



Teaching Guide				
Identifying Data			2017/18	
Subject (*)	Programming I		Code	614G01001
Study programme	Grao en Enxeñaría Informática			
Descriptors				
Cycle	Period	Year	Type	Credits
Graduate	1st four-month period	First	FB	6
Language	SpanishEnglish			
Teaching method	Face-to-face			
Prerequisites				
Department	Computación			
Coordinador	Boveda alvarez, Maria del Carmen		E-mail	carmen.boveda@udc.es
Lecturers	Arcay Varela, Bernardino Boveda alvarez, Maria del Carmen Castro Martinez, Alfonso Garcia Martin, Esteban Martinez Perez, Maria Munteanu , Cristian Robert Rabuñal Dopico, Juan Ramon		E-mail	bernardino.arcay@udc.es carmen.boveda@udc.es alfonso.castro@udc.es esteban.garcia@udc.es maria.martinez@udc.es c.munteanu@udc.es juan.rabunal@udc.es
Web	moodle.udc.es/			



General description	<p>This is an undergraduate course introduction to programming. The student will learn about the following:</p> <ul style="list-style-type: none">- The importance of the objectives of programming;- The general aspects of the languages and the programming paradigms;- The pseudocode and syntax of Pascal ISO10206 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 Pascal ISO_10206;- 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 Pascal ISO-10206;- 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. <p>At the end of the course, students will have the following abilities:</p> <ul style="list-style-type: none">- 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	
Code	Study programme competences
A3	Capacidade para comprender e dominar os conceptos básicos de matemática discreta, lóxica, algorítmica e complexidade computacional e a súa aplicación para a resolución de problemas propios da enxeñaría.
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.
A13	Coñecemento, deseño e utilización de forma eficiente dos tipos e estruturas de datos máis adecuados á resolución dun problema.
B1	Capacidade de resolución de problemas
B2	Traballo en equipo



B3	Capacidade de análise e síntese
B4	Capacidade para organizar e planificar
B5	Habilidades de xestión da información
B6	Toma de decisións
B7	Preocupación pola calidade
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.
C4	Desenvolverse para o exercicio dunha cidadanía aberta, culta, crítica, comprometida, democrática e solidaria, capaz de analizar a realidade, diagnosticar problemas, formular e implantar solucións baseadas no coñecemento e orientadas ao ben común.
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.
C8	Valorar a importancia que ten a investigación, a innovación e o desenvolvemento tecnolóxico no avance socioeconómico e cultural da sociedade.

Learning outcomes			
Learning outcomes	Study programme competences		
Knowing and understanding the importance of the programming objectives. Knowing the general aspects of programming languages and paradigms. Knowing the pseudocode and syntax of Pascal language (ISO 10206 standard) 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 Pascal ISO 10206. 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		
Being able to track an algorithm (in pseudocode) or program (Pascal ISO 10206), 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.	A3 A5 A13	B1 B2 B3 B4 B5 B6 B7	C3
Independent learning, planning activities to develop, capacity for abstraction, decision making, initiative and participation.			C3 C4 C6 C7 C8

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 Description of languages

1.5.1 Backus-Naur Form notation

1.5.2 Conway diagrams

1.6 The structure of a program

1.7 Elements of a program

1.7.1 Predefined symbols

1.7.2 Special symbols

1.7.3 Identifiers

1.7.4 Tags

1.7.5 Comments

1.7.6 Directives

1.7.7 Constants

1.7.8 Numbers

1.7.9 Strings

1.7.10 Variables: declaration and initiation

1.8 Output and input

1.8.1 Output sentences

1.8.2 Input sentences

1.9 Data types and operators

1.9.1 Data types

1.9.2 Operators

1.9.3 Expressions



2 Control statements	<ul style="list-style-type: none">2.1 Sequential flow2.2 Alternative syntax<ul style="list-style-type: none">2.2.1 IF statement2.2.2 CASE statement2.3 Repetitive statement<ul style="list-style-type: none">2.3.1 Introduction2.3.2 Variables associated with loops2.3.3 WHILE loop2.3.4 Examples to perform in class2.3.5 FOR loop2.3.6 REPEAT loop2.3.7 Equivalence between loops2.3.8 Errors with loops2.3.9 Loop design
3 Program structure	<ul style="list-style-type: none">3.1 Procedures<ul style="list-style-type: none">3.1.1 Description3.1.2 Types of procedures3.1.3 Value and reference parameters3.1.4 Protected parameters3.1.5 Memory management for procedures3.1.6 Global and local variables3.1.7 Side Effects3.2 Functions<ul style="list-style-type: none">3.2.1 Description3.2.2 Predefined functions3.2.3 User-defined functions3.3 Recursion<ul style="list-style-type: none">3.3.1 Why recursion3.3.2 Direct and indirect recursion. FORWARD directive.3.3.3 Infinite recursion



