DEGREE CURRICULUM

LANGUAGE PROCESSING ALGORITHMS

Coordination: ALSINET BERNADO, MARIA TERESA

Academic year 2022-23

LANGUAGE PROCESSING ALGORITHMS 2022-23

Subject's general information

<table>
<tr><td>Subject name</td><td>LANGUAGE PROCESSING ALGORITHMS</td></tr>
<tr><td>Code</td><td>102043</td></tr>
<tr><td>Semester</td><td>2nd Q(SEMESTER) CONTINUED EVALUATION</td></tr>
<tr><td>Typology</td><td>
<table>
<tr><th>Degree</th><th>Course</th><th>Character</th><th>Modality</th></tr>
<tr><td>Bachelor's Degree in Computer Engineering</td><td>4</td><td>COMPULSORY</td><td>Attendance-based</td></tr>
<tr><td>Bachelor's Degree in Computer Engineering</td><td>4</td><td>OPTIONAL</td><td>Attendance-based</td></tr>
</table>
</td></tr>
<tr><td>Course number of credits (ECTS)</td><td>9</td></tr>
<tr><td>Type of activity, credits, and groups</td><td>
<table>
<tr><th>Activity type</th><td>PRALAB</td><td>TEORIA</td></tr>
<tr><th>Number of credits</th><td>3.6</td><td>5.4</td></tr>
<tr><th>Number of groups</th><td>1</td><td>1</td></tr>
</table>
</td></tr>
<tr><td>Coordination</td><td>ALSINET BERNADO, MARIA TERESA</td></tr>
<tr><td>Department</td><td>COMPUTER SCIENCE AND INDUSTRIAL ENGINEERING</td></tr>
<tr><td>Teaching load distribution between lectures and independent student work</td><td>70 hours lectures / 115 hours independent student work</td></tr>
<tr><td>Important information on data processing</td><td>Consult this link for more information.</td></tr>
<tr><td>Language</td><td>Catalan</td></tr>
<tr><td>Distribution of credits</td><td>The classes of the subject are structured in 3 weekly hours aimed at solving practical problems in the laboratory and 3 weekly hours of a more expository nature where the algorithms, techniques and translation tools of each stage of the translation process will be presented.</td></tr>
</table>

LANGUAGE PROCESSING ALGORITHMS 2022-23

Teaching staff	E-mail addresses	Credits taught by teacher	Office and hour of attention
ALSINET BERNADO, MARIA TERESA	teresa.alsinet@udl.cat	9	

Subject's extra information

The following course is highly recommended:

- Computational Models and Complexity

Learning objectives

The learning objectives of the course are:

- Learning the levels, techniques, and algorithms involved in the translation process of programming languages,
- Using the tools to support the design and implementation of each level.
- Analyzing the main features and implementation techniques associated with non-imperative languages as logical, functional, scripting, object-oriented and distributed and concurrent languages.

Competences

Strategic competences of the University of Lleida

- **CT3.** Training Experience in the use of the new technologies and the information and communication technologies.
- **CT2.** Mastering a foreign language, especially English.

Degree competences

- **EPS6.** Ability of analysis and synthesis.

Specific Degree competences

- **GII-C2**. Ability to know the theoretical basics of the programming languages and the techniques of lexical, syntactic and associated semantic processing, and know how to apply them for the creation, design and processing of languages.

Subject contents

Structure of the course topics:

1. Introduction to programming languages and translation techniques

2- Lexical analysis

3- Tool: Flex (The Fast Lexical Analyzer)

4 - Syntactic analysis: top-down and bottom-up parsers

5- Tool: Yacc (Yet Another Compiler Compiler)

6- Syntax-directed translation

7- Symbol Tables

8- Tool: SymTab

9- Type checking

10- Intermediate representation and code generation

11- Runtime Memory Management

12- Code optimization

13. Generating object code

Methodology

The classes of the subject are structured in 3 weekly hours aimed at solving practical problems in the laboratory and 3 weekly hours of a more expository nature where the algorithms, techniques and translation tools of each stage of the translation process will be presented. .

Students will solve practical exercises during the laboratory sessions and will approach in a group the preparation and presentation of three works:

- Use of regular expressions in programming languages
- Aspects of design and implementation of a particular programming language
- Translator synthesis phase: Memory management in the execution environment and Code optimization

Development plan

The syllabus of the course is divided into two parts.

The **first part** deals with the specification and recognition of lexical components of programming languages, the techniques of parsing routines and how to integrate semantic parsing algorithms. The student's training is complemented by the study of specialized tools supporting the design and implementation of specific components or translation systems. The following tools are introduced: JFLAP for specification and recognition of languages, flex for generating lexical analyzers, yacc for generating bottom-up parsers, and SymTab for symbol tables.

