Universidad del País Vasco Euskal Herriko Unibertsitatea

COURSE GUIDE	2025/26		
Faculty	345 - Faculty of Engineering - Bilbao	**Cycle**	.
Degree	GITECI30 - Bachelor`s Degree in Industrial Technology Engineering	**Year**	First year

COURSE

26570 - Computer Science	**Credits, ECTS:**	6

COURSE DESCRIPTION

This course is the first contact of students with computer science, in particular with "software". Therefore, it is not required any prior knowledge of programming or automatic data processing. However, the student must have some knowledge of computers and how to use windows based software to achieve a better performance during lab classes.

The examples, papers and projects undertaken in this course are related to others subjects in the basic module (linear algebra, calculus, physics). However, the knowledge acquired will be applied in other subjects throughout the career, particularly in relation to the practical aspects of them.

COMPETENCIES/LEARNING RESULTS FOR THE SUBJECT

Taking into account the new teaching model, in which is crucial to monitor the student learning, the teaching objectives are specified in objective terms related to the acquired acquired knowledge and skills.

* Skills

This course develops the CM03 basic module skills designated: "Basic knowledge on using and programming computers, operating systems, databases and software engineering with application in the engineering". Likewise, other skills related to the whole career will be acquired.

* Learning Outcomes

This course enables students to:

- Set out clearly the functional description of the hardware components of a processor.
- Differentiate the different levels of software involved in solving a problem using a data.
- Differentiate the different types of programming languages and understand the fundamental concepts of computer languages of high and low level.
- Coding, setting up and running simple programs in high-level language.
- Use correctly the essential aspects of programming: data types, control structures, data structures, subroutines, and files.
- Self-assess the work done and identify mistakes and areas for improvement.
- Design simple algorithms: search, insertion, deletion and sorting.
- Acquire information independently, explaining to colleagues and ensure that they have assimilated.
- Agree on decisions with mates on aspects of the implementation of a program, specify the tasks, distribute evenly and integrate results.

Theoretical and Practical Contents

Topics of the master classes:

Unit 1. General concepts: Hardware schema, software schema, programming languages ¿¿and methodology, and algorithms.
Unit 2. Programming: General structure of a program.
Unit 3. Basic data types and input-output operations.
Unit 4. Control statements: sequential (assignment), conditional and iterative.
Unit 5. Subprograms: types and parameter passing.
Unit 6. Homogeneous data structures: Arrays (declaration, use, and algorithms).
Unit 7. Homogeneous data structures: Strings (declaration, use, and algorithms).
Unit 8. Heterogeneous data structures: Structures (declaration and use).
Unit 9. Permanent data storage: Files.

Content of the practical classes (seminars and computer practices): Development of activities to put into practice the concepts and elements explained in the lectures.

TEACHING METHODS

The teaching will be three types: Master classes, seminars and computer practices. Master classes aim at the theoretical explanation of the concepts. Nevertheless, different practical examples, seminars and computer practices help the student to work on the theoretical concepts.

The theoretical programme is continuously developed at a rate of 2 hours per week. The seminars and the computer practices are weekly alternated.

This subject consists of six seminar sessions of 2 hours each which are distributed within the academic. During these sessions, students put into practice the concepts related to the software design methodologies.

In alternate weeks, the course consists of computer practical sessions (2 hours) in which students work individually, particularly in programming in a high level programming language and performing all the required programming stages: design, coding, verification and validation of the solutions to problems proposed by the professor prior to class.

During the master classes as well as the seminar and computer practice sessions, the latest ICT technologies will be also introduced and practiced.

In case of sanitary emergency, the university will activate an online teaching protocol, and students will be informed accordingly.

LEARNING RESOURCES

Computational Centres for the realization of individual computer practices and seminars.
Virtual classroom materials for the distribution of work, monitoring the work and keeping the information related to the sections covered in this document (software, books, tools...).

