worked on in the following session through various activities offered by the teaching staff, such as viewing videos, reading texts, etc.

Class face-to-face: The objective of this phase is to consolidate the concepts seen and put them into value within the context of the subject. The faculty will ensure that the students deepen these concepts through exercises (more or less) guided during the session, and adding new nuances to concepts learned autonomously when necessary. In this way the face-to-face class is used much more since the descriptive concepts are already known by the student and one can fully enter into its use and expansion, which are the points in which the student may need the help of the teacher more.

Work autonomous: so that the students take ease in the use of complex structures and the algorithms associated with them. They will have to do a part of the work on their own, whether solving new exercises or within a project.

The work of the project and of the laboratory sessions should be done in groups of 2 people.

The teaching management of the subject will be done through Charon (http://caronte.uab.cat/), which will serve to see the materials, manage the groups of practices, make the corresponding deliveries, see the notes, communicate with the teachers, etc.

Transversal competences

- T01.02 Develop the capacity for analysis, synthesis and prospective. This competence will be developed during the face-to-face sessions, where the student, after having consolidated some theoretical concepts, will analyze a practical problem and propose an optimal solution. It will be taken into account in the evaluation of the different parts of the subject.

Activities

Title	Hours	ECTS	Learning Outcomes
Type: Directed			
Face-to-face classes	50	2	1, 6, 2, 3, 5, 4, 7

Type: Supervised			
Consultations	1	0.04	1, 6, 2, 3, 5, 4, 7
Type: Autonomous			
Autonomous work	46	1.84	1, 2, 3, 5, 4, 7
Individual Studio	13	0.52	6, 2, 5, 4
Previous preparation of the classes	34	1.36	6, 2, 5, 4

Assessment

The evaluation of the subject will take into account three types of evaluation activities: delivery of problems, individual evaluation and programming project. The Final note of the subject is obtained by combining the evaluation of these 3 activities as follows:

Final Note = (0.2 * Evaluation Problems) + (0.4 * Project) + (0.4 * Individual Evaluation)

- Delivery problems: This section includes the delivery of the exercises that are proposed throughout the course and other activities that take place in the problem sessions.

You should get a minimum grade of 5 in this activity to be able to pass the subject.
Exercises that are delivered late or that have an evaluation of suspended can be recovered and redeliver at any time during the course before the final exam date of the subject, with a reduction on the grade of the twenty%. The problems will be weighted according to the weight of the subject to the whole of the subject, and the number of problems that they have to deliver for each subject.

- Evaluation individual: This section includes the results of the individual tests that will be done throughout the course. There will be two partial tests that will be done during the class period of the course during class time and a final test during the official exam period. This final test will be recovery and will only do students who have not passed any of the two sets. If one of the two partials has been passed, but the other does not, in this test only the part of the subject corresponding to the part that has not been passed must be recovered.

You should get a minimum grade of 4 in each of the two partials and a note average minimum of 5 to be able to pass the subject.

The final grade will be the average of the two partials: Individual Evaluation = (0.5 * Partial1) + (0.5 * Partial2)

- Draft: includes all the work of the programming project. It includes the evaluation of the two deliveries of the project (a partial delivery in the middle of the course and the final delivery) and the evaluation of the follow-up of the project that will be made to the face-to-face sessions correspond The final grade will be calculated as follows:

Project = (0.2 * Project monitoring evaluation) + (0.3 * Partial Delivery 1) + (0.5 * Final Delivery)

- You should get a minimum grade of 4 in the evaluation of tracing of the project and a minimum note of 5 in the final delivery of the project to be able to approve the project.
- You should get a minimum grade of 5 at draft to be able to pass the subject.
- The note of the final delivery of the project can berecovered if the project note is> = 3 and the individual evaluation note is> = 5.

