Curs 2010-11

21595 Fundamentals of Programming

Degree: Bachelor's degree in Computer Engineering, Bachelor's degree in Telecommunications Network Engineering and Bachelor's degree in Audiovisual Systems Engineering
Course: first
Trimester: second and third
Number of ECTS credits: 8 crèdits
Hours of student dedication: 200 hours
Language or teaching languages: Catalan and Spanish
Teacher:

1. Presentation of the subject

Fundamentals of Programming is integrated into a block of subjects on algorithmics and programming that are carried out in the first and second year of the Degree in Computer Engineering, Degree in Telematics Engineering and Degree in Audiovisual Systems Engineering. For this first subject of the block it is assumed that students do not have previous knowledge about algorithmics and programming. Within this subject, the bases of the algorithmic, data structures and programming in C language are also established. In the subjects Object Oriented and Data Structures and Algorithm Programming (first quarter of the second year) It will be deepened in the competences acquired here, so that at the end of the first quarter of the second year students must be able to develop programs of considerable size with the appropriate data structures, both imperatively and object-oriented. It is also important to note that the bases acquired in Programming Fundamentals are essential in order to implement the practical part of many subjects throughout the studies.

Learning activities have been divided into different categories essentially based on their typology:

• theory sessions: professors present a series of concepts and techniques, apart from examples of their use. Students will have to review the notes and their notes outside the classroom to finish assimilating the contents.

• seminar sessions: students must solve a series of small exercises, putting into practice the concepts and techniques presented in the theory sessions. These activities begin in computer classrooms, and they must be completed outside the classroom.

• practical sessions: students have to solve problems more broadly than the aforementioned exercises, so they must decide which concepts and techniques they should use in each case. These activities begin in computer classrooms, and they must be completed outside the classroom. In addition, in the practical sessions students learn to use the programming tools.

• self-evaluation exercises: exercises so that students, at the end of each didactic unit, verify that the concepts and techniques presented have been assimilated. This activity is carried out by students outside the classroom.

2. Prerequisites for the follow-up of the training itinerary

This subject does not require any prior programming or algorithmic knowledge. In order to carry out some exercises, the knowledge of maths of the baccalaureate level is required.

3. Skills to be achieved

The main objective of this subject is that students acquire the bases of the algorithmic and data structures, as well as that they are able to develop medium-sized programs fluently using the C language.

This chapter details what students are expected to have learned after completing the course. First of all, the general competences refer to skills not directly related to the programming itself, but to the professional field of an engineer. The specific competences are those that refer to aspects of the subject.

3.1. General competences

Instrumental
CG1: Capacity for synthesis
Students must be able to write solutions with the essential elements, in a simple, elegant and more efficient way.

CG2: Capacity for analysis
Students must be able to, from a specific problem, analyze it and propose suitable solutions.

Systemic
CG3: Capacity to apply knowledge in practice
Students must be able to apply the knowledge acquired to solve specific problems, choosing the technique that best suits each case.

CG4: Interest in quality
Students must be able to have their code, as well as efficient, easy to read and maintain. It is also important that you document correctly, both within the same code and in a memory.

3.2. Specific competences

CE1: Capacity to work with programming tools
Students must be able to work with the basic tools of the programming: a compiler and a debug. In addition, you must be able to work with a programming editor and an integrated development environment. This competence is essential for the correct development of others.

CE2: Domain of basic and composite static data types
Students must be able to distinguish the different types of basic and composite static data and to decide the appropriate type in each specific situation.

CE3: Domain of control structures
Students must be able to distinguish the different control structures and decide the most suitable to solve specific problems.

CE4: Ability to solve problems through downward design and mastery of the use of functions and libraries
Students must be able to solve problems of considerable complexity using down-design techniques. In particular, students must understand the function calls and parameter steps, they must master the use and creation of bookstores, and they should be able to divide a problem into the appropriate units.

