Teaching Guide					
Identifying Data					2019/20
Subject (*)	Programming I			Code	614G01001
Study programme	Grao en Enxeñaría Informática				
Descriptors					
Cycle	Period	Year		Type	Credits
Graduate	1st four-month period	First		Basic training	6
Language	SpanishEnglish				
Teaching method	Face-to-face				
Prerequisites					
Department	Ciencias da Computación e Tecnoloxías da InformaciónComputación				
Coordinador	Rabuñal Dopico, Juan Ramon		E-mail	juan.rabunal@udc.es	
Lecturers	Arcay Varela, Bernardino		E-mail	bernardino.arcay@udc.es	
	Boveda alvarez, Maria del Carmen			carmen.boveda@udc.es	
	Castro Martinez, Alfonso			alfonso.castro@udc.es	
	Cedrón Santaeufemia, Francisco Abel			francisco.cedron@udc.es	
	Martinez Perez, Maria			maria.martinez@udc.es	
	Munteanu , Cristian Robert			c.munteanu@udc.es	
	Rabuñal Dopico, Juan Ramon			juan.rabunal@udc.es	
Web	moodle.udc.es/				

General description	This is an undergraduate course introduction to programming. The student will learn about the following: - The importance of the objectives of programming; - The general aspects of the languages and the programming paradigms; - The pseudocode and syntax of C language in order to be able to describe algorithms and applications; - The steps to follow for building an application and its main components; - The basic data types using C language; - The control structures and the differences between them; - All aspects related to the implementation of functions and procedures; - Tracking an algorithm in pseudocode and the source program using C language; - Explaining what is the output of the code and finding the potential errors; - Solving small algorithms and programs starting from low- to moderate-difficulty problems: given the objectives of the program, to choose and use the best data types and structures, the control structures, to decompose and implement the functions and procedures; - Using an appropriate programming style with identifiers, comments, good design of procedures and functions. At the end of the course, students will have the following abilities: - To understand and master the basics of discrete, logic, algorithmic mathematics and computational complexity, and their application for solving engineering problems; - Basic knowledge on using and programming computers, operating systems, databases and software with applications in engineering; - Knowledge of the structure, organization, operation and interconnection of computer systems, foundations of programming and their application for solving engineering problems. - Knowledge, design and efficient use of the types and structures more suited to solve a data problem. - Solving problems; - Teamwork; - Capacity for analysis and synthesis; - Ability to organize and plan; - Information Management Skills; - How to make decisions; - Concern for quality of programming and applications; - Using basic tools of information technology and communications (ICT) necessary for the exercise of their profession and for learning throughout life; - Evolve to exercise an open, educated, critical, committed, democratic and united citizenship, capable of analyzing reality, diagnose problems, formulate and implement solutions based on knowledge and for the common good; - Critically assess the knowledge, technology and information available to solve real problems; - As professionals and citizens, assume the importance of learning throughout life. - Value the importance of research, innovation and technological development in the social, economic and cultural development of society.

Study programme competences / results	
Code	Study programme competences / results
A4	Coñecementos básicos sobre o uso e a programación dos ordenadores, sistemas operativos, bases de datos e programas informáticos con aplicación na enxeñaría.
A5	Coñecemento da estrutura, organización, funcionamento e interconexión dos sistemas informáticos, os fundamentos da súa programación e a súa aplicación para a resolución de problemas propios da enxeñaría.
B1	Capacidade de resolución de problemas
B3	Capacidade de análise e síntese
B4	Capacidade para organizar e planificar
C3	Utilizar as ferramentas básicas das tecnoloxías da información e as comunicacións (TIC) necesarias para o exercicio da súa profesión e para a aprendizaxe ao longo da súa vida.

C6	Valorar criticamente o coñecemento, a tecnoloxía e a información dispoñible para resolver os problemas cos que deben enfrontarse.
C7	Asumir como profesional e cidadán a importancia da aprendizaxe ao longo da vida.

