Course guide
270030 - CL - Compilers

Last modified: 30/01/2024

Unit in charge: Barcelona School of Informatics
Teaching unit: 723 - CS - Department of Computer Science.

Degree: BACHELOR'S DEGREE IN INFORMATICS ENGINEERING (Syllabus 2010). (Optional subject).

Academic year: 2023 **ECTS Credits:** 6.0 **Languages:** Catalan

LECTURER

Coordinating lecturer: JOSE MIGUEL RIVERO ALMEIDA

Others: Segon quadrimestre:
LLUIS PADRO CIRERA - 12
JOSE MIGUEL RIVERO ALMEIDA - 11, 12

PRIOR SKILLS

To follow the course on Compilers, it is necessary that students have an adequate knowledge of theory of formal languages □ (regular □□and context-free languages) and automata (finite and stack). It is also necessary to have knowledge of data structures (lists, trees, graphs) and algorithms on these basic structures (traversal, search, recursion). Finally, a basic knowledge of machine language and assembler is also required. For the reasons mentioned above, the student should have completed the course of Algorithmics, Theory of Computation and Computer Structure.

REQUIREMENTS

- Prerequisite TC

DEGREE COMPETENCES TO WHICH THE SUBJECT CONTRIBUTES

Specific:
CCO1.2. To demonstrate knowledge about the theoretical fundamentals of programming languages and the associated lexical, syntactical and semantic processing techniques and be able to apply them to create, design and process languages.
CCO2.3. To develop and evaluate interactive systems and systems that show complex information, and its application to solve person-computer interaction problems.
CT1.2C. To use properly theories, procedures and tools in the professional development of the informatics engineering in all its fields (specification, design, implementation, deployment and products evaluation) demonstrating the comprehension of the adopted compromises in the design decisions.
CT4.1. To identify the most adequate algorithmic solutions to solve medium difficulty problems.

Generical:
G5. TEAMWORK: to be capable to work as a team member, being just one more member or performing management tasks, with the finality of contributing to develop projects in a pragmatic way and with responsibility sense; to assume compromises taking into account the available resources.

TEACHING METHODOLOGY

The theoretical contents of the subject are taught in theory classes. These classes are complemented by practical examples and problems that students have to solve in Autonomous Learning hours. During the theory and sporadically in some laboratory classes some of the most representative exercises of the course will be solved.

In the laboratory sessions, that knowledge that have to support the realization of the project of the subject is imparted. In the first sessions, simple examples will be reviewed in which students will have to make modifications to become familiar with the proposed solutions. After a few introductory sessions, students will focus their efforts on the course project. During the laboratory classes, the teacher will introduce new techniques and will leave an important part of the class for the students to work on their project with the help of the teacher when necessary. In this phase it will be essential to use the Autonomous Learning hours assigned to the project.

The laboratory project will be done in groups of 2 or 3 people, depending on the complexity of the problem. Students are expected to organize their group so that tasks are distributed and synchronized. Each student must have a significant and individual responsibility in the project.

LEARNING OBJECTIVES OF THE SUBJECT

1.Knowing the general structure of compilers and interpreters of programming languages.
2.Learn the techniques of lexical analysis and methods for generating scanners
3.Knowing parsing techniques and methods for generating parsers.
4.Learn the techniques of semantic analysis of programming languages.
5.Knowing the structure of an interpreter and the basic concepts on virtual machines.
6.Learn techniques of compiler code generation.
7.Learn the for code optimization of a compiler.
8.Learn how to organize a working group to perform a task of average complexity.
9.Learn how to design programming languages □□for applications in various areas of IT.
10.Learn how to develop a medium-complexity project for the design and translation/interpretation of a programming language.

STUDY LOAD

Type	Hours	Percentage
Hours small group	30,0	20.00
Guided activities	6,0	4.00
Self study	84,0	56.00
Hours large group	30,0	20.00

Total learning time: 150 h

CONTENTS

Introduction.

Description:
Elements of a programming language. Compilers and interpreters: concept and differences. Structure and phases of a compiler and an interpreter.

Lexical analysis.

Description:
Objectives of lexical analysis. Review of the theory of regular languages and finite automata. Construction of lexical analyzers from indeterministic automata. Lexical analysis tools.

Syntax analysis.

Description:
Goals of parsing. Review of the theory of context-free languages and pushdown automata. Ambiguous grammars and right and left recursion. Construction of top-down parsers. Construction of bottom-up parsers.

Syntax trees.

Description:
Concrete and abstract syntax trees. Construction of syntax trees. Information stored in syntax trees.

Semantic Analysis.

Description:
Names and objects in a programming language. Objects lifetime. Static and dynamic name visibility. Function activation blocks. Symbol tables. Type checking.