CE5: Domain of dynamic data types and dynamic memory management
Students must understand the memory management mechanism, as well as the use of pointers and dynamic control of data structures. The processing of text files is also included.

CE6: Documentation and code structure
Students must acquire the habit of structuring and documenting the code properly in order to facilitate their subsequent reading.

CE7: Ability to read (fast) code in C
Students must be able to understand code written by other programmers, relatively quickly.

CE8: Domain of the fundamental elements of algorithm
Students must know and be able to apply the fundamental concepts of algorithmics in a suitable way, such as recursion and search and sort algorithms.

CE9: Capacity to analyze algorithms
Students must know the notation and mechanisms of the analysis of algorithmic complexity, and they must be able to calculate the execution time of a program.

4. Contents

Block 1: Introduction and general concepts

Concepts	Procedures
- Brief history of programming and its languages and paradigms - Mechanisms of compilation and interpretation - Difference between program and algorithm	

Block 2: Basic data types

Concepts	Procedures
- Variables and constants - The different types of basic data: - Numeric types - Characters - Boolean	- Declaration of constants and variables of the different types

Block 3: Expressions, sentences and control structures

Concepts	Procedures
- Formation of expressions - Introduction to Boolean logic - Sistencies or instructions: - Assignments - Input and exit operations - The order of precedence of the operators - The control structures: - The conditional structures - Iterative structures	- Evaluation of expressions - Resolution of small problems using the sentences and adequate control structures

Block 4: Functional decomposition and downward design

Concepts	Procedures
- Downward design - Declaration of functions - Definition of parameters - Function calls and parameters pass - The void type	- Decomposition of problems in subproblems - Resolution of small problems by defining functions appropriately

Block 5: Types of static composite data

Concepts	Procedures
- One-dimensional arrays - Multidimensional arrays - Strings or character strings - Structures (structs)	- Resolution of small problems with the typical operations with each type of data

Block 6: Declaration of own types

Concepts	Procedures
- The types of structures - Definition of typedef types - Type conversions	- Resolution of small problems of definition of own data types

Block 7: The pointers and the dynamic management of memory

Concepts	Procedures
- Declaration of pointers - Direction and indirection operations - Assignment of securities to pointers - The null pointer - Parameter step by reference - Arrays and pointers in C	- Resolution of small problems with the typical operations with pointers - Resolution of small typical problems of arrays using pointers

Block 8: Functional decomposition and downward design (second part)

Concepts	Procedures
- Parameter step by reference - The main function and its arguments - Visibility and scope - Bookstores and preprocessor	- Resolution of problems by decomposition using functions with parameters by reference

Block 9: Text files

Concepts	Procedures
- The file type (FILE) - File operations: - Open - Close - Write - Read - Standard input and output	- Resolution of small problems on typical operations with text files

Block 10: Style and good practices

Concepts	Procedures
- Style, readability and obfuscation - Best practices - Common errors - Code analysis - Debugging	- Troubleshoot error detection and style improvement. - Use of code analysis and debugging tools.

Block 11: Functional decomposition and downward design (third part)

Concepts	Procedures
- Library programming - Segmentation of the code in files - Header files	- Resolution of problems creating and using libraries.

Block 12: Search and sorting algorithms

Concepts	Procedures
- Linear and binary search - Basic sorting algorithms - Bubble - Insertion	- Resolution of problems on search and sort algorithms

- Selection	

Block 13: Recursion

Concepts	Procedures
- Recursion concept - Recursive algorithms - Base case and recurrence - Advantages and disadvantages - Direct and indirect recursion - Transformation to iterative algorithms - Specific examples of recursive algorithms (fractals, towers of Hanoi, Quicksort, etc.)	- Problem solving by defining recursive functions

Bloc 14: Anàlisi d'algorismes

Concepts	Procedures
- Runtime of a program - Asymptotic notation - Execution time calculation	- Resolution of computation problems and comparison of program run time

5. Evaluation

5.1. General evaluation criteria

This subject consists of two main parts:

• Theoretical-practical foundations, which encompass the activities related to the theory and seminar sessions

• Practices

Each one represents 50% of the final grade, although it is necessary to approve the two parts to pass the subject.