Learning outcomes			
Learning outcomes	Study programme competences / results		
Knowing and understanding the importance of the programming objectives. Knowing the general aspects of programming languages and paradigms. Knowing the pseudocode and syntax of C language in order to be able to describe algorithms and programs. Knowing the steps to follow for building an application and its main components. Knowing the basic data types using C language. Knowing the control structures for structured programming and the differences between them. Knowing all aspects related to the implementation of functions and procedures.	A4 A5	B1 B3 B4	
Knowing and understanding the importance of the programming objectives. Knowing the general aspects of programming languages and paradigms. Knowing the pseudocode and syntax of C language in order to be able to describe algorithms and programs. Knowing the steps to follow for building an application and its main components. Knowing the basic data types using C language. Knowing the control structures for structured programming and the differences between them. Knowing all aspects related to the implementation of functions and procedures.	A4 A5	B1 B3 B4	
Being able to track an algorithm (in pseudocode) or program (C language), explaining what it is generating and finding possible errors. Being able to solve small algorithms and programs. Solving small algorithms and programs starting from low-to moderate-difficulty problems: given the objectives of the program, to choose and use the best data types and structures, the control structures, to decompose and implement the functions and procedures. Using an appropriate programming style. Learning to make good use of identifiers, appropriate comments, the establishment of preconditions and postconditions, and the good design of procedure and function interfaces.	A4 A5	B1 B3 B4	C3 C6 C7
Being able to track an algorithm (in pseudocode) or program (C language), explaining what it is generating and finding possible errors. Being able to solve small algorithms and programs. Solving small algorithms and programs starting from low-to moderate-difficulty problems: given the objectives of the program, to choose and use the best data types and structures, the control structures, to decompose and implement the functions and procedures. Using an appropriate programming style. Learning to make good use of identifiers, appropriate comments, the establishment of preconditions and postconditions, and the good design of procedure and function interfaces.		B1 B3 B4	C3 C6 C7
Independent learning, planning activities to develop, capacity for abstraction, decision making, initiative and participation.		B3 B4	C3 C6 C7
Independent learning, planning activities to develop, capacity for abstraction, decision making, initiative and participation.		B3 B4	C3 C6 C7

Contents	
Topic	Sub-topic

1 BASIC CONCEPTS	1.1 Algorithms 1.1.1 Representation of algorithms 1.2 Programs (applications) 1.2.1 Types of programs 1.3 Programming languages 1.3.1 A historical overview 1.3.2 Classification of languages 1.3.3 Most important language instructions 1.3.4 Properties of languages 1.4 Code compilers 1.5 The structure of a program 1.6 Elements of a program 1.6.1 Predefined symbols 1.6.2 Special symbols 1.6.3 Identifiers 1.6.4 Labels 1.6.5 Comments 1.6.6 Directives 1.6.7 Constants 1.6.8 Numbers 1.6.9 Strings 1.6.10 Variables: declaration and initiation 1.6.11 Variables: memory address 1.7 Output and input 1.7.1 Output sentences 1.7.2 Input sentences 1.8 Data types and operators 1.8.1 Data types 1.8.2 Operators 1.8.3 Expressions

2 Control statements	2.1 Sequential flow 2.2 Alternative syntax 2.2.1 Single statement 2.2.2 Multiple statement 2.3 Repetitive statement 2.3.1 Introduction 2.3.2 Variables associated with loops 2.3.3 Types of loops 2.3.4 FOR loop 2.3.5 Equivalence between loops 2.3.6 Errors with loops 2.3.7 Loop design
3 Program structure	3.1 Functions and Procedures 3.1.1 Types of functions and procedures 3.1.2 Value and reference parameters 3.1.3 Protected parameters 3.1.4 Memory management for procedures 3.1.5 Global and local variables 3.1.6 Side Effects 3.2 Recursion 3.2.1 Why recursion 3.2.2 Infinite recursion
4 Simple data structures	4.1 Arrays and Matrix 4.1.1 ARRAY data type 4.1.2 Declaring an Array 4.1.3 Arrays of more than one dimension 4.1.4 Operations with Arrays and Matrix 4.2 Records 4.2.1 Record data type 4.2.2 Record operations 4.3 Strings 4.3.1 Fixed-length strings 4.3.2 Variable-length strings 4.4 Basic Operations on Arrays 4.4.1 Search operations 4.4.2 Sort operations
5 Input / Output	5.1 Files 5.2 Types 5.3 Operations and access modes 5.4 Specific predefined functions and procedures

Planning				
Methodologies / tests	Competencies / Results	Teaching hours (in-person & virtual)	Student?s personal work hours	Total hours

Guest lecture / keynote speech	A4 A5 B1 B3 C6 C7	30	30	60
Laboratory practice	A4 A5 B1 B3 B4 C3 C6 C7	20	50	70
Seminar	B4 C3 C6	8	10	18
Personalized attention		2	0	2
(*)The information in the planning table is for guidance only and does not take into account the heterogeneity of the students.				

