Universitat Autònoma de Barcelona

**2019/2020**

# Programming Methodology

Code: 102764
ECTS Credits: 6

| Degree | Type | Year | Semester |
|---|---|---|---|
| 2502441 Computer Engineering | FB | 1 | 2 |

## Contact

Name: Ernest Valveny Llobet

Email: Ernest.Valveny@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 |

## Prerequisites

There are no prerequisites. However, students should be familiar with the most basic programming structures. Thus, it is recommended for students to have taken the course "Fonaments d'Informàtica".

## Objectives and Contextualisation

This subject is part of the Computer Science subject and it should be regarded as the logical continuation of the programming part of the Fundamentals of Computer Science subject. The basic objective is to deepen in the basic notions of programming introduced in Fundamentals of Computer Science and to complete them with other programming concepts that should allow the student to have complete vision of the imperative programming and the principles of object-oriented programming.

In this way, the objectives for this course are the following:

- Understanding the life cycle of the software: analysis (understand the problem), design (propose a solution to the problem), implementation (coding the solution in a programming language), test (carrying out a systematic test to ensure the correctness of the implemented solution).
- Understanding the concept of algorithm as a tool for solving problems with the computer, learn the fundamental concept of algorithm, understand the main structures of imperative programming and use them correctly to solve algorithmic problems of a certain complexity.
- Knowing the different structures to represent information within the algorithms, both static data types (tables, records and strings) and dynamic data types (stacks, tails, lists), to be able to use the most appropriate data structure to represent the associated information an algorithmic problem.
- Understanding and correctly applying the basic principles of object-oriented programming: class concept and data encapsulation.
- To provide the student with the ability to design algorithms for the resolution of complex problems, introducing progressively and systematically a rigorous and structured programming methodology, based fundamentally on the technique of descending design.
- Programming in a real programming language and knowing the different stages of development of a program: writing, compilation and linking, execution and testing.
- Developing the programs following a coding style tending to achieve quality programs. Such a coding style include rules that facilitate the understanding of the code, such as the use of comments, the indentation of the code, the use of appropriate names for variables and functions, etc.

## Competences

- Acquire personal work habits.
- Acquire thinking habits.
- Have basic knowledge of the use and programming of computers, operating systems, databases and computer programs with applications to engineering.
- Have the capacity to understand and master the basic concepts of discreet mathematics, logic, computational algorithms and complexity, and their application to the resolution of engineering problems.
- Know the basic materials and technologies to enable the learning and development of new methods and technologies, as well as those that that provide large-scale versatility to adapt to new situations.

## Learning Outcomes

1. Develop a mode of thought and critical reasoning.
2. Know and be able to use operating systems, databases and programs that are commonly used in engineering.
3. Know the basic principles of the structure and programming of computers.
4. Make ones own decisions.
5. Manage time and resources available. Work in an organized manner .
6. Recognise and identify the methods, systems and technologies of computer engineering.
7. Show the capacity for the organisation of information in files and databases.
8. Show the capacity to design algorithms and analyse their computational complexity.
9. Show the capacity to display information and program computers.
10. Understand and master algorithms and computational complexity and their application to problem solving.

## Content

Topic 1: Introduction to object-oriented programming

• Review of the basic programming structures: conditional and iterative structures, functions and procedures, tables, matrices, registers.

• Introduction to the concept of class. Methods and attributes. Private and public part. Constructors and destructors. Data composition. Composition.

• Persistence and serialization of objects. Reading and writing files.

Topic 2: Dynamic data structures

• Motivation of dynamic data structures.

• The concept of pointer. Operations with pointers.

• Dynamic objects.

• Definition, representation and implementation of dynamic data structures: lists, stacks and queues.

• Use of dynamic data structures.

Topic 3: Cost analysis and complexity

## Methodology

The teaching methodology of the course is based on the principle that "programming is the only way to learn to program" and, therefore, will be mainly focused on the practical work of the student. Lectures will introduce the theoretical contents of the subject, from a very practical perspective using examples and exercises and programming problems that must be solved in class directly with the computer. On the other hand, a programming project will have to be developed mainly autonomously throughout the course (with follow-up and control by the teacher in specific sessions). It will involve applying almost all concepts and programming tools introduced in lectures to solving a complex real problem. In addition, a set of exercises will have to be solved individually throughout the course (some of them will be solved and discussed in lectures). They should serve to understand, integrate and apply the concepts developed in lectures. In all the activities of the course (lectures, problems and project) the C++ programming language will be used.