4 Simple data structures	4.1 Arrays 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 4.1.5 Arrays as parameters 4.1.6 Array-type functions 4.1.7 Constants array type 4.2 Records 4.2.1 RECORD data type 4.2.2 WITH statement 4.2.3 Record operations 4.2.4 Variant records 4.2.5 Records as parameters 4.2.6 Record type functions 4.2.7 Record type constants 4.3 Strings 4.3.1 Fixed-length strings 4.3.2 Variable-length strings 4.4 Sets 4.4.1 Operations and relationships between sets 4.4.2 Processing sets 4.5 Basic Operations on Arrays
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	Ordinary class hours	Student's personal work hours	Total hours
Guest lecture / keynote speech	A3 A4 A5 A13 B5 B7 C3 C4 C6 C7	30	30	60
Seminar	C8	8	10	18
Laboratory practice	A4 A5 A13 B1 B2 B3 B4 B6 B7	20	50	70
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	<p>In the theory sessions, the teacher describes the objectives and contents of the course as a personal point of view on programming.</p> <p>The teacher will present the available programming methods and tools. In the case of special issues, the students should deepen their self-learning.</p> <p>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 Extended Pascal, ISO 10206 standard.</p> <p>The course slides will be available on UDC Moodle before each course lecture.</p>
Seminar	<p>In the seminar sessions practical exercises will be conducted in order to detect and address the students knowledge gaps.</p>
Laboratory practice	<p>In the practice sessions, students will write program pseudocodes and they will encode them with Extended Pascal ISO 10206 Standard, they will compile, run and check the codes.</p> <p>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 (Pascal errors will be detected by the compiler).</p> <p>The problems will be available on the UDC Moodle before each laboratory class.</p> <p>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.</p>

Personalized attention

Methodologies	Description
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.
Laboratory practice	In the seminar sessions, students can be reached to understand their gaps, and they are shown how to close them.
Seminar	Students with part-time enrollment should, at the beginning of the course, talk to the teachers in charge of their group.

Assessment

Methodologies	Competencies	Description	Qualification
---------------	--------------	-------------	---------------



Guest lecture / keynote speech	A3 A4 A5 A13 B5 B7 C3 C4 C6 C7	<p>Course grade = continuous assessment grade + final exam grade</p> <p>Continuous assessment grade is divided into two parts:</p> <ol style="list-style-type: none"> 1. First test in the middle of the course (2 points): 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. <p>The final exam will consist of three exercises where the student must develop code (5 points).</p> <p>The July extraordinary exam will consist of three problems to develop code (5 points). This grade will be added to that one obtained in the continuous evaluation.</p> <p>The evaluation in December consists into a single test with three problems (10 points).</p>	70
Laboratory practice	A4 A5 A13 B1 B2 B3 B4 B6 B7	<p>All the tests will be held on computer (programming code, writing pseudocodes, short questions). There is no Pascal code programming on paper.</p> <p>The students will randomly choose the exam tasks. Any attempt to cheat during an exam will be punished with grade 0.</p>	30

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 three programming exercises in the language used in the practice sessions.

Sources of information

Basic	<ul style="list-style-type: none"> - Carmen Bóveda, Esteban García, Alejandra Martínez (2014). Programación estructurada en un lenguaje didáctico y estándar. La Coruña , Reprografia del Noroeste - Carmen Bóveda, Esteban García, Alejandra Martínez (2016). Problemas en Pascal Estándar ISO-10206. La Coruña , Reprografia del Noroeste - ISO (1990). Extended Pascal ISO 10206. ISO
Complementary	<ul style="list-style-type: none"> - Leestma, S e Nyhoff, L.. (1999). Programación en Pascal. Madrid Prentice Hall - Leestma, S e Nyhoff, L.. (1993). Pascal Programming and Problem Solving. Prentice Hall - Grogono, P (). Programación en Pascal. Addison-Wesley I - Valls, J. e Camacho, D. (2004). Programación estructurado y algoritmos en Pascal. Madrid Prentice Hall

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.