UNIVERSIDAD AUTÓNOMA DE MADRID

17817 - PROGRAMMING I

Information of the subject

Code - Course title: 17817 - PROGRAMMING I
Degree: 473 - Graduado/a en Ingeniería Informática
474 - Graduado/a en Ingeniería Informática y Matemáticas
722 - Graduado/a en Ingeniería Informática
Faculty: 350 - Escuela Politécnica Superior
Academic year: 2019/20

1. Course details

1.1. Content area

Programming and data structures, Computing

1.2. Course nature

Basic Training

1.3. Course level

1.4. Year of study

473 - Graduado/a en Ingeniería Informática: 1
722 - Graduado/a en Ingeniería Informática: 1
474 - Graduado/a en Ingeniería Informática y Matemáticas: 1

1.5. Semester

Primer semestre

1.6. ECTS Credit allotment

6.0

1.7. Language of instruction

English

1.8. Prerequisites

This subject is taught in the first semester of the first course. Therefore, no previous programming requirement is required.

1.9. Recommendations

Programming I is part of Matter 1 of the Programming and Data Structures module of the curriculum. This subject is split into three semester subjects that complement each other: Programming I, Programming II and Programming Project (the last two are taught in the second semester of the first course). It is also complemented with the subject Seminar-workshop of Software, of the seminar-workshop module of Computer Science, which is taught simultaneously to Programming I in the first semester. Therefore, it is essential to make good use of each one of these subjects in order to successfully overcome all of them.

In order to assure the assimilation of the contents and the acquisition of skills, it is recommended the critical reading of the bibliography, the use of the electronic material available in the UAMx platform (http://uamx.uam.es) and the active search for complementary material in the network.

It is very important the student's disposition to carry out the exercises that the lecturers will propose everyday. The student must check their solution proposed by the lecturers daily in order to gradually improve their programming style.

1.10. Minimum attendance requirement

This subject has two evaluation methods: continuous and not continuous.

It is highly recommended to follow the continuous assessment method. Students who submit to the final test of the ordinary call will be evaluated by the non-continuous evaluation method.

The details of the evaluation methods can be found in section 3 of this guide.

Attendance at theory and practice classes is highly recommended.

1.11. Faculty data

Add @uam.es to all email address below:

Dr. Germán Montoro Manrique (Coordinator)
Department of Computer Engineering
Higher Polytechnic School
Office - Module: B-410 Building B - 4th Floor
Phone: +34 91 497 2210
Email: german.montoro
Website: http://www.eps.uam.es/~montoro

Student attention schedule: Request a prior appointment by email.

1.12. Competences and learning outcomes

Competences

B4 Conocimientos básicos sobre el uso y programación de los ordenadores, sistemas operativos, bases de datos y programas informáticos con aplicación en ingeniería.
C3 Capacidad para comprender la importancia de la negociación, los hábitos de trabajo efectivos, el liderazgo y las habilidades de comunicación en todos los entornos de desarrollo de software.
C4 Capacidad para elaborar el pliego de condiciones técnicas de una instalación informática que cumpla los estándares y normativas vigentes.
C5 Conocimiento, administración y mantenimiento sistemas, servicios y aplicaciones informáticas.
C6 Conocimiento y aplicación de los procedimientos algorítmicos básicos de las tecnologías informáticas para diseñar soluciones a problemas, analizando la idoneidad y complejidad de los algoritmos propuestos.
C7 Conocimiento, diseño y utilización de forma eficiente los tipos y estructuras de datos más adecuados a la resolución de un problema.

