English version at the end of this document

Ano Letivo	2019-20
Unidade Curricular	COMPILADORES
Cursos	ENGENHARIA INFORMÁTICA (1.º ciclo)
Unidade Orgânica	Faculdade de Ciências e Tecnologia
Código da Unidade Curricular	14781061
Área Científica	CIÊNCIA DE COMPUTADORES
Sigla	
Línguas de Aprendizagem	Português-PT
Modalidade de ensino	Presencial
Docente Responsável	Fernando Miguel Pais da Graça Lobo

DOCENTE	TIPO DE AULA	TURMAS	TOTAL HORAS DE CONTACTO (*)
Fernando Miguel Pais da Graça Lobo	PL; T	T1; PL1; PL2	30T; 60PL

* Para turmas lecionadas conjuntamente, apenas é contabilizada a carga horária de uma delas.

ANO	PERÍODO DE FUNCIONAMENTO*	HORAS DE CONTACTO	HORAS TOTAIS DE TRABALHO	ECTS
3º	S2	30T; 30PL	168	6

* A-Anual;S-Semestral;Q-Quadrimestral;T-Trimestral

Precedências

Sem precedências

Conhecimentos Prévios recomendados

Programação, Programação Orientada a Objectos, Algoritmos e Estruturas de Dados, Matemática Discreta, Arquitectura de Computadores.

Objetivos de aprendizagem (conhecimentos, aptidões e competências)

- compreender os objetivos, arquitetura e abrangência de um compilador
- perceber a integração das diversas fases de compilação de linguagens imperativas
- ser capazes de especificar a sintaxe e a semântica de uma linguagem de programação
- saber utilizar os principais algoritmos e estruturas de dados usados na implementação de compiladores
- desenvolver componentes de um compilador para uma linguagem imperativa simples

Conteúdos programáticos

1. Conceitos introdutórios, panorâmica geral de um compilador
2. Linguagens formais e autómatos
2.1. Linguagens regulares e linguagens independentes do contexto
2.2. Autómatos finitos
3. Análise lexical
4. Análise sintática (parsing)
4.1. gramáticas independentes do contexto
4.2. derivações
4.3. árvores sintáticas abstractas
4.4. parsing top-down
4.4.1. parsing recursivo descendente
4.4.2. transformação de gramáticas, parsing preditivo
4.4.3. gramáticas LL(1)
4.5. parsing bottom-up
4.5.1. parser Shift-Reduce
4.5.2. parsing LR: LR(0) e SLR
5. Análise semântica
5.1. identificação
5.2. verificação de tipos
6. Ambientes de execução
6.1. representação de variáveis
6.2. alocação de memória numa máquina de pilha
6.3. avaliação de expressões
6.4. registos de activação para procedimentos e funções
6.5. acesso a variáveis locais e globais numa máquina de pilha
7. Geração de código
7.1. templates de código, casos especiais
7.2. algoritmo de geração de código baseado na visita da árvore sintática abstracta

Demonstração da coerência dos conteúdos programáticos com os objetivos de aprendizagem da unidade curricular

A matéria abordada nos conteúdos programáticos irá capacitar os alunos com as competências necessárias para que os objectivos de aprendizagem enunciados sejam atingidos. Obviamente que para tal acontecer, espera-se que os alunos acompanhem a matéria ao longo do semestre e realizem os exercícios e trabalhos práticos propostos.

Metodologias de ensino (avaliação incluída)

Nas aulas teóricas o professor faz a exposição da matéria usando o quadro e o computador para apresentar os conteúdos. As aulas práticas servem para consolidar os conceitos apresentados nas aulas teóricas, através da resolução de problemas e discussão de questões relacionadas com os trabalhos práticos.

A avaliação é feita por frequência comforme estipulado pelo Artº 9, ponto 1-b) do Regulamento de Avaliação da UAlg, e consite em:

- trabalhos práticos (30%)
- exame final (70%)

É necessário uma nota não inferior a 7,0 valores no trabalho prático para serem admitidos a exame final.

Os trabalhos práticos são feitos individualmente ou em grupo de 2 pessoas. O trabalho é composto por várias partes, cada qual com o seu prazo de entrega. A nota prática do grupo é convertida numa nota prática individual no momento da discussão dos trabalhos (que ocorrerá na última semana de aulas). A nota prática é, portanto, individual, e depende do desempenho de cada elemento do grupo na discussão.

