UNIVERSITÀ DEGLI STUDI DI PALERMO

DEPARTMENT	Matematica e Informatica
ACADEMIC YEAR	2024/2025
MASTER'S DEGREE (MSC)	DATA, ALGORITHMS AND MACHINE INTELLIGENCE
SUBJECT	LANGUAGES AND COMPILERS DESIGN
TYPE OF EDUCATIONAL ACTIVITY	B
AMBIT	50341-Discipline Informatiche
CODE	24091
SCIENTIFIC SECTOR(S)	INF/01
HEAD PROFESSOR(S)	MANTACI SABRINA Professore Associato Univ. di PALERMO
OTHER PROFESSOR(S)	
CREDITS	6
INDIVIDUAL STUDY (Hrs)	102
COURSE ACTIVITY (Hrs)	48
PROPAEDEUTICAL SUBJECTS	
MUTUALIZATION	
YEAR	1
TERM (SEMESTER)	2° semester
ATTENDANCE	Not mandatory
EVALUATION	Out of 30
TEACHER OFFICE HOURS	**MANTACI SABRINA** Thursday 15:00 18:00 Room 217 - second Floor. DMI - Via Archirafi 34

DOCENTE: Prof.ssa SABRINA MANTACI

PREREQUISITES	In order to understand the content and to achieve the learning goals, it is necessary for the student to be able to code in a programming language (especially C language), to know elements of theoretical computer science (languages, automata, regular expressions and Grammars) and algorithms (lists, piles, trees, graphs, hash tables).
LEARNING OUTCOMES	Knowledge and understanding skills The course intends to provide students with the notions needed to understand and address the various issues related to the compilation phases with special attention to the lexical, syntactic, and semantic analysis that they also apply in others contexts (language translations, parsers, scanners). The course also aims to convey the knowledge of important automatic generation tools parser and scanner. Ability to apply knowledge and understanding The course focuses on the understanding of computational models and techniques that manage the operation of lexical and syntactic analyzers in a modern compiler, in order to apply such methodologies on the one hand to automatic generation or manual of such analyzers, and on the other to the transformation and analysis of the texts guided by the syntax. Students will have the opportunity to apply the knowledge acquired also through an elaborate made in group using the automatic generation tools for parsers and scanners. Judgment autonomy Students will be guided to learn critically and responsibly topics that will be presented to them in the classroom and in the laboratory. Each student will also have occasion to enrich its autonomy of judgment by means of the realization of a project or an elaborate that will be an integral part of the evaluation test. Communication skills Through the laboratory activities planned, the course will tend to develop students' interaction and the ability to know how to work together, to confront one another on the issues in order to find solutions based on knowledge acquired during the course. The acquisition of communication skills will be realized through the active participation of the student to the activities of laboratory as well as by showing the results of individual or group work on topics or issues proposed by the teacher. Learning ability The material offered in classroom and in the laboratory will develop the learning ability of students who will be able to "question" in integrated way their knowledge-skills in dealing with issues addressed. Students will also be stimulated to learn a more in-depth and critical knowledge of the programming languages they already know, through the study on how the process of compiling such languages takes place. Acquired competencies: The student will acquire the skill of autonomously build a lexical analyzer and a syntactic analyzer, both for compilation purposes and for other purposes, for example related to the automatic analysis of a text.
ASSESSMENT METHODS	A midterm test is scheduled in order to evaluate the acquisition of the knowledge on lexical analysis and the syntactic analysis. In particular, the test provides four open questions which aim to verify the ability of constructing a parser for a given language and parse a string. The above test is evaluated in the scale 1-30 (according to the table described below) and it aims to verify the knowledge acquired, the elaborative skills and the correct use of technical language. The maximum score is obtained if full possession of knowledge is shown about the algorithms and techniques, a strong ability to process and apply acquired knowledge to solve the proposed problems and an excellent command of language. Those who do not pass the midterm test, will have to take a similar written test, which will be scheduled before the final exam is held. At the end of the course students (preferably in groups) are assigned a study project where it is required to design and implement a lexical-semantic with the help of Flex and Bison programs. This test aims to verify the ability to know how to apply the acquired knowledge, that is, the ability to define a lexical analyzer, a syntax analyzer and semantic actions for solving a specific problem. This test also aims to verify the student’s interaction ability and the teamwork. At the end of the project, students will have to describe their solution proving their exhibition ability. Such activity is evaluated over scale 1- 30 by referring to the evaluation table described below. The average of the evaluations of the two tests becomes the starting vote for the oral test, which consists of up to three questions that aim to verify the acquisition of the topics covered during the class.