Theoretical-practical foundations

There will be two exams, one in the middle of the subject (at the end of the second trimester) and another at the end (at the end of the third trimester). The first one represents 25% of the note and focuses on the block of exercises carried out in the sessions of exercises in the second quarter. The second exam represents the remaining 75% and covers theoretical and practical questions of the whole course.In the sessions of the sessions, the professors will review the work of the students throughout the quarter, and may request the delivery of an exercise at the end of each session. Depending on these revisions and deliveries, the note can be uploaded (but never lowered) from the theoretical-practical basis up to 1.5 points.

In addition, several self-assessment activities have been programmed, one at the end of each didactic unit, with the aim of the students evaluating their progress.

These activities consist of a series of multiple-choice test questions, of a level similar to those of the exams. Also, in seminar activities students can also assess their progress, comparing their solutions with the published results. In any case, these self-assessment activities do not affect the note.

Practices

The note of this part is obtained from three deductible practices, distributed throughout the course. Each of these practices has the same weight in the note (that is, each evaluable practice has a weight of 33.3%). Regarding the grade of deductible practices, apart from the overall assessment, teachers will be able to review progress in the work of the students, throughout the practical sessions. In the event that a group does not make this review, or that this review is not satisfactory, you must make a defense of the practice practice after its delivery.

Apart from the delivering practices, the other practices will help students to evaluate their progress. These activities do not affect the note.

Recovery in September
In the case of not approving the two parts in the June exam, only one approved part will be saved in September in case the note of the other party is equal to or greater than 3. Otherwise, it will have to be returned to take the exam and deliver a new practice (with a different statement).

5.2. Concretion

First evaluable practice

The resolution of a practice in which the student will be able to show the degree of assimilation of the concepts and competences acquired during the first third of the subject will be evaluated. In particular, the competences CE2, CE3 and certain aspects of CE4 and CE6 (commenting code structuring) will be evaluated. The practice will be to solve a problem, including its analysis and the programming of a suitable solution in C language. The student will have to decide on the static data structures as well as the solution flow of the solution and perform a simple downward design. Likewise, the four general competences defined will be valued.

Second practice evaluable
The resolution of a practice in which the student will be able to show the degree of assimilation of the concepts and competences acquired during the first two thirds of the subject, and especially during the second, will be evaluated. In particular, the competences CE2, CE3, CE4, CE5 and CE6 will be assessed. The practice will consist of solving a considerably complex problem, including its analysis, choosing the appropriate (static and dynamic) data structures, downward design, programming an appropriate C language solution (following the recommendations of good practices) and its correct documentation in a report in which the decisions adopted will also be justified. Likewise, the four general competences defined will be valued.

Third practice evaluable
The resolution of a practice in which the student will be able to show the degree of assimilation of the concepts and competences acquired throughout the course will be evaluated. In particular, the competences CE2, CE3, CE4, CE5, CE6, CE8 and CE9 will be evaluated. The practice will consist of solving a considerably complex problem, including its analysis, the choice of adequate (static and dynamic) data structures, downward design, programming of a suitable C language solution (following the recommendations of good practices) and its correct documentation in a memory in which the decisions adopted will also be justified and where a part of analysis of the algorithmic complexity will be included. Likewise, the four general competences defined will be valued.

Midterm exam
The understanding and application of the concepts and techniques acquired during the first half of the subject (corresponding to the second quarter) will be evaluated. Specifically, the CE2, CE3, CE4, CE5 and CE7 competencies will be evaluated. The assessment method consists of a test test of approximately 20 questions with 4 options for each question with only one valid answer. The questions will be extracted (with small modifications) of the proposals for the sessions of exercises and of the activities of self-evaluation. The assessment will take place during the second trimester exams period.

