DEGREE CURRICULUM

LANGUAGE PROCESSING ALGORITHMS

Coordination: PLANES CID, JORDI

Academic year 2016-17

LANGUAGE PROCESSING ALGORITHMS 2016-17

Subject's general information

Subject name	LANGUAGE PROCESSING ALGORITHMS
Code	102043
Semester	2nd Q(SEMESTER) CONTINUED EVALUATION
Typology	<table><tr><th>Degree</th><th>Course</th><th>Typology</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></table>
ECTS credits	9
Groups	1GG
Theoretical credits	2
Practical credits	7
Coordination	PLANES CID, JORDI
Department	INFORMATICA I ENGINYERIA INDUSTRIAL
Teaching load distribution between lectures and independent student work	70 hours lectures / 115 hours independent student work
Important information on data processing	Consult this link for more information.
Language	Catalan
Office and hour of attention	To set an appointment for tutoring email the teacher.

LANGUAGE PROCESSING ALGORITHMS 2016-17

Teaching staff	E-mail addresses	Credits taught by teacher	Office and hour of attention
ALSINET BERNADÓ, MARIA TERESA	tracy@diei.udl.cat	2	
PLANES CID, JORDI	jplanes@diei.udl.cat	7	

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.

LANGUAGE PROCESSING ALGORITHMS 2016-17

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 lectures of the course are 4 hours per week of problem solving and laboratory practice, and two hours a week of lecturing which will present more algorithms, techniques and tools for translating each level of the translation process.

Students solve exercises during the laboratory sessions where they will address the presentation of a non-classic programming language to the other groups, and implement a compiler.

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, in addition to the traditional imperative languages, we analyze the main features and implementation techniques associated with other languages such as logical, functional, scripting, object-oriented and distributed and concurrent languages. For their study and analysis, students choose a language which will be presented to the rest of the group.

LANGUAGE PROCESSING ALGORITHMS 2016-17

Week	Description	Classroom Activity Big Group	Classroom/independent work
1	Lecture and problems	Lesson 1	6h/9h
2	Lecture and problems	Lesson 2,3	6h/9h
3	Lecture and problems	Lesson 2,3	6h/9h
4	Lecture and problems	Lesson 2,3	6h/9h
5	Lecture and problems	Lesson 4,5	6h/9h
6	Lecture and problems	Lesson 4,5	6h/9h
7	Lecture and problems	Lesson 4,5	6h/9h
8	Lecture and problems	Lesson 4,5	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	6h/9h
14	Lecture and problems	Lesson 13	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	Compusory	Recuperable
PR1	JFLAP	10%	-	No	Yes	Yes
P1	Lex	10%	-	No	Yes	Yes
P2	Yacc	15%	-	No	Yes	Yes
P3	Projecte	45%	-	Yes	Yes	Yes
A1	Language analysis	15%	-	Yes	Yes	Yes
A2	Regular expressions	5%	-	No	Yes	Yes

The course activities consist of:

- JFLAP: Resolution of a collection of problems of recognition and specification languages ??using the tool

JFLAP
- Lex: use of automated tools lex lexical analyzer.
- YACC use automatic generation tool yacc parsers.
- Project: Use the tool 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.
- Language Analysis: Analysis of the characteristics of a programming language is not classical. Implementation issues. Work to be done in groups of two or three people. Oral presentation to the rest of the group.
- Regular Expressions: Regular Expression Activity: Practice exercises performed in a first programming language.

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/