Interpreters and virtual machines.

Description:
Interpretation of a programming language. Concept of virtual machine. Types of interpreters. Phases of an interpreter. Internal and intermediate representations: tree, three-address code, stack machine code. Access to local and non-local names. Examples of virtual machines.

Code generation.

Description:
Intermediate code. Code generation for arithmetic and Boolean expressions. Code generation for instructions: assignment, conditional instruction, iterative instruction. Functions: activation, parameter and result return blocks. Access to data structures.

Code optimization.

Description:
Basic blocks and local optimizations. Control flow graph. Data flow analysis for global optimizations. Variable life and use/definition. Global optimizations: common subexpressions, dead code, constant propagation and copy propagation. Loop optimizations: invariants and induction variables.

ACTIVITIES

Lab test

Specific objectives:
3, 7

Full-or-part-time: 5h
Guided activities: 3h
Self study: 2h

Project

Specific objectives:
8, 9, 10

Related competencies :
G5. TEAMWORK: to be capable to work as a team member, being just one more member or performing management tasks, with the finality of contributing to develop projects in a pragmatic way and with responsibility sense; to assume compromises taking into account the available resources.

Full-or-part-time: 9h
Guided activities: 1h
Self study: 8h

Learning general concepts about languages, compilers and interpreters.

Description:
The basic activity is to attend class to acquire knowledge of this topic and to have an overall view of the course. The student will be able to consult specialized books in the case that he or she wishes to expand on some particular aspect.

Specific objectives:
3, 7, 9

Related competencies :
G5. TEAMWORK: to be capable to work as a team member, being just one more member or performing management tasks, with the finality of contributing to develop projects in a pragmatic way and with responsibility sense; to assume compromises taking into account the available resources.

Full-or-part-time: 4h
Theory classes: 2h
Self study: 2h

Theory test

Specific objectives:
3, 7, 9

Related competencies :
G5. TEAMWORK: to be capable to work as a team member, being just one more member or performing management tasks, with the finality of contributing to develop projects in a pragmatic way and with responsibility sense; to assume compromises taking into account the available resources.

Full-or-part-time: 9h
Guided activities: 3h
Self study: 6h

Learning about the theory and techniques of lexical analysis.

Description:
The student will attend the theoretical classes and devote time to studying the theory of lexical analysis and performing exercises proposed by the professor. Some of the knowledge acquired will have to be applied to the subject project.

Specific objectives:
2

Full-or-part-time: 8h
Theory classes: 3h
Self study: 5h

Learning about the theory and design techniques of descending syntactic analyzers.

Description:

The student will attend the class to acquire the theoretical knowledge. In addition, you will have to consolidate these concepts with your personal study and with the resolution of the proposed problems in class. Some of the knowledge acquired will be used in the project.

Specific objectives:
3

Full-or-part-time: 9h
Theory classes: 3h
Self study: 6h

Learning about the theory and design techniques of ascending syntactic analyzers.

Description:
The student will attend the class to acquire the theoretical knowledge. In addition, you will have to consolidate these concepts with your personal study and with the resolution of the proposed problems in class.

Specific objectives:
3

Full-or-part-time: 5h
Theory classes: 2h
Self study: 3h

Learning about the construction of abstract syntax trees.

Description:
The student will attend the class to acquire the theoretical knowledge. In addition, it will have to consolidate these concepts with its personal study and the resolution of proposed problems in class.

Specific objectives:
3, 4

Full-or-part-time: 5h
Theory classes: 2h
Self study: 3h

Learning from semantic analysis theory and techniques

Description:
The student will attend the class to acquire the theoretical knowledge. In addition, it will have to consolidate these concepts with its personal study and the resolution of proposed class problems.

Specific objectives:
4

Full-or-part-time: 7h
Theory classes: 4h
Self study: 3h

Learning about the design of interpreters and virtual machines

Description:
The student will attend the class to acquire the knowledge on the subject. In addition, it will have to consolidate these concepts with its personal study and the resolution of proposed problems in class. The knowledge acquired in the class will later be used in the course project.

Specific objectives:
5

Full-or-part-time: 6h
Theory classes: 2h
Self study: 4h

Learning about code generation theory and techniques

Description:
The student will attend the class to acquire the knowledge on the subject. In addition, it will have to consolidate these concepts with its personal study and the resolution of proposed class problems.

Specific objectives:
6

Full-or-part-time: 10h
Theory classes: 4h
Self study: 6h

Learning about code optimization theory and techniques

Description:
The student will attend the class to acquire the knowledge of the subject. In addition, it will have to consolidate these concepts with its personal study and the resolution of proposed problems in class.

