DEGREE CURRICULUM

LANGUAGE PROCESSING ALGORITHMS

Coordination: ALSINET BERNADO, MARIA TERESA

Academic year 2023-24

LANGUAGE PROCESSING ALGORITHMS 2023-24

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 ENGINEERING AND DIGITAL DESIGN</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 2023-24

Teaching staff	E-mail addresses	Credits taught by teacher	Office and hour of attention
ALSINET BERNADO, MARIA TERESA	teresa.alsinet@udl.cat	9	EPS Office 2.13 Send an e-mail to the teacher to establish the hour of attention convenient for the student.

Subject's extra information

The following course is highly recommended:

- Algorithms and Complexity
- Languages, automata and grammars
- Computational Models and Complexity

Learning objectives

The student's learning outcomes in the subject are:

- To know, design and implement the stages involved in the translation process of programming languages,
- To formally specify the characteristics of programming languages.
- To apply the techniques and algorithms of the translation process to the implementation of formal languages.
- To apply automatic generation tools to the design and implementation of compilers.
- To know and analyze the main characteristics and implementation techniques associated with non-imperative languages such as logical, functional, scripting, object-oriented, and distributed and concurrent languages.
- To know the evolution of programming languages and the corresponding interpreters, translators and compilers.

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 2023-24

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.

In addition, the student will address the study and application of regular expressions to programming languages and will choose a programming language and analyze the main design and implementation features of the language.

The learning outcomes will be presented orally to the rest of the group.

Week	Description	Classroom Activity Big Group	Classroom/independent work
1	Lecture and problems	Lesson 1, 2	6h/9h
2	Lecture and problems	Lesson 2, 3	6h/9h
3	Development of activity1 and practice 1	Team work	6h/9h
4	Activity 1	Oral presentations	6h/9h
5	Lecture and problems	Lesson 4	6h/9h
6	Lecture and problems	Lesson 5	6h/9h
7	Development of activity 2 and practice 2	Team work	6h/9h
8	Activity 2	Oral presentations	6h/9h
9		**Exams week**	
10	Lecture and problems	Lesson 6, 7	6h/9h
11	Lecture and problems	Lesson 8, 9	6h/9h
12	Development of practice 3	Team work	6h/9h
13	Lecture and problems	Lesson 10, 11	6h/9h
14	Lecture and problems	Lesson 12, 13	6h/9h
15	Development of practice 4	Team work	6h/9h

Evaluation

The continuous evaluation of the subject is based on 5 blocks:

1. **Activities block: 20%**. It consists of two activities: **Activity 1 (10%)** and **Activity 2 (10%)**. They cannot be improved. A minimum grade is not required. Presentation date of Activity 1: During the 4th week. Presentation date of Activity 2: During the 8th week.
2. **Lexical analysis block: 20%**. It consists of a practice: **Practice 1**. It can be Improved. It has no minimum grade. Delivery date of Practice 1: During the 6th week. The date for improving Practice 1 is the date defined by the EPS for the realization of the 1st exam on the subject.
3. **Parsing block: 20%**. It consists of a practice: **Practice 2**. It can be Improved. It has no minimum grade. Delivery date of Practice 2: During the 8th week. The date for improving Practice 2 is the date defined by the EPS for the realization of the 2nd exam on the subject.
4. **Semantic analysis block: 20%**. It consists of a practice: **Practice 3**. It can be Improved. It has no minimum grade. Delivery date of Practice 3: During the 14th week. The date for improving Practice 3 is the date defined by the EPS for the realization of the improvement exam on the subject.
5. **Code generation block: 20%**. It consists of a practice: **Practice 4**. It can be Improved. It has no minimum grade. Delivery date of Practice 4: the date defined by the EPS for the realization of the 2nd exam on the subject. The date for improving Practice 4 is the date defined by the EPS for the realization of the improvement exam on the subject.

Blocks Improvement: It consists of a new date to deliver the corresponding practice. The improvement of the blocks does not condition the maximum grade achieved in the subject.

LANGUAGE PROCESSING ALGORITHMS 2023-24

Evaluation activities

Acronym	Evaluation activities	Weighting	Minimum grade required	Team work	Compulsory	Improvement
PRE1	Activity 1	10%	NO	Yes (<=3)	No	No
PRE2	Activity 2	10%	NO	Yes (<=3)	No	No
PRA1	Practice 1	20%	NO	Yes (<=3)	No	Yes
PRA2	Practice 2	20%	NO	Yes (<=3)	No	Yes
PRA3	Practice 3	20%	NO	Yes (<=3)	No	Yes
PRA4	Practice 4	20%	NO	Yes (<=3)	No	Yes
To pass the subject, the final grade must be >= 5						
Final grade = 0,1*PRE1 + 0,1*PRE2 + 0,2*PRA1 + 0,2*PRA2 + 0,2*PRA3 + 0,2*PRA4						

The course activities consist of:

- **Practice 1**: The student must make use of the automatic generation tool of lexical analyzers lex.
- **Practice 2**: The student must make use of the automatic generation tool for yacc parsers.
- **Practice 3**: The student must make use of the symbol table implementation tool and its integration with the lex and yacc tools.
- **Practice 4**: The student must implement a translator to intermediate code (3-directions) for a reduced imperative language.
- **Activity 1**: Regular expressions. Study and application of regular expressions in programming languages, lexical analyzer generation tools, implementation of the lexical analysis phase. Oral presentation to the rest of the group.
- **Activity 2**: Study of design aspects of programming languages. Oral presentation to the rest of the group.
- The **practices** will be carried out individually or in a group of two or three people. In case of being carried out in a group, each student will self-evaluate her participation in the development of the practice.
- The **activities** will be developed in groups of two or three people. The evaluation of the activities will be collaborative among all those attending the oral presentations. In addition, in each group, each student will self-evaluate her participation in the development of the activity.

Alternative assessment (students who waive continuous assessment):

- **Practice 1: 25%**. Individual. It can be improved. No minimum grade is required. Practice 1 delivery date: Date of the 1st partial exam defined by the EPS. The date for improving Practice 1: Date of the 2nd partial exam defined by the EPS.

- **Practice 2: 25%.** Individual. It can be improved. No minimum grade is required. Practice 2 delivery date: Date of the 1st partial exam defined by the EPS. The date for improving Practice 2: Date of the 2nd partial exam defined by the EPS.

- **Practice 3: 25%**. Individual. It can be improved. No minimum grade is required. Practice 3 delivery date: Date of the 2nd partial exam defined by the EPS. The date for improving Practice 3 is the date defined by the EPS for the improvement exam of the 2nd partial on the subject.
- **Practice 4: 25%**. Individual. It can be improved. No minimum grade is required. Practice 4 delivery date: Date of the 2nd partial exam defined by the EPS. The date for improving Practice 4 is the date defined by the EPS for the improvement exam of the 2nd partial on the subject.
- **Practice Improvement**: It consists of a new date to deliver the corresponding practice. The presentation of the practice on the improvement date does not condition the maximum grade achived in the subject.

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/