COURSE DATA

Data Subject

Code	34672
Name	Programming languages
Cycle	Grade
ECTS Credits	6.0
Academic year	2023 - 2024

Study (s)

Degree	Center	Acad. year	Period
1400 - Degree in Computer Engineering	School of Engineering	3	First term
1407 - Degree in Multimedia Engineering	School of Engineering	4	First term

Subject-matter

Degree	Subject-matter	Character
1400 - Degree in Computer Engineering	11 - Computing and programming	Obligatory
1407 - Degree in Multimedia Engineering	19 - Optatividad	Optional

Coordination

Name	Department
AREVALILLO HERRAEZ, MIGUEL	240 - Computer Science

SUMMARY

"Programming Languages" is a 6 ECTS credit compulsory subject in the Third Year of the Bachelor degree in Computer Science. The subject is part of the module "Programming and Computation"

The content builds on the programming knowledge previously acquired and elaborates on advanced aspects. In particular, the following contents are covered: the compilation process and keys aspects of functional programming.

An integrative approach is adopted. Building on contents learned in the subject "Automata, Formal Languages and Applications", the subject starts introducing the compilation process, by using generation tools to construct lexical analyzers and parsers. In addition, the functional programming paradigm is presented. This paradigm is studied from the perspective of the application of concepts such as immutability and higher-order functions in writing code.

PREVIOUS KNOWLEDGE

Relationship to other subjects of the same degree

There are no specified enrollment restrictions with other subjects of the curriculum.

Other requirements

It is highly recommended that students choosing this module are fluent in java programming and have successfully coursed the subjects Algorithms and Data Structures and Automata, Formal Languages and Applications in the same module; the subjects Computer Organization; and the entire module Computer Science. More precisely, contents on Grammars and forma languages studied in Automata, Formal Languages and Applications are essential to understand concepts related to the compilation process. In addit

OUTCOMES

1400 - Degree in Computer Engineering

- G4 - Ability to define, evaluate and select hardware and software platforms for the development and implementation of computer systems, services and applications, in accordance with both the knowledge and the specific skills acquired in the degree.
- G5 - Ability to design, develop and maintain computer systems, services and applications using software engineering methods as an instrument for quality assurance, in accordance with both the knowledge and the specific skills acquired in the degree.
- G9 - Ability to solve problems with initiative, decision making, autonomy and creativity. Ability to communicate and transmit the knowledge, skills and abilities of a computer engineer.
- R6 - Knowledge and application of basic algorithmic procedures of computer technology to design solutions to problems, by analysing the suitability and complexity of the algorithms proposed.
- R7 - Knowledge, design and efficient use of the types and structures of data most suitable for solving a problem.
- R8 - Ability to analyse, design, build and maintain applications in a robust, secure and efficient manner by choosing the most suitable paradigm and programming languages.

LEARNING OUTCOMES

The subject contributes to the following learning results of the module:

- Reasoning with finite state models and extend then to practical situations in programming and device modeling.
- Use construction tools to build lexical and syntactical analyzers.
- Know different programming paradigms.
- Understand the theoretical foundations of programming languages.
- Use abstraction and recursion to design algorithms and data structures (lists and trees)

In particular, after studying this module, the student should be able to:

- Classify a programming language with a given description under the correct programming paradigm.
- Build a simple compiler using generation tools to produce lexical analyzers and parsers.
- Use key concepts of the functional paradigm

DESCRIPTION OF CONTENTS

1. Introduction

Historical evolution of programming Languages
Main characteristics of programming languajes.
Programming Language classification

2. Language Processors

Types: compilers, interpreters
Sintax: criteria, syntatic elements
Processing:
Source code analysis
Lexical analysis
Syntactic analysis (descending and ascending)
Semantic Analysis
Code Generation
Intermediate code
Optimization
Tools

3. Functional Programming

Why functional programming
What is functional programming
The LISP programming language
Application of functional paradigm concepts

WORKLOAD

ACTIVITY	Hours	% To be attended
Theory classes	30,00	100
Laboratory practices	20,00	100
Classroom practices	10,00	100
Study and independent work	30,00	0
Readings supplementary material	30,00	0
Preparation of evaluation activities	10,00	0
Preparing lectures	10,00	0
Preparation of practical classes and problem	10,00	0
TOTAL	**150,00**	

TEACHING METHODOLOGY

In-class learning activities in this subject include both theoretical and practical ones. Theoretical activities, in the form of lectures, aim at developing an integrative view of the contents. This type of lessons cover the key and most complex aspects of the subject and attempt to foster student participation. The knowledge acquired during the lecture sessions is consolidated by a series of practical activities. These include:

• Problem sessions in the classroom

• Tutorial sessions to discuss independent work

• Laboratories

• In class quizzes

In addition to in class activities, a reasonable amount of independent work is required, to prepare for lectures, search for contents, solve a wide range of problems, complete practical course works and study for the quizzes. The University Learning Management System (LMS) Aula Virtual will be used to support the delivery of this course. This LMS will be used to disseminate materials and to facilitate course work submissions

EVALUATION

At the Universitat de València, the student evaluation should follow the regulations approved in Consell de Govern held on 30th May 2017.

In this module, the evaluation is composed of three tests about the theoretical and practical contents of the subject. These will generally take place during the first half of the term (P1), during the second half (P2) and at the examination period (P3). The content of P1, P2 and P3 will be accumulative and hence the tests will have an increasing value in the final mark, which will be computed as 0.1*P1+0.3*P2+0.6*P3

In the second call, the mark will be the one obtained in a final test that will take place during the corresponding examination period. The marks in P1, P2 and P3 will not be considered in this case.

All tests could contain parts that require a minimum mark in order to pass the module.

In any case, the student should hand in all laboratory work and activities, correctly solved and on time. Otherwise, the mark will be capped at 4.

REFERENCES

Basic

- Lenguajes de programacion. Principios y paradigmas, Allen B. Tucker, Robert Noonan, McGraw-Hill/Interamericana de España, S.A.U., 2003
- Java a tope: Compiladores, Sergio Gálvez Rojas y Miguel Ángel Mora Mata, Universidad de Málaga. Disponible en Web: http://www.giaa.inf.uc3m.es/docencia/II/PL2/herramientas/JFlex_JavaCC.pdf
- Functional Programming for Java Developers, Dean Wampler, OReilly Media, Inc, 2011. ISBN 978-1-4493-1103-2
- Java: How to Program, Ninth Edition, Paul Deitel - Deitel & Associates, Inc.; Harvey Deitel - Deitel & Associates, Inc., Prentice Hall, 2011, ISBN-13: 978-0-13-257566-9
- On LISP, Advanced Techniques for Common LISP, Paul Graham, Prentice Hall, 1993, ISBN 0130305529. Versión gratuita en http://paulgraham.com/onlisp.html