Learning outcomes

At the end of each unit the student should be able to:

SPECIFIC OBJECTIVES BY SUBJECT	
UNIT 1.- Introduction	
1.1.	Solve generic programming problems
1.2.	Edit, compile and link a simple program in a programming environment.
1.3.	Write information on the computer screen.
1.4.	Read information from the keyboard and save it in a variable..
1.5.	Debug a program with loops and functions.
UNIT 2.- Types of data and basic operators	
2.1.	Declare variables of types int, float and char
2.2.	Work with characters through its ASCII code representation.
2.3.	Use assignment and arithmetic operators
2.4.	Use parentheses in arithmetic operations.
UNIT 3.	**Conditionals**
3.1	Write and evaluate simple logical operations.
3.2	Distinguish the if-else statement from the switch statement.

3.3	Use enumerations combined with switch instructions.
UNIT 4.- Arrays, strings and structures	
4.1.	Distinguish a macro from a variable
4.2.	Declare arrays of basic types, strings and structures.
4.3.	Access the elements of a one-dimensional and two-dimensional arrays
4.4.	Access the members of a structured variable.
4.5.	Access the members of structure array
4.6.	Access the members of a variable of a nested structured type.
UNIT 5.- Loops	
5.1.	Transform a for loop into an equivalent while loop
5.2.	Use for loops to read and write the elements of an array
5.3.	Use nested loops to navigate two-dimensional arrays.
UNIT 6.- Functions and pointers	
6.1.	Define and call functions with arguments.
6.2.	Declare pointers
6.3.	Assign the address of a variable to a pointer.
6.4.	Access the variable pointed from a pointer.
6.5.	Distinguish an argument passed by value of those passed by reference.
6.6.	Access the characters of a string using a pointer.
6.7.	Dynamic memory allocation for one-dimensional arrays
6.8.	Use strings as arguments and the string manipulation functions defined in string.h
6.9.	Access the structure members by using a pointer to the structure
6.10.	Pass structures by value and by reference.
6.11.	Pass arrays and strings as arguments.
UNIT 7.- Text files	
7.1.	Read a text file character by character and word by word.
7.2.	Read a text file with numeric data.
7.3.	Write a text file character by character.
UNIT 8.- Structure of a program	
8.1.	Create header files
8.2.	Distribute a program among several files.
8.3.	Design a program in a descending manner.

Course objectives

Programming I is an introductory subject to programming. It is intended that students acquire basic knowledge about programming techniques and software design methodology applicable to traditional high-level languages. In particular, the student will learn to program in the C language.

1.13. Course contents

Synthetic Program

- UNIT 1. Reading and writing information
- UNIT 2. Types of data and basic operators
- UNIT 3. Conditional
- UNIT 4. Arrays, strings and structures
- UNIT 5. Loops

- UNIT 6. Functions and pointers
- UNIT 7. Text files
- UNIT 8. Structure of a program

Detailed Program

1. **Reading and writing information**
    1. Editing, compiling and linking a program
    2. Information writing
    3. Reading of information
2. **Data types and basic operators**
    1. Atomic types
    2. The ASCII code
    3. Basic operators
3. **Conditionals**
    1. Relational and logical operators
    2. The if-else statement
    3. The switch statement and the enumerations
4. **Arrays, strings and structures**
    1. Macros
    2. Introduction to arrays
    3. Introduction to strings
    4. Introduction to structures
    5. Nested structures
    6. Arrays of structures
5. **Loops**
    1. The while loop
    2. The do-while loop
    3. The for loop
    4. Nested loops
6. **Functions and pointers**
    1. Functions without arguments
    2. Functions with arguments
    3. Scope of the variables
    4. Pointers
    5. Passing arguments
    6. Pointers and arrays
    7. Passing arrays as arguments
    8. Passing strings as arguments
    9. Passing structures as arguments
    10. Dynamic memory allocation
7. **Text files**
    1. Reading text files
    2. Writing text files
8. **Structure of a program**
    1. Header files
    2. Projects with more than one file
    3. Makefile
    4. Descending design

1.14. Course bibliography

Bibliography:

Below are some textbooks ordered by their difficulty level. It is convenient that the student consult these books in the library before deciding to buy any of them.