Final exam
The understanding and application of the concepts acquired throughout the entire subject in small programs will be evaluated. Specifically, the competences CE2, CE3, CE4, CE5, CE7, CE8 and CE9 will be

assessed. The assessment method consists of a test type of approximately 30 questions with 4 options for each question with only one valid answer. The assessment will take place during the period of examinations of the third quarter.

6. Bibliography and didactic resources

6.1. Basic bibliography

· Toni Navarrete Terrasa. Introducción a la programación con lenguaje C.
· Jesús Bisbal Riera. Manual d'algorísmica: Recursivitat, complexitat i disseny d'algorismes. Editorial UOC. ISBN: 978-84-9788-570-6
. Brian W. Kernighan, Dennis M. Ritchie: El lenguaje de programación C. Segunda edición. Prentice-Hall. ISBN: 968-880-205-0

6.2. Bibliographie complémentaire

Other books about C:

· Herbert Schildt: C Manual de referencia. Mc Graw Hill. 84-481-0335-1
· James L. Antonakos, Kenneth C. Mansfield Jr.: Programación estructurada en C. Prentice-Hall. ISBN: 84-89660-23-9
· Marco A. Peña, José M. Cela: Introducción a la programación en C. Edicions UPF. ISBN: 84-8301-429-7
· Luis Joyanes, Ignacio Zahonero: Programación en C. Mc Graw Hill. ISBN: 84-481-3013-8
· Félix García, Jesús Carretero, Javier Fernández, Alejandro Calderón: El lenguaje de programación C. Diseño e implementación de programas. Prentice-Hall. ISBN: 84-205-3178-2
· P.J. Plauger, Jim Brodie: C Estándar. Guía de referencia rápida para programadores. Anaya. ISBN: 84-7614-264-1

Other "classic" algorithmic books (more advanced):

· Alfred V. Aho, Jeffrey D. Ullman, John E. Hopcroft: Estructuras de datos y algoritmos. Addison Wesley, 1988. ISBN: 968-444-345-5
· Niklaus Wirth: Algoritmos + estructuras de datos = programas. Ediciones del Castillo, 1980. ISBN: 84-219-0172-9
· Niklaus Wirth: Algoritmia y estructuras de datos. Prentice Hall, 1987. ISBN: 968-880-113-5
· D.E. Knuth: El arte de programar ordenadores (3 volums). Editoria Reverté. ISBN: 84-291-2661-9
· Terrence W. Pratt, Marvin V. Zelkowitz: Lenguajes de programación. Diseño e implementación. Prentice Hall, 3ª edic., 1997. ISBN: 970-17-0046-5
. G. Brassard, P. Bratley: Fundamentos de algoritmia. Prentice-Hall. ISBN: 84-89660-00-X

6.3. Didactics resources

In the Moodle classroom of the subject (in the Classroom Global) will be uploaded the teaching material of the subject. In particular they will upload:
- Notes
- Annotated for the seminar sessions
- Annotated for the practical sessions
- Self-evaluation exercises

7. Methodology

The usual process of learning in a teaching unit begins with a theory session where some theoretical-practical foundations are presented. This activity is carried out in a large group of students. Students will have to complete this activity later with a careful reading of the notes. For example, a typical 2-hour theory session, conveniently utilized, will need a personal work outside the classroom for 1 hour.

Next, one or several sessions of seminars and / or practicals are carried out. In seminar sessions, students put into practice the concepts and techniques presented in the theory session, through the implementation of programs to solve small problems. The objective is for students to consolidate the foundations so that later they can carry out bigger problems.

This activity is done individually, in a group of about 15 students. Each activity of this type is programmed with a duration of 4 hours, of which 2 are done with the help of the teacher. The first exercises of the session, for which the solutions are offered, must be resolved before arriving in the classroom. The teacher may request the delivery of one of the exercises proposed at the end of each session.

