Bachelor in Computer Science Engineering

Course information

Year 2020-21

GENERAL SPECIFICATIONS				
English name				
Programming Fundamentals / Introduction to programming				
Spanish name				
Fundamentos de Programación				
Code		**Type**		
606010104		Basic		
Time distribution				
	Total	**In class**	**Out class**	
Working hours	150	60	90	
ECTS:				
Standard group	**Small groups**			
	Classroom	**Lab**	**Practices**	**Computer classroom**
3	0			3
Departments		**Knowledge areas**		
Tecnologías de la Información		Lenguajes y Sistemas Informáticos		
Year		**Semester**		
1º		1º		

TEACHING STAFF			
Name	**E-Mail**	**Telephone**	**Office**
Victoria Pachón Álvarez	vpachon@uhu.es	87373	ETP-119
Roche Beltrán, Francisco	roche@dti.uhu.es	959217654	ETP-126

SPECIFIC INFORMATION OF THE COURSE

1. Contents description

1.1. In English:

Introduction to Programming Languages
- Definition and history of programming languages
- Programming Paradigms
- Compilers and Interpreters.

Introduction to Object Oriented Programming

Algorithms and Data Types.
- Algorithms. Data Types, Operators and Expressions.
- Control Structures.
- Basic Types of Structured Data.
- Methods and interfaces.

Programming Techniques.
- Top-down design.
- Modular Design.

1.2. In Spanish

Introducción a los Lenguajes de Programación
- Concepto e historia de los lenguajes de programación
- Paradigmas de Programación

- Compiladores e Intérpretes.

Introducción a la Programación Orientada a Objetos

- Algoritmos y Tipos de Datos.
- Algoritmos. Tipos de Datos, Operadores y Expresiones.
- Estructuras de Control.
- Tipos Básicos de Datos Estructurados.

Métodos e interfaces.

Técnicas de Diseño de Programas.

- Diseño Descendente.
- Diseño Modular.

2. Background

2.1.Situation within the Degree:

Programming Fundamentals is a first-year, first-semester course that provides an introduction to the fundamental principles in programming with particular focus on the concepts of algorithms and object oriented programming. This course is specially intended for students in the first year of a Computer Science degree.

2.2. Recommendations:

Students should study the subject with the support of suggested literature and attending regularly to tutorials. Due to the practical nature of the subject, it is recommended that students do most of the exercises included in problem sets

3. Objectives (as result of teaching):

Programming Fundamentals has as main objectives:

• Provide a general overview about specification, implementation, verification and documentation of programs.
• Understand the central role of abstraction in the task of programming
• Develop ability to solve problems using algorithm design techniques and apply it to program coding.
• Understand and use basic data structures, algorithms and general-purpose schemes.
• Gain exposure to object-oriented programming.
• Provide the theoretical and practical fundamentals for further studies in programming.

4. Skills to be acquired

4.1. Specific Skills:

CB04: Basic knowledge of the use of computers and programming techniques, operating systems, databases and computer software with application in engineering.

CB05: Knowledge of the structure, organization, operation and interconnection of computer systems, the fundamentals of their programming, and their application for solving engineering problems.

4.2. General Skills:

CB1: Demonstrate to understand and have acquired knowledge about an area of study that starts from basic Secondary Education, and is often at supported by advanced textbooks, but also includes some aspects that involve knowledge related to the forefront of their field of study.

CB5: Develop learning skills necessary to undertake studies later with a high degree of autonomy

CT2: Develop a critical attitude, being able to analyse and synthesize.

CT3: Develop an attitude of inquiry that permanently enables to review and deepen in the knowledge.

CT4: Ability to use the Computer and Information Competences in practice

5. Training Activities and Teaching Methods

5.1. Training Activities:

- Lecture
- Problem Solving Sessions
- Practical sessions in specialized laboratories
- Evaluation activities and self-evaluation and other activities (Essay, debates, tasks delivery, conferences…)

5.2. Teaching Methods:

- Participatory magisterial class.
- Development of practices in specialized laboratories or computer classrooms in small groups.
- Problem solving and practical exercises.
- Evaluations and exams.

5.3. Development and Justification:

In each participatory magisterial class, main concepts of each subject will be explained.
The practices of this subject will consist in the design and implementation of object-oriented programs in C ++. The work will be carried out individually.
It will be compulsory to attend at least 80% of the practical laboratory sessions for these students with continuous evaluation.

6. Detailed Contents:

SECTION 1: INTRODUCTION TO PROGRAMMING LANGUAGES
TOPIC 1: PROGRAMMING LANGUAGES.

1.1. Concept and history of programming languages.
1.2. Programming Paradigms.
1.3. Classification.
1.4. Compilers and Interpreters. Executable code generation process. Compilation and linking.
1.5. Object-oriented programming.

SECTION 2: ALGORITHMS AND DATA TYPES
TOPIC 2: ALGORITHMS. DATA TYPES, OPERATORS AND EXPRESSIONS

2.1. Concept of algorithm. General structure.
2.2. Keywords, Identifiers, constants and comments. Variables and objects.
2.3. Data Types and Classes.
2.4. Input output operations.
2.5. Assignment, arithmetic, relational and logical operators.
2.6. Expressions and order of precedence.