Specific objectives:
7

Full-or-part-time: 10h
Theory classes: 4h
Self study: 6h

Introduction to the ANTLR tool: Design of an Expression Evaluator.

Description:
The student will make a modification of the example proposed by the teacher in the laboratory class. During class, it will use the ANTLR tool, which will be the same as the course project.

Specific objectives:
3, 9

Related competencies :
G5. TEAMWORK: to be capable to work as a team member, being just one more member or performing management tasks, with the finality of contributing to develop projects in a pragmatic way and with responsibility sense; to assume compromises taking into account the available resources.

Full-or-part-time: 3h
Laboratory classes: 2h
Self study: 1h

Extending an interpreter from a programming language.

Description:
The student will have to perform the modification of the interpreter proposed by the teacher during laboratory and personal study hours. Finally, it will have to build a test game that validates the correctness of the modifications made.

Specific objectives:
1, 2, 3, 5, 9, 10

Related competencies :
G5. TEAMWORK: to be capable to work as a team member, being just one more member or performing management tasks, with the finality of contributing to develop projects in a pragmatic way and with responsibility sense; to assume compromises taking into account the available resources.

Full-or-part-time: 7h
Laboratory classes: 4h
Self study: 3h

Design of a programming language and its parser.

Description:
The activity will consist of designing a simple programming language aimed at a specific type of application. This activity will allow the student himself to propose the language and scope. Otherwise, the student may choose to perform the project proposed by the subject teacher. Some examples of programming languages that could be proposed: (1) a language for creating and displaying simple object animations, (2) a programming language for controlling a Lego-Robot, (3) a language for generating Escher patterns, (4) a language for generating music scores, (5) a language for defining and manipulating mathematical functions, etc. Once the language is chosen, the student will have to perform the lexical and syntactic analyzer using the course tools.

Specific objectives:
3, 9

Related competencies :
G5. TEAMWORK: to be capable to work as a team member, being just one more member or performing management tasks, with the finality of contributing to develop projects in a pragmatic way and with responsibility sense; to assume compromises taking into account the available resources.

Full-or-part-time: 20h
Laboratory classes: 4h
Self study: 16h

Design of the interpretation phase of a programming language.

Description:
The student will have to build the semantic and language interpreter using course tools during laboratory classes and personal working hours. Finally, you will need to build test games and user-level documentation so that the project can be evaluated.

Specific objectives:
9, 10

Related competencies :
G5. TEAMWORK: to be capable to work as a team member, being just one more member or performing management tasks, with the finality of contributing to develop projects in a pragmatic way and with responsibility sense; to assume compromises taking into account the available resources.

Full-or-part-time: 33h
Laboratory classes: 17h
Self study: 16h

GRADING SYSTEM

The method aims to evaluate both theoretical and practical knowledge of the student.

The evaluation will consist of three evaluative acts. The first of them (L1) aims to assess practical knowledge of the course and will consist of a laboratory test where the student will have to make small modifications to an existing compiler to add new functionality. This test will evaluate the student's knowledge of the structure of a compiler and their ability to implement new programming language concepts and express them in a grammar. The evaluation will be carried out by validating the implementation with previously prepared test sets.

The second evaluative act (L2) will be based on the development of the semantic analysis and code generation of the compiler proposed in the course. This work will be done in group. The evaluative act will consist of a laboratory test where it will be necessary to execute some sets of tests to validate the compiler and the proposed extensions to the same test.This act will evaluate the student's ability to apply the concepts acquired during the course and the knowledge of the work done in group.

Finally, the third act will consist of a written test where the theoretical knowledge imparted during the course will be evaluated.

The final grade of the course (F) will be calculated with a weighted sum of the two laboratory tests (L1 and L2) and the theoretical knowledge test (T):

BIBLIOGRAPHY

Basic:
- Aho, A.V.; Lam, M.S.; Sethi, R.; Ullman, J.D. Compilers: principles, techniques, and tools. 2nd ed. Addison-Wesley, 2007. ISBN 9780321491695.
- Parr, T. The definitive ANTLR 4 reference. The Pragmatic Programmers, 2012. ISBN 9781934356999.
- Cortadella, J. Compilers: lecture notes. 2015.

Complementary:
- Appel, A.W.; Ginsburg, M. Modern compiler implementation in C. Cambridge University Press, 2004. ISBN 0521607655.
- Appel, A.W.; Palsberg, J. Modern compiler implementation in Java. 2nd ed. Cambridge University Press, 2002. ISBN 052182060X.
- Arnold, K.; Gosling, J.; Holmes, D. The Java programming language. 4th ed. Addison-Wesley, 2006. ISBN 0321349806.