Methodologies	
Methodologies	**Description**
Guest lecture / keynote speech	In the theory sessions, the teacher describes the objectives and contents of the course as a personal point of view on programming. The teacher will present the available programming methods and tools. In the case of special issues, the students should deepen their self-learning. The goal is that students learn to create algorithms for real problems, to use the basic data structures and to apply programming techniques for simple problems. The course programming language is represented by C language.
Laboratory practice	In the practice sessions, students will write program pseudocodes and they will encode them with C language, they will compile, run and check the codes. The goal of the teacher is to supervise the code generated by the student, to resolve doubts, to correct bad programming styles and logical errors (C errors will be detected by the compiler). The problems will be available on the UDC Moodle before each laboratory class. The Moodle forum will be used to respond to any related question about any aspect of the course. This way, all the students are able to have the same information in the same time.
Seminar	In the seminar sessions practical exercises will be conducted in order to detect and address the students knowledge gaps.

Personalized attention	
Methodologies	**Description**
Laboratory practice Seminar Guest lecture / keynote speech	In both the lectures and the labs sessions, there is a personalized attention of the student, based on the type of class, detecting the level of assimilation and understanding of the issues and explaining the practices required to implement. In the seminar sessions, students can be reached to understand their gaps, and they are shown how to close them. Students with part-time enrollment should, at the beginning of the course, talk to the teachers in charge of their group.

Assessment			
Methodologies	**Competencies / Results**	**Description**	**Qualification**
Laboratory practice	A4 A5 B1 B3 B4 C3 C6 C7	All the tests will be held on computer (programming code, writing pseudocodes, short questions). There is no C code programming on paper. The students will randomly choose the exam tasks. Any attempt to cheat during an exam will be punished with grade 0.	30

Guest lecture / keynote speech	A4 A5 B1 B3 C6 C7	Course grade = continuous assessment grade + final exam grade Continuous assessment grade is divided into two parts: 1. First test in the middle of the course (1 point): pseudocode and code programming with each code line explained for one random exercise. 2. Second test in the last week of the course (3 points): code programming only for two random exercises. The final exam will consist of several exercises where the student must develop code (6 points). The July extraordinary exam will consist of several problems to develop code (6 points). This grade will be added to that one obtained in the continuous evaluation.	70

Assessment comments
The final grades will be determined by the continuous assessment grades and the one obtained in the final exam. The final exam will consist of several programming exercises in the language used in the practice sessions.

Sources of information	
Basic	- Kernighan, Brian W. Englewood Cliffs (1988). The C Programming Language. New Jersey. Prentice Hall - K.N. King (2008). C programming. A modern Approach. Second Edition.. - James L. Antonakos , Kenneth C. Mansfield (2004). Programación estructurada en C. Madrid. Prentice-Hall - Luis Joyanes Aguilar, Ignacio Zahonero Martínez (2005). Programación en C metodología, algoritmos y estructura de datos. Madrid. McGraw-Hill - José R. García-Bermejo Giner (2008). Programación estructurada en C. Pearson - Luis Joyanes Aguilar (2011). Fundamentos de programación : algoritmos, estructuras de datos y objetos. Madrid. McGraw-Hill
Complementary	- Gabriela Márquez, Sonia Osorio, Noemí Olvera (2011). Introducción a la Programación Estructurada en C. Pearson - Andrés Marzal, Isabel García (2017). Introducción a la Programación con C. Publicacions de la Universitat Jaume I. Servei de Comunicació i Publicacions - Luis Joyanes Aguilar (2002). Programación en C. libro de problemas. Madrid. McGraw-Hill

Recommendations
Subjects that it is recommended to have taken before
Subjects that are recommended to be taken simultaneously
Computer Science Preliminaries/614G01002
Subjects that continue the syllabus
Programming II/614G01006
Other comments
The student must keep in mind that you must perform a very important self-taught work by following the flow: reading, attending, understanding, asking, studying and practicing.   Reading: Read the issue to be addressed before attending the theoretical sessions. Even if it seem strange, it is very important. Attending classes: Pay attention in class, do not rest, do not spend time just to take notes. Understanding: Understand the theory sessions and, if not, please ask. Asking: Ask what you do not understand. You have this right. Studying: to retain what you understood. Practicing: Program many applications, which are asked by the professor and others on their own.   Programming is a subject that cannot be learned in two days. The student must go maturing concepts, and program many applications. During these classes, the students will be continuous evaluated.

(*)The teaching guide is the document in which the URV publishes the information about all its courses. It is a public document and cannot be modified. Only in exceptional cases can it be revised by the competent agent or duly revised so that it is in line with current legislation.