In lectures, the course will not distinguish between theory, problem and practical lectures. Lectures will be organized in four hours per week in groups of around 40 students. Each registration group is divided into two groups of about 40 students for lectures. The division of students between the two groups will be done at the beginning of the semester and will be fixed for the whole course. The sessions will be held in a classroom with computers to facilitate the practical work of the student. It is recommended that the student bring his own laptop to class if he has one.

In the lectures, the concepts of the syllabus of the subject will be introduced. In some cases, explanatory videos may be made available to the student to watch before the class session. Lectures will have a very practical approach with examples and exercises that will be presented to students to facilitate understanding and learning of the concepts. These exercises will be carried out anddiscussed during the session and will be used to introduce the contents of the subject and see its practical application.

The student must complete the lectures with autonomous personal work to do the exercises that are proposed and that should serve to understand the contents of the course. It must be borne in mind that the syllabus of the subject has a logical continuity throughout the course, so that to follow correctly a class it is necessary to understand what was explained in the previous sessions. Some of these exercises must be submitted individually as part of the evaluation of the course.

In addition, students must develop in groups of 2 people a programming project that will be developed autonomously throughout the course apart from the lectures. The programming project will allow addressing a programming problem of a certain complexity that integrates most of the concepts explained during the course. During the course, some lectures sessions will be devoted to the control, monitoring and evaluation of the work carried out by the student in the programming project.

The management of the course will be done through the platform Caronte (http://caronte.uab.cat/), which will be used to view the materials, manage the groups of the project, submitting exercises, publish evaluation marks, communicate with teachers, etc.

Transversal Competences

• T01.01 Develop critical thinking and reasoning. This competence will be developed during the lectures, from the presentation and discussion of examples and practical cases. It will be taken into account in the evaluation of students problem submission.

• T02.03 Manage time and available resources. Work in an organized way. This competence will be developed mainly thorugh the programming project. The student aims to develop the project autonomously and must be able to organize the time and resources to achieve this goal. It will be evaluated in the follow-up sessions of the project.

• T02.05 Make their own decisions. This competence will be developed during the programming project in which the student must choose and take the best options to complete the project. It will be evaluated in the follow-up sessions of the project.

## Activities

| Title | Hours | ECTS | Learning Outcomes |
|---|---|---|---|
| Type: Directed | | | |
| Lectures | 50 | 2 | 10, 3, 7, 9, 8, 1, 6 |

| | | | |
|---|---|---|---|
| Type: Supervised | | | |
| Follow-up of the implementation of the programming project | 1 | 0.04 | 8, 1, 5, 4 |
| Type: Autonomous | | | |
| Implementing a programming project | 48 | 1.92 | 2, 7, 9, 8, 5, 4 |
| Individual study | 11 | 0.44 | 3, 6 |
| Solving problems | 36 | 1.44 | 7, 9, 8, 5 |

## Assessment

Subject assesement will take into account three types of assesement activities: resolution of problems, individual assesement and programming project. The final grade of the subject is obtained by combining the grade of these 3 activities as follows:

Final Grade = (0.3 * Continuous assesement) + (0.3 * Project) + (0.4 * Individual assesement)

- Continuous assesement: this section includes the assesement of the exercises that are proposed throughout the course and other activities that are carried out in the lectures or are proposed to be delivered on the Caronte platform.
- A minimum grade of 5 is required in this activity in order to pass the subject.
- The exercises that are not delivered within the deadline or that are suspended can be re-delivered at any time during the course before the final exam date of the subject, with a reduction of the 20% in the grade.

- Individual assesement: 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 period of lectures in the course during class time and a final test during the official exam period. This final test will permit to compensate any of the partial tests and will only have to be done by students who have not passed any of the two partial test. If one of the two partial test has been passed, but the other does not, in this test only the part of the subject corresponding to the partial test that has not been passed must be re-assesed.

- A minimum grade of 4.5 is required in each of the two partial tests in order to pass the subject.

- The final grade will be the average of the two partial tests:

  Individual assesement = (0.5 * Partial Test 1) + (0.5 * Partial Test 2)

- A minimum grade of 5 is required in the final grade of the individual assesement in order to pass the subject.

- Project: includes the assesement of all the work carried out in the programming project: the two deliveries of the project (a partial delivery in the middle of the course and the final delivery) and the conitnuous monitoring of the project that will be done throughout the course.
- A minimum grade of 5 is required in the final delivery of the project in order to pass the subject.
- A minimum grade of 5 is required in the overall grade of the project in order to pass the subject.
- The final delivery of the project can be re-assesed if the project grade is> = 3 and the individual assesement grade is> = 5.