2.7. Generic functions and methods.

TOPIC 3. CONTROL STRUCTURES

3.1. Sentences.
3.1. Sequential sentences.
3.2. Conditional sentences.
3.3. Iterative sentences.
3.4. Macros.

TOPIC 4. TYPES OF STRUCTURED DATA

4.1. Records and Records Structures. Hierarchical records.
4.2. Tables.
4.3. Search and find schemes.
4.4. Strings
4.5. Objects and classes concepts.

SECTION 3: PROGRAM DESIGN TECHNIQUES

TOPIC 5: DESCENDING DESIGN.

5.1. Structured and Modular Programming.
5.2. Structured Programming Concepts.
5.3. Global declarations and local declarations.
5.4. Private and public variables.
5.5. Arguments by value and by reference.
5.6. Constructors and destroyers.
5.7. Passing complex data structures to functions and methods.
5.8. Overloading methods and operators.

COMPUTER SCIENCE LABORATORY:
SECTION 2: ALGORITHM AND DATA TYPES.
Practices 1, 2 and 3. Introduction to C ++, classes, strings and tables.
SECTION 3: PROGRAM DESIGN TECHNIQUES
Practice 4. Top-down design.

7. Bibliography

7.1. Basic Bibliography

- Data structures and algorithms in C++. Goodrich, Michael T.; Tomassia, Roberto; Mount, David M. 2004
- C ++ for Dummies. Davis, Stephen R. 2014
- The C++ programming language , Stroustrup, Bjarne. 2004
- Kernighan B. W., Ritchie D. M.: C Programming Language (2nd Edition), Prentice Hall Software Series
- C programming; a modern approach, 2d ed. Scitech Book News, Jun 2008, Vol.32(2)

-

7.2. Additional Bibliography:

- METODOLOGÍA DE LA PROGRAMACIÓN I: INTRODUCCIÓN AL DISEÑO ORIENTADO A OBJETOS. A. Márquez, Lourdes Ortiz, Mª Pilar Polo, Fco. Roche y Ana Mª Roldán. Servicio de Publicaciones de la Universidad de Huelva.
- COMO PROGRAMAR EN C/C++. H.M. Deitel. Edt. PEARSON Prentice Hall.
- C++ ESTÁNDAR. E. Hernández Orallo. Edt. Paraninfo, Thomson Learning.
- PROGRAMACIÓN EN C++ PARA INGENIEROS. F. Xhafa, P. Vázquez, J. Marco, X. Molinero y A. Martín. Edt. Thomson.
- EL LENGUAJE DE PROGRAMACIÓN C++, B. Stroustrup. Ed. PEARSON Addison Wesley.
- PROGRAMACIÓN Y DISEÑO EN C++, J.P. Cohoon, J.W. Davidson. Edt. Mcgraw-Hill
- RESOLUCIÓN DE PROBLEMAS CON C++. W. Savitch. PEARSON Addison Wesley.

8. Systems and Assessment Criteria

8.1. System for Assessment:

- Examination of theory / problems
- Examen of practice

8.2. Assessment Criteria and Marks:

The evaluation will be continuous, except for those students who request to take the final single evaluation following the procedure provided in this teaching guide.

Continuous Evaluation

- **Examination Sitting I (February)**
  Theory / Problems Exam. It will consist of a written exam in which the student must solve different problems and / or theoretical questions related to the topics of theory developed during the semester.
  Practices (practical exams in computer lab). Throughout the semester there will be 2 partial tests about laboratory practices. The maximum score of this criterion is 50% of the final grade. The first test will have a maximum score of 1.5 points out of 5 and the second of 3.5 points out of 5. A minimum attendance to 80% of practical sessions is required. Final grades will be calculated as follows:
  Final Grade** = 0.5 * (**Theory / Problems Exam)** + 0.5 * **(Practical score)

- **Examination Sitting II (September)**
  Theory / Problems Exam. It will consist of a written exam in which the student must solve different problems and / or theoretical questions related to the topics of theory developed during the semester.
  Practices (practical exams in computer lab). It will consist of a single practical exam in a computer lab, to solve one or more practical exercises. Final grades will be calculated as follows:
  Final Grade** = 0.5 * (**Theory / Problems Exam)** + 0.5 * **(Practical score)
- **Examination Sitting III and Extraordinary**
  Theory / Problems Exam. It will consist of a written exam in which the student must solve different problems and / or theoretical questions related to the topics of theory developed during the semester.
  Practices (practical exams in computer lab). It will consist of a single practical exam in a computer lab, to solve one or more practical exercises. Final grades will be calculated as follows:
  Final Grade** = 0.5 * (**Theory / Problems Exam)** + 0.5 * **(Practical score)

Final Single Assessment

Theory / Problems Exam. It will consist of a written exan in which the student must solve different problems and / or theoretical questions related to the topics of theory developed during the semester.
Practices (practical exams in computer lab). It will consist of a single practical exam in a computer lab, to solve one or more practical exercises. Final grades will be calculated as follows:
Final Grade** = 0.5 * (**Theory / Problems Exam)** + 0.5 * **(Practical score)