Basic level:

1. C Programming: A modern approach (second edition). K. N. King. W. W. Norton and Company, 2008. Excellent C manual, well structured, updated with the C99 standard and accessible not only for beginners in C, but even for novice programmers. It is one of the C texts of reference in many American universities.
2. C. Primer Plus (sixth edition). Stephen Prata. Sams Publishing, 2013. This is a book that explains very clearly the changes experienced by the language from one standard to another, including C99 and C11. Each chapter contains review questions and programming exercises.
3. Head first C. David Griffiths. O'Reilly, 2013. Like other books in the Head First collection, this text of C uses the latest results on learning processes to facilitate the learning of the programming language C. Its reading is very entertaining and sure to surprise the student.
4. A book on C. Programming in C (fourth edition). Kelley, Al and Pohl, Ira. Addison-Wesley, 2004. This text uses the method of dissection, devised by the author in 1984, which is nothing more than a detailed description of each program instruction to instruction. The latest edition contains material to facilitate the student's transit to C ++ and Java.

Advanced level:

1. The C programming language. Brian W. Kernighan, Dennis M. Ritchie. Prentice Hall, 1988. It is without a doubt the most famous C-programming book in history and is written by language designers. His influence has been such that today the first example of almost any language is an example devised by the authors: the hello world program. Another feature that many subsequent texts have incorporated is its chapter 0 that contains a quick introduction to the language in less than 30 pages. It is an indispensable reference for any programmer that is recommended to acquire in its original language.

Electronic material: electronic work documents are published in the section of Programming 1 of the UAMx platform (https://uamx.uam.es)

2. Teaching-and-learning methodologies and student workload

Contact hours

	#horas
Contact hours (minimum 33%)	82
Independent study time	68

List of training activities

Activity	# hours
Lectures	15
Seminars	
Practical sessions	55
Clinical sessions	
Computer lab	
Laboratory	
Work placement	
Supervised study	
Tutorials	6
Assessment activities	6
Other	68

3. Evaluation procedures and weight of components in the final grade

3.1. Regular assessment

There are two evaluation methods in this subject: continuous and not continuous.

The student who attends to the final test will be evaluated by the non-continuous evaluation method.

It is essential that the student carefully read the evaluation regulations of the EPS and the UAM since they will be applied rigorously, specifically to the copies.

Continuous evaluation method

Students will be qualified through 5 practical tests (P1, ..., P5)
The tests are individual.
Each test will be scored between 0 and 10.
The student who does not attend a test will be qualified with a 0 in that test.
The grade will be equal to the weighted average according to the following equation:

$$C = (P1 + 2*P2 + 2*P3 + 2*P4 + 3 * P5) / 10$$

If the student attends less than 3 tests, he will receive the grade "Not evaluated". Otherwise, he will receive a numerical grade corresponding to the average of the tests.

Non-continuous evaluation method.

The final grade of the subject will be that corresponding to a global test.

List of evaluation activities

Evaluatory activity	%
Final exam	0
Continuous assessment	100

3.2. Resit

The qualification of the extraordinary test will be that corresponding to a global test.

List of evaluation activities

Evaluatory activity	%
Final exam	100
Continuous assessment	0

4. Proposed workplan

Week	Content	Contact hours	Independent study hours
1	- Presentation of the subject supported by the course guide. Description of the e-learning platform. **UNIT 1. Reading and writing information** - 1.1 Editing, compiling and linking a program - 1.2 Information output - 1.3 information input	3 - Edit, compile, link and execute a greeting program. - Exercises.	- Read teaching guide. - Register for Moodle and UAMx and register for the subject. - Become familiar with the programming environment. - review UNIT 1 documentation
2	**UNIT 2. Basic data types and operators** - 2.1 Atomic types - 2.2 The ASCII code - 2.3 Basic operators	5 - Exercises - Work in the laboratory.	- review documentation of UNIT 2. - Do exercises proposed and / or compare the solution of the exercises with that provided
3	**UNIT 3. Conditional** - 3.1 Relational and logical operators - 3.2 The if-else statement - 3.3 The switch instruction and the enumerations	5 - Exercises - Work in the laboratory.	- Review UNIT 3 documentation. - Do exercises proposed in class and / or compare the solution of the exercises with that provided by teacher
4	**UNIT 4. Arrays, strings and structures** - 4.1 Macros - 4.2 Introduction to arrays - 4.3 Introduction to strings	5 - Exercises - Work in the laboratory.	- Review UNIT 4 documentation. - Do exercises proposed in class and / or compare the solution of the exercises with that provided by teacher