The **second part** of the course deals with the levels of semantic analysis, code optimization and object code generation. We show how to incorporate the process of semantic analysis routines that enable scope management, type checking, intermediate code generation for major constructions of imperative languages and memory allocation. Code optimizations dependent of the intermediate representation and machine object code are studied.

Finally, students choose a topic corresponding to the 2nd part of the subject and present it to the rest of the group, in addition, they choose a programming language and present the main design and implementation characteristics.

Week	Description	Classroom Activity Big Group	Classroom/independent work
1	Lecture and problems	Lesson 1,2	6h/9h

Week	Description	Classroom Activity Big Group	Classroom/independent work
2	Lecture and problems	Lesson 2,3	6h/9h
3	Development of activities	Team work	6h/9h
4	Practices	Presentation of practices	6h/9h
5	Lecture and problems	Lesson 4,5	6h/9h
6	Lecture and problems	Lesson 4,5	6h/9h
7	Development of activities	Team work	6h/9h
8	Practices	Presentation of practices	6h/9h
9		First mid-term exam	
10	Lecture and problems	Lesson 6	6h/9h
11	Lecture and problems	Lesson 7,8	6h/9h
12	Lecture and problems	Lesson 9,10	6h/9h
13	Lecture and problems	Lesson 11,12, 13	6h/9h
14	Development of activities	Team work	6h/9h
15	Practices	Presentation of practices	6h/9h
16	Written tests	Second mid-term exam	
17	Written tests	Second mid-term exam	
18		Study week	
19	Written tests	Recovery exam	

Evaluation

Acronym	Evaluation activities	Weighting	Min. Score	Group work	Compulsory	Recuperable
PR1	Practice 1: Lex	20%	-	Yes	Yes	Yes
PR2	Activity 1: Regular expressions	10%	-	Yes	Yes	No
PR3	Practice 2: Yacc	20%	-	Yes	Yes	Yes
PR5	Activity 2: Design and implementation of programming languages	10%	-	Yes	Yes	No
PR4	Practice 3: Project	30%	-	Yes	Yes	Yes
PR5	Activity 3: Semantics and Synthesis stage	10%	-	Yes	Yes	No

The course activities consist of:

- Lex: use of the generation tool lex (lexical analyser).

- Regular expressions: Regular expressions in programming languages. Oral presentation to the rest of the group.
- YACC: use of the generation tool yacc (parser).
- Design and implementation of programming language: Oral presentation to the rest of the group.
- Project: Use the tool SymTab to implement symbol tables and integration with tools lex and yacc. It is proposed to develop a translator intermediate code (three-addresses) for a small imperative language.
- Synthesis stage: Analysis of the characteristics of some synthesis stage. Implementation issues. Oral presentation to the rest of the group.

- Practical exercises will be carried out individually or in groups of two or three people. In the case of group work, each person will evaluate the participation in the activity of the rest of the group components.

- Activities will be carried out in groups of two or three people. The evaluation of the activities will be collaborative among all those attending the presentations. In addition, in each group, each person will evaluate the participation in the activity of the rest of the group components.

Bibliography

References

[1] A.V. Aho, M. Lam, R. Sethi, and J.D. Ullman. Compilers: Principles, Techniques, and Tools. Addison-Wesley Series in Computer Science, Reading, Massachusetts. Second Edition. 2006.

[2] Dick Grune, Henri E. Bal, Ceriel J. H. Jacobs, Koen G. Langendoen. Modern Compiler Design. Jonh Wiley and Sons, England, 2000.

[3] Andrew W. Appel, Maia Ginsburg. Modern Compiler Implementation in C. Cambridge University Press, 1998.

[4] John Levine. Flex & bison: Text Processing Tools. O'Reilly, 2009.

[5] Reinhard Wilhelm, Helmut Seidl, Sebastian Hack: Compiler Design - Syntactic and Semantic Analysis. Springer 2013.

[6] Helmut Seidl, Reinhard Wilhelm, Sebastian Hack: Compiler Design - Analysis and Transformation. Springer 2012.

[7] Reinhard Wilhelm, Helmut Seidl: Compiler Design - Virtual Machines. Springer 2010

Tools:

- Flex: http://flex.sourceforge.net/
- Yacc: http://www.gnu.org/software/bison/

- JFLAP: http://www.jflap.org/jflaptmp/

- JFlex: http://jflex.de/

- Cup: http://www2.cs.tum.edu/projects/cup/

- Ant: http://ant.apache.org/

- ANTLR**:** http://www.antlr.org/

- https://pypi.org/project/ply/

- https://www.haskell.org/alex/

- https://www.haskell.org/happy/