In practical sessions, greater problems are proposed, especially in the 3 evaluable practices, which require a prior design of the solution that will be implemented and which integrate different concepts and techniques. In the last practice, all the specific competences that students must acquire in this subject meet. Each activity of this type is done in pairs, in a group of about 30 students, and must continue outside the classroom.

In each session of seminars and internships, it is time to discuss the main problems that have been presented in the previous session.

The last step of the teaching unit is the resolution of some self-assessment exercises through which students can verify they have acquired the competences that will later be evaluated through the partial and final exams.

7.1. Didactic units

Didactic unit 1: First steps: introduction, basic data types and control structures

Content blocks	Learning activities			
1: Introduction and general concepts 2: Basic data types 3: Expressions, sentences and control structures	Sessions of Theory	Seminar sessions	Practical sessions	Self-evaluation
	T1, T2 and T3	S1	P1	A1

Total dedication Didactic Unit 1: 17 hours (10 in the classroom, 7 outside)
Details of the activities:
T1 theory session. 3 hours (2 in the classroom, 1 outside): brief history of the programming and explanation of the models of compilation and interpretation.
T2 and T3 theory sessions. 6 hours (for each session 2 in the classroom, 1 outside): explanation of the types of basic data and main examples of use. Explanation of the formation of expressions and basic sentences with the main examples of its use. Conditional and iterative control structures and main examples of its use.
S1 seminar session. 4 hours (2 in the classroom, 2 outside): exercises on expressions formation and conditional and iterative control structures.
Practical session P1. 3 hours (2 in the classroom, 1 outside): will explain how to install the compiler, and how to edit, compile and execute a program. Small bug programs will be given to students to become accustomed to typical compiler error messages.
Self-assessment A1. 1 hour (outside the classroom): resolution of self-assessment exercises on expression formation and control structures.

Didactic unit 2: Functions and descending design

Content blocks	Activitats d'aprenentatge			
4:Functional decomposition and descending design	Sessions of Theory	Seminar sessions	Practical sessions	Self-evaluation
	T4	S2		A2

Total dedication Didactic Unit 2: 8 hours (4 in the classroom, 4 outside) Details of the activities:
T4 theory session. 3 hours (2 in the classroom, 1 outside): explanation of the basis of the descending design as well as the definition of functions and the mechanism of call and passage of parameters by value, with examples of use.
Seminar S2 session. 4 hours (2 in the classroom, 2 outside): exercises on descending design and step of parameters by value.
Self-assessment A2. 1 hour (outside the classroom): Resolution of self-evaluation exercises on functions and step of parameters by value.

Didactic unit 3: Types of static composite data and type declaration

Content blocks	Learning activities			
5: Types of static composite data 6: Declaration of own types	Sessions of Theory	Seminar sessions	Practical sessions	Self-evaluation
	T5 and T6	S3	P2 and P3	A3

Total dedication Didactic Unit 3: 29 hours (10 in the classroom, 19 outside) Details of the activities:
T5 and T6 theory sessions. 6 hours (for each session 2 in the classroom, 1 outside): explanation of the types of compound data (arrays, chains and structures) with examples of use. Explanation of the definition of data types by the developer, with examples of use.
Seminar S3 session. 4 hours (2 in the classroom, 2 outside): exercises on compound data types.
Practical sessions P2 and P3 (first evaluable practice). 18 hours (for each session, 2 in the classroom, 7 outside): the student must solve (programming in C) some average problems, using arrays, chains and structures, and in some cases defining their own types. Includes the preparation of an explanatory report of the work.
Self evaluation A3. 1 hour (outside the classroom): Resolution of self-assessment exercises on compound data types.