TYPES OF TEACHING

Types of teaching	M	S	GA	GL	GO	GCL	TA	TI	GCA
Hours of face-to-face teaching	30	12			18				
Horas de Actividad No Presencial del Alumno/a	45	18			27				

Legend:			
	M: Lecture-based	S: Seminar	GA: Applied classroom-based groups
	GL: Applied laboratory-based groups	GO: Applied computer-based groups	GCL: Applied clinical-based groups
	TA: Workshop	TI: Industrial workshop	GCA: Applied fieldwork groups

Evaluation methods

- End-of-course evaluation

Evaluation tools and percentages of final mark

- Written test, open questions 70%
- Individual assignments 30%

ORDINARY EXAMINATION PERIOD: GUIDELINES AND OPTING OUT

The evaluation of the subject is performed considering the skills that the students should acquire during the course. Therefore, we use different instruments to measure the acquisition of those skills.

The evaluation of the subject is based on both a continuous evaluation (30%) and a final exam (70%).

Continuous evaluation (30%): Seminar (15%) and Computer Practices (15%)
- Reports and exams linked to the seminars (15 %)
- Reports and controls of computer practices (15%)

Final exam (70%):
- Multiple choice tests or open-ended questions (21%)
- Program Development (49%)

* In order to pass the subject, it is mandatory to achieve at least 50% of the grade in the final exam. If the 50% of the grade in the Final exam is not obtained, the final grade will be the one obtained on the final exam.

Refusing the continuous evaluation:
- During the first 9 weeks of the course, the student can refuse the continuous evaluation by sending a formal writing to the assigned professor
-A complementary practical computer-based exam will be carried out to evaluate this 30%. However, in order to pass the subject, it is mandatory to achieve at least 50% of the grade in that complementary exam. If the 50% of the grade in the Final exam or in the complementary computer-based exam is not obtained, the final grade will be the one obtained on test where the minimum requirements have not been met.

A student who does not present the final exam will be graded with a "Not taken"

EXTRAORDINARY EXAMINATION PERIOD: GUIDELINES AND OPTING OUT

Final exam (100%):
- Multiple choice tests or open-ended questions (30%)
- Program Development (70%)

* In order to pass the subject, it is mandatory to achieve at least 50% of the grade in the Program Development part within the final exam. Otherwise, the final grade will be the one obtained on the "Program Development" part of the exam.

A student who does not present the final exam will be graded with a "Not taken"

MANDATORY MATERIALS

Theory notes and practices available on the course's virtual platform (eGela).

Software:

Software development environment of a high-level language.
Virtual platform eGela is the teacher-student communication where you will find information and timely notifications.

BIBLIOGRAPHY

Basic bibliography

"Introducción a la informática", Alberto Prieto, Antonio Lloris Ruiz, Juan Carlos Torres Cantero, McGraw-Hill
"Fundamentos de programación. Algoritmos y estructuras de datos y Objetos", Luis Joyanes Aguilar. McGraw-Hill.
"Algoritmos + Estructuras de Datos = Programas", Niklaus Wirth, Editorial Castillo
"C in a nutshell, the definitive reference" Peter Prinz y Tony Crawford, O¿Reilly (1st ed. 2002, 4th ed. 2010)
"C, the complete reference" Herbert Schildt, McGraw Hill (2000)
"C Programming Absolute Beginner¿s Guide", Greg Perry and Dean Miller, Pearson (2013)
"C Programazio Lengoaia", Iñaki Alegria eta Nestor Garai, Elhuyar (1995)

Detailed bibliography

"Fundamentos de los computadores. Estructura, funcionamiento interno, software de sistemas", Pedro de Miguel Anasagasti, Editorial McGraw-Hill
"C in depth", Deepali Srivastava, BPB Publications
"The Programming language C", Brian Kernighan and Dennis Ritchie, Prentice-Hall (1988)

Journals

Web sites of interest

https://www.codeblocks.org/
https://www.geeksforgeeks.org/c-programming-language/
https://es.wikibooks.org/wiki/Fundamentos_de_programaci%C3%B3n
http://www.lawebdelprogramador.com/
http://www.nachocabanes.com

OBSERVATIONS