	The final evaluation is expressed in base 1-30 and takes into account the evaluation table described below. Evaluation table: Rating: Excellent Rating: 30 - 30 cum laude Outcome: Excellent knowledge of the topics, excellent command of language, good analytical capacity, effective ability of interaction in the teamwork, the student is able to apply knowledge to solve the proposed problems. Rating: Very good Rating: 27-29 Outcome: Good mastery of the arguments, full command of language, good ability to interact and work in groups, the student is able to apply knowledge to solve the proposed problems. Rating: Good Rating: 24-26 Outcome: Basic knowledge of the main topics, discrete command of language, discreet ability to work in groups, with limited ability to independently apply knowledge to the solution of the proposed problems. Rating: Satisfactory Rating: 21-23 Outcome: He/she does not fully master the main topics of the teaching but possesses the knowledge, satisfactory command of language, poor ability to know how to apply knowledge in independent fashion. Rating: Sufficient Rating: 18-20 Outcome: Minimum basic knowledge of the main topics of teaching and technical language, scarce or no ability to know how to apply knowledge autonomously acquired. Rating: Poor Outcome: He/she does not possess any knowledge on the main topics of the course Compensatory tools and dispensatory measures will be guaranteed by the Disability and Neurodiversity Center - University of Palermo (Ce.N.Dis.) to students with disabilities and neurodiversity, based on specific needs and in implementation of current legislation.
EDUCATIONAL OBJECTIVES	This teaching has two goals. The first one is cultural. We want to introduce a theory that is the basis for compilation of programming languages, which is probably the most important scientific contribution to the development and dissemination of computer technology. The second goal is more applicative. At the end of the course the student will be able to independently build a lexical analyzer and a syntactic analyzer, and carry out the relative semantic actions, making him able to solve problems of this nature when working for a company. Even if the student will never be involved in his/her work career in the design of a new compiler, it is very likely that he/she will often use methodologies and techniques that belong to that theory: from automata and Grammars used as formalisms to define the behavior of a system, the techniques to make translators that are simpler than a real compiler (for example, the interpreter of a configuration file). The general structure of a compiler will be studied, by giving a particular emphasis on the aspects of lexical analysis and parsing. Top-down parser (recursive descent, predictive and LL(1)) and bottom-up (SLR, LR and LALR) type parser will be studied. Finally, examples of translators for simple languages will be examined.
TEACHING METHODS	Teaching is organized in classroom and laboratory lectures, where it is expected to have a transfer of knowledge but also, with the support of computer, the contextual verification of the acquisition of the practical skills required. In order to increase students' engagement collaborative learning methodologies are used.
SUGGESTED BIBLIOGRAPHY	Testo principale (main textbook) A. Aho, M. Lam, R. Sethi, J. Ullman COMPILATORI. Principi, Tecniche e strumenti Addison Wesley (2009) Seconda edizione ISBN: 9788871925592 (ogni altra edizione del libro va bene ai fini del corso / any other edition of the book (even the one in english) is good for the porpouse of the course) Testi di consultazione (consultation textbooks)

	A. Aho, M. Lam, R. Sethi, J. Ullman Compilers, Principles, Techniques & Tools Addison Wesley Stefano Crespi Reghizzi Linguaggi Formali e Compilazione Esculapio (2015)

SYLLABUS

Hrs	Frontal teaching
4	Introduction: Programming Languages and Programming Language Processors. Compilers and Languages. Machine Languages, Assembly Languages and Programming Languages evolution. Compilers and Interpreters. Virtual Machine. Structure of a compiler: Preprocessor. Linker and Loader. Phases of compilation. Front End and Back End. Scanning of a compiler.
6	Lexical analysis: Preliminary operations. Token and Lexemes. Lexical errors. Tokens and regular expressions. Regular definitions. Elimination of ambiguity. Finite Automata. DFA implementations. NFA simulations. Automatic Generation of a scanner. Use of Flex.
6	Resolution of Exercises about lexical analysis using FLEX software
6	Syntactic analysis: Context-free Grammars. Derivation trees. Ambiguous grammars. Deterministic and non-deterministic stack automata. Computational Complexity of a stack automata. Earley Algorithm. Syntactic errors and methods of error management.
6	Descending or top-down parser: Recursive descent parser. Parser LL(1). Elimination of left recursion. Left Factorization. First and Follow sets. Ascending or bottom up-parser: Parser shift-reduce. Parser LR(0). Parser SLR. Parser LR(1). Parser LALR(1). Properties of languages and of Grammars LR (k). Comparison between LL(k) and LR(k) grammars. Automatic parser generators. Use of Bison.
6	Semantical analysis: Static and dynamic semantics. Grammars with attributes. Semantics syntax driven. Decorated Syntactic tree. Computation of attributes. Dependency graph. Grammar with S- attributes. Grammar with L-attributes. Topological sorting of dependency graphs. Symbols table. Various types of implementations through arrays, chained lists, and ABRs. Implementation through hash table with chaining. Visibility Attributes and Methods of Implementation. Type checking. Equivalence of types. Type coercion.
6	Resolving exercises about syntactic analysis with the help of Bison software.
6	Use the symbol table.
2	Code Generation: Intermediate code. Three-way code. Data Structures for the Implementation of 3AC. Code for the virtual machine. P-code. Code optimization. Examples of machine-independent optimizations. Generators of object code. Examples of machine-dependent optimizations.