Didactic unit 4: Pointers and step parameters by reference

Content blocks	Learning activities

7: The pointers and the dynamic management of memory 8: Functional decomposition and downward design (second part)	Sessions of Theory	Seminar sessions	Practical sessions	Self-evaluation
	T7 and T8	S4	P4	A4

Total dedication Didactic Unit 4: 18 hours (8 in the classroom, 10 outside) Details of the activities:
T7 and T8 theory sessions. 8 hours (for each session 2 in the classroom, 2 outside): explanation of the operation of pointers with examples of use. Explanation of the step of parameters by reference with examples of use. Explanation of the visibility rules.
Seminar S4 session. 4 hours (2 in the classroom, 2 outside): exercises on pointers and step of parameters by reference.
P4 practice session. 5 hours (2 in the classroom, 3 outside): the student must solve (programming in C) some small problems, using pointers and passage of parameters by reference.
Self evaluation A4. 1 hour (outside the classroom): Resolution of self-assessment exercises on pointers and step of parameters by reference.

Didactic unit 5: Text files

Content blocks	Learning activities			
9: Text files	Sessions of Theory	Seminar sessions	Practical sessions	Self-evaluation
	T9	S5		A5

Total dedication Didactic Unit 5: 8 hours (4 in the classroom, 4 outside) Details of the activities:

T9 theory session. 3 hours (2 in the classroom, 1 outside): explanation of the text files and examples of use.
Seminar S5 session. 4 hours (2 in the classroom, 2 outside): Exercises on text files.

Self evaluation A5. 1 hour (outside the classroom): Resolution of self-evaluation exercises on files.

Didactic unit 6: Good practices and programming of libraries

Content blocks	Learning activities			
10: Style and good practices 11: Functional decomposition and downward design (third part)	Sessions of Theory	Seminar sessions	Practical sessions	Self-evaluation
	T10 and T11		P5, P6, P7 and P8	A6

Total dedication Didactic Unit 6: 33 hours (12 in the classroom, 21 outside) Details of the activities:
T10 theory session. 3 hours (2 in the classroom, 1 outside): explanation of best practices for programming readably and obtaining programs less prone to errors. Explanation of common errors when coding a program. Explanation of code analysis and debugging techniques.
T11 theory session. 3 hours (2 in the classroom, 1 outside): explanation of the process of creating libraries and code segmentation in several files. Review of visibility rules.
Practical session P5. 4 hours (2 in the classroom, 2 outside): an integrated development environment will begin to be used and will explain how to debug a program using the development environment. Students will debug several programs.
Practical session P6. 4 hours (2 in the classroom, 2 outside): will explain how to create a program composed of several modules (or files) using an integrated development environment. Students will create a program divided into several modules.
Practical sessions P7 and P8 (second evaluable practice). 18 hours (for each session, 2 in the classroom, 7 forums): the student must solve (programming in C) a problem of considerable size that integrates pointers, step of parameters by reference, text files and segmentation of the code in several files. The solution must be programmed following the recommendations of good practices. Includes the preparation of an explanatory report of the work.

Self evaluation A6. 1 hour (outside the classroom): Resolution of self-assessment exercises about style, good practices, and segmentation of the code in modules or files.

Didactic unit 7: Search, ordering and recursion

Content blocks	Learning activities			
12: Search and sorting algorithms 13: Recursion	Sessions of Theory	Seminar sessions	Practical sessions	Self-evaluation
	T12, T13, T14 and T15	S6 and S7		A7

Total dedication Didactic Unit 7: 24 hours (12 in the classroom, 12 outside) Details of the activities:
T12 theory session. 3 hours (2 in the classroom, 1 outside): explanation of the linear and binary search algorithms, as well as the basic sorting algorithms (bubble, insertion and selection).
Theory sessions T13, T14 and T15. 12 hours (T13 2 in the classroom, 2 outside, T14 2 in the classroom, 3 outside, T15 2 in the classroom and 1 outside): explanation of the recursion concept. Explanation of the bases of recursive algorithms and their advantages and disadvantages. Explanation of the different types of recursion. Explanation of the transformation of recursive algorithms to iterative algorithms. Discussion of several concrete examples of recursive algorithms.