Not assesed: A student will be considered not assesed (NA) if he does not submit at least 50% of the deliveries of exercises and does not do any of the evaluation tests: partial test 1, partial test 2, final test, final delivery of the project.

Suspended: If the final grade is equal to or greater than 5 but the student does not reach the minimum required in any of the assesement activities, the final grade will be suspended and , the grade in the Transcript of Records (ToR) will be 4.5.

Pass the course with honours: In order to pass the course with honours, the final grade must be a 9.0 or higher. Because the number of students with this distinction cannot exceed 5% of the number of students enrolled in the course, this distinction will be awarded to whoever has the highest final grade.

Compensations: For the students who are following the course for the second time or more, the project grade of the previous year can be compensated if these conditions are met:

- The final grade of the project of the previous year is greater than or equal to 7
- The grade of the individual assesement of the previous year is greater than or equal to 3.

Review of assesement: For each assesement activity, a place, date and time of review will be indicated allowing students to review the activity with the lecturer. In this context, students may discuss the activity grade awarded by the lecturers responsible for the subject. If students do not take part in this review, no further opportunity will be made available.

Important note: copies and plagiarism

Notwithstanding other disciplinary measures deemed appropriate, and in accordance with the academic regulations in force, assessment activities will receive a zero whenever a student commits academic irregularities that may alter such assessment. Assessment activities graded in this way and by this procedure will not be re-assessable. If passing the assessment activity or activities in question is required to pass the subject, the awarding of a zero for disciplinary measures will also entail a direct fail for the subject, with no opportunity to re-assess this in the same academic year. Irregularities contemplated in this procedure include, among others:

- the total or partial copying of a practical exercise, report, or any other evaluation activity;
- allowing others to copy;
- presenting group work that has not been done entirely by the members of the group;
- presenting any materials prepared by a third party as one's own work, even if these materials are translations or adaptations, including work that is not original or exclusively that of the student;
- having communication devices (such as mobile phones, smart watches, etc.) accessible during theoretical-practical assessment tests (individual exams).
- talk with peers during the theoretical evaluation tests-individual practices (exams);
- copy or attempt to copy other students during the theoretical-practical assessment tests (exams);
- use or attempt to use writings related to the subject during the performance of theoretical-practical assessment tests (exams), when these have not been explicitly allowed.

In these cases, the grade in the Transcript of Records (ToR) will be the lowest value between 3.0 and the weighted average grade (and therefore re-assessment will not be possible).In the assesment of problems and the project, tools to detect code plagiarism will be used.

Note on the planning of the assesment activities: Continuous-assessment dates will be published on Caronte and on the presentation slides. Specific programming may change when necessary. Any such modification will always be communicated to students through Caronte, which is the usual communication platform between lecturers and students.

## Assessment Activities

| Title | Weighting | Hours | ECTS | Learning Outcomes |
|---|---|---|---|---|
| Individual evaluation | 40% | 4 | 0.16 | 3, 6 |
| Problem submission | 30% | 0 | 0 | 10, 7, 9, 8, 1, 5 |

| | | | | |
|---|---|---|---|---|
| Programming project | 30% | 0 | 0 | 10, 2, 7, 9, 8, 5, 4 |

## 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
- https://www.sololearn.com/: SoloLearn
- L. Joyanes, I. Zahonero: Programación en C: metodología, estructura de datos y objetos, Mc Graw-Hill, 2001.
- 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.
- E. Valveny, R. Benavente, A. Lapedriza, M. Ferrer, J. García: Programació en Llenguatge C. Amb 56 problemes resolts i comentats. Servei publicacions UAB, 2009.
- L. Joyanes, A. Castillo, L. Sánchez, I. Zahonero: Programación en C: libro de problemas, Mc Graw-Hill, 2002.
- B.W. Kernighan, D.M. Ritchie: El lenguaje de programación C. 2ª Edición, Prentice Hall, 1986.
- B.W. Kernighan, R. Pike: La Práctica de la Programación. Pearson Educación, 2000.
- M. Jesús Marco Galindo et al. Fonaments de Programació. Publicacions UOC, 2008.
- L. Joyanes Aguilar : Fundamentos de Programación: Algoritmos, Estructuras de Datos y Objetos. 3ª Edición, Mc. Graw-Hill, 2003.
- J. Pujol: Algorismes i Programes. Servei de Publicacions de la UAB, 1996.