Do not evaluable: A student will be considered not evaluable (NA) if he does not make at least 50% of the deliveries of exercises and does not do any of the evaluation tests: partial 1, partial 2, final test of recovery, final delivery of the practice.

suspended: If the calculation of the final grade is equal to or greater than 5 but does not reach the minimum required in any of the evaluation activities, the grade final it will be suspended and a 4.5 will be placed on the note in the student's file .

validations: For the repeating students, the previous year's project grade (course 201 8 -1 9) will be validated if these conditions are met:

to) The final grade of the project of the previous course is greater than or equal to 7

b) The grade of the individual evaluation of the previous course is greater than or equal to 3

MH: There will be as many enrollments as possible within the regulations of the university, starting with the highest grades and as long as the minimum grade is a 9.

Reviews: For Each evaluation activity will indicate a place, date and time of revision in which the student can review the activity with the teacher. In this context, claims may be made on the activity grade, which will be evaluated by the faculty responsible for the subject. If the student does not appear in this review, this activity will not be reviewed later.

Important note about copies and plagiarism:
Without prejudice to other disciplinary measures deemed appropriate, and in accordance with current academic regulations, the Irregularities committed by a student that may lead to a variation of the grade will be scored with a zero (0). the evaluation activities qualified in this way and by this procedure will not be recoverable. If it is necessary to overcome any of these evaluation activities p ara appro b ar the subject, this subject will be suspended directly, without the opportunity of recover it in the same course. These irregularities include, otr to s:

- the total or partial copy of a practice, report, or any other evaluation activity;
- let copy;
- present a group work not done entirely by the members of the group;
- present as own materials prepared by a third party, although they are translations or adaptations, and in general works with non original and exclusive elements of the student;
- have communication devices (such as mobile phones, smart watches , etc.) accessible during the evaluation tests theoretical Individual practices (exams).

In these cases, the numerical note of the file will be the lower value between 3.0 and the weighted average of the notes (and therefore the approved by compensation will not be possible).
In the evaluation of the delivery of problems and practices, copy detection tools of the program code will be used.

Note on the planning of evaluation activities:
The dates of continuous evaluation and delivery of works will be published at the beginning of the course and may be subject to changes in programming for reasons of adaptation to possible incidents. Always be informed Cerberus about these changes since it is understood that this is the usual platform for exchange of information between teachers and students.

Assessment Activities

Title	Weighting	Hours	ECTS	Learning Outcomes
Delivery problems	20% Final grade	0	0	6, 2, 5, 4, 7
First partial	50% Individual evaluation note.	2	0.08	1, 6, 2, 3, 5, 4, 7
Programming Project	40% Final grade	0	0	1, 2, 3, 4, 7
Recovery test	see the description of the evaluation method	2	0.08	1, 6, 2, 3, 5, 4, 7

Second partial	50% Individual evaluation note.	2	0.08	1, 6, 2, 3, 5, 4, 7

Bibliography

- http://www.cplusplus.com/ : The C++ Resources Network
- https://es.wikibooks.org/wiki/Programaci%C3%B3n_en_C%2B%2B: Programación en C++ - Wikilibros
- Mark Allen Weiss. Data Structures and Data Analysis in C++. Pearson. 2014.
- B. Eckel. Thinking in C++, Volume 1: Introduction to Standard C++, Prentice-Hall, 1999
- B. Eckel. Thinking in C++, Volume 2: Standard Libraries and Advanced Topics, Prentice-Hall, 1999
- F. Xhafa, P. Vázquez, J. Marco, X. Molinero, A. Martín: Programación en C++ para ingenieros. Thomson, 2006
- Thinking in PYTHON Bruce Eckel (se puede descargar de http://www.bruceeckel.com).
- Learning PYTHON 2nd Edition. Mark Lutz and David Ascher, Safari Tech Books Online.
- Manuals de Python (de la pagina web oficial).
- Llibres electronics interactius de python:
    - http://interactivepython.org/runestone/static/thinkcspy/toc.html#t-o-c
    - http://interactivepython.org/runestone/static/pythonds/index.html
      http://www.pythontutor.com/