	- 4.4 Introduction to structures - 4.5 Nested structures - 4.6 Arrays of structures		
5	**ITEM 5. Loops** - 5.1 The while loop - 5.2 The do-while loop	3 - Exercises - Work in the laboratory.	Review UNIT 5 documentation. - Do exercises proposed in class and / or compare the solution of the exercises with that provided by the teacher
6	**UNIT 5. Loops** - 5.3 The for loop - 5.4 Nested loops	5 - Exercises - Work in the laboratory.	Review UNIT 5 documentation. - Do exercises proposed in class and / or compare the solution of the exercises with that provided by the teacher
7	**REVIEW EXERCISES** - Review exercises to review what has been learned so far **PRACTICAL TEST 1.**	5 - Exercises. - Work in the laboratory. - Evaluation test 1	- Do exercises proposed in class and / or compare the solution of the exercises with that proposed by the teacher.
8	**UNIT 6. Functions and pointers** - 6.1 Functions without arguments - 6.2 Functions with arguments - 6.3 Scope of the variables	3 - Exercises - Work in the laboratory.	- Review UNIT 6 documentation. - Do exercises proposed in class and / or compare the solution of the exercises with that provided by the teacher
9	**UNIT 6. Functions and pointers** - 6.4 Pointers - 6.5 Passing arguments by reference - 6.6 Pointers and arrays **PRACTICAL TEST 2**	3 - Exercises. - Work in the laboratory. - Evaluation test 2	- Review UNIT 6 documentation. - Do exercises proposed in class and / or compare the solution of the exercises with that provided by the teacher
10	**UNIT 6. Functions and pointers** - 6.7 Passing arrays as arguments - 6.8 Passing strings as arguments - 6.9 Passing structures as arguments	5 - Exercises. - Work in the laboratory.	- Review UNIT 6 documentation. - Do exercises proposed in class and / or compare the solution of the exercises with that provided by the teacher
11	**UNIT 6. Functions and pointers** - 6.10 Dynamic memory allocation **PRACTICAL TEST 3**	5 - Exercises. - Work in the laboratory. - Evaluation test 3	- Review UNIT 6 documentation. - Do exercises proposed in class and / or compare the solution of the exercises with that provided by the teacher
12	**REVIEW EXERCISES** - Review exercises to review what has been learned so far	5 - Exercises. - Work in the laboratory	- Do exercises proposed in class and / or compare the solution of the exercises with that proposed by the teacher.
13	**UNIT 7. Text files** - 7.1 Reading text files - 7.2 Writing text files **PRACTICAL TEST 4**	3 - Exercises. - Work in the laboratory. - Evaluation test 4	- Review UNIT 7 documentation. - Do exercises proposed in class and / or compare the solution of the exercises with that provided by the teacher
14	**UNIT 8. Structure of a program** - 8.1 Header files - 8.2 Projects with more than one file - 8.3 Makefile - 8.4 Descending design	5 - Exercises. - Work in the laboratory.	- Review UNIT 7 documentation. - Do exercises proposed in class and / or compare the solution of the exercises with that provided by

			the teacher
15	**REVIEW EXERCISES** - Review exercises to review what has been learned so far **PRACTICAL TEST 5**	5 - Evaluation test 5	- Do exercises proposed in class and / or compare the solution of the exercises with the one proposed by the teacher.