Seminar S6. 4 hours (2 in the classroom, 2 outside): Exercises on search algorithms, sort algorithms and recursive algorithms.
S7 seminar session. 4 hours (2 in the classroom, 2 outside): Exercises on recursive algorithms.
Self evaluation A7. 1 hour (outside the classroom): Resolution of self-assessment exercises on search algorithms, sorting algorithms and recursion.

Didactic unit 8: Analysis of algorithms

Content blocks	Learning activities			
14: Anàlisi d'algoritmes	Sessions of Theory	Seminar sessions	Practical sessions	Self-evaluation
	T16, T17 i T18	S8	P9 i P10	A7

Total dedication Didactic Unit 8: 33 hours (12 in the classroom, 21 outside) Details of the activities:

T16 and T17 theory sessions. 7 hours (T16 2 in the classroom, 2 outside; T17 2 in the classroom, 1 outside): explanation of the factors involved in the efficiency of an algorithm. Explanation of asymptotic notation. Explanation of the procedures to calculate the execution time of an algorithm, with illustrative examples.

T18 theory session. 3 hours (2 in the classroom, 1 outside): general review and resolution of doubts.

Seminar S8. 4 hours (2 in the classroom, 2 outside): Exercises on the analysis of the complexity of algorithms.

Practical sessions P9 and P10 (third practical assessment). 18 hours (for each session, 2 in the classroom, 7 forums): the student must solve (programming in C) a problem of considerable size that integrates all the concepts and techniques of the subject (including recursive algorithms and analysis of the algorithmic complexity). The solution must be programmed following the recommendations of good practices. Includes the preparation of an explanatory report of the work.

Self evaluation A8. 1 hour (outside the classroom): Resolution of self-assessment exercises on the analysis of the complexity of algorithms.

Note: in addition to the hours of work planned and indicated in the description of the didactic units, 10 hours (8.5 + 1.5) are reserved to prepare and carry out the partial exam, and 20 (17.5 + 2.5) hours for the final exam.

Total dedication of the subject: 200 hours (76 in the classroom, 124 outside)

8. Programming of activities

8.1. Hours of dedication of the students

	Activity	Hours in the classroom			Hours outside the classroom	
		Large group	Medium group	Small group		
Didactic unit 1	T1	2			1	
	T2	2			1	
	P1		2		1	

	T3	2			1	
	S1			2	2	
	A1				1	
Didactic unit 2	T4	2			1	
	S2			2	2	
	A2				1	
Didactic unit 3	T5	2			1	
	S3			2	2	
	T6	2			1	
	P2		2		7	
	P3		2		7	
	A3				1	
Didactic unit 4	T7	2			2	
	S4			2	2	
	T8	2			2	
	P4		2		3	
	A4				1	
Didactic unit 5	T9	2			1	
	S5			2	2	
	A5				1	
Didactic unit 6	T10	2			1	
	P5		2		2	
	T11	2			1	

	P6		2		2	
	P7		2		7	
	P8		2		7	
	A6				1	
Didactic unit 7	T12	2			1	
	T13	2			2	
	T14	2			3	
	S6			2	2	
	T15	2			1	
	S7			2	2	
	A7				1	
Didactic unit 8	T16	2			2	
	T17	2			1	
	S8			2	2	
	P9		2		7	
	T18	2			1	
	P10		2		7	
	A8				1	
Exams	Midterm	1,5			8,5	
	Final	2,5			17,5	
Total		40	20	16	124	200 (ECTS * 25)

8.2 Programming of face-to-face sessions

In this section tables with the programming of the sessions are included. In the tables, the sessions T are of theory, S are of seminary, and P are of practices. Keep in mind that, for each group, half of your seminar subgroups have one-day seminar sessions, and the other half has one another day different On the other hand, all the practice subgroups of each group have the practice sessions the same day and at the same time, although each subgroup has them in a different classroom. To see the classrooms for each session, see:

http://www.upf.edu/esup/docencia/graus/http://www.upf.edu/esup/docencia/graus/