Demonstração da coerência das metodologias de ensino com os objetivos de aprendizagem da unidade curricular

Os objectivos de aprendizagem só serão alcançados se a matéria dada nas aulas teóricas for exercitada na prática.
A metodologia de ensino adoptada e o método de avaliação proposto dão ênfase a este aspecto.

Bibliografia principal

Livro de texto principal

- Compiler Principles, Techniques, and Tools}, 2nd Edition, Addison Wesley, 2007.
Alfred V. Aho, Monica Lam, Ravi Sethi, Jeffery D. Ullman.

Referência útil para o trabalho prático

- Programming Language Processors in Java, 1st Edition, Pearson, 2000.
David Watt, Deryck Brown.

Referência complementar

Programming Language Pragmatics, 4th Edition, 2015.
- Michael L. Scott.

Referência complementar em língua portuguesa

Compiladores - da teoria à prática, Editora FCA, 2014
- Pedro Reis Santos, Thibault Langlois.

Academic Year	2019-20
Course unit	COMPILERS
Courses	INFORMATICS (COMPUTER SCIENCE) (1st Cycle)
Faculty / School	FACULTY OF SCIENCES AND TECHNOLOGY
Main Scientific Area	CIÊNCIA DE COMPUTADORES
Acronym	
Language of instruction	Portuguese-PT
Teaching/Learning modality	In-person lectures.
Coordinating teacher	Fernando Miguel Pais da Graça Lobo

Teaching staff	Type	Classes	Hours (*)
Fernando Miguel Pais da Graça Lobo	PL; T	T1; PL1; PL2	30T; 60PL

* For classes taught jointly, it is only accounted the workload of one.

Contact hours

T	TP	PL	TC	S	E	OT	O	Total
30	0	30	0	0	0	0	0	168

T - Theoretical; TP - Theoretical and practical ; PL - Practical and laboratorial; TC - Field Work; S - Seminar; E - Training; OT - Tutorial; O - Other

Pre-requisites

no pre-requisites

Prior knowledge and skills

Computer programming, Object Oriented Programming, Algorithms and Data Structures, Discrete Mathematics, Computer Arquitecture.

The students intended learning outcomes (knowledge, skills and competences)

- understand the goals and arquitecture of a compiler
- understand the various phases of compiler construction
- be capable of specifying the syntax and semantics of a programming language
- know the major algorithms and data structures used in the implementation of a compiler
- develop components of a compiler for a simple imperative language

Syllabus

1. Introductory concepts, overview of a compiler
2. Formal languages and Automata
2.1. Regular languages and Context-free languages
2.2. Finite automata
3. Lexical analysis
4. Syntax analysis (parsing)
4.1. context-free grammars
4.2. derivations
4.3. abstract syntax trees
4.4. top-down parsing
4.4.1. recursive descent parsing
4.4.2. grammar transformations, predictive parsing
4.4.3. LL(1) grammars
4.5. bottom-up parsing
4.5.1. Shift-Reduce parsing
4.5.2. LR parsing: LR(0), SLR
5. Semantic analysis
5.1. identification
5.2. type checking
6. Run-Time Environments
6.1. data representation
6.2. memory allocation on a stak machine
6.3. expression evaluation
6.4. activation records for functions and procedures
6.5. access to local and non-local names on a stack machine
7. Code Generation
7.1. code templates, special cases
7.2. code generation algorithm based on visiting an abstract syntax tree

Demonstration of the syllabus coherence with the curricular unit's learning objectives

The topics covered in the syllabus allow the students to acquire the necessary skills to reach the learning outcomes.
It is however expected that students follow the material throughout the semester and do the proposed practical assignments and the course practical project.

Teaching methodologies (including evaluation)

In the main T lectures, the course materials are explained to students along with illustrative examples.
In the P lectures students have hands-on experience on the course materials doing exercises and working on their projects.

The grading follows "avaliação por frequência" as stated in Art. 9, 1-b) of the 'Regulamento de Avaliação' of UAlg, and consists of:

- programming project (30%)
- final exam (70%)

Students need a grade greater or equal to 7,0 in the project to be admitted to the final exam.

The programming project has to be done individually or in a group of a maximum of 2 people. The project is composed of several parts, each with its own deadline. Each part corresponds to a task required in the development of a compiler. The project (practical) grade of the group will be converted into a student grade upon the mandatory individual project discussion that will take place in the last week of lectures of the semester.

Demonstration of the coherence between the teaching methodologies and the learning outcomes

The learning outcomes will only be reached if the material covered in the main T lectures is exercised in practice.
The teaching methodology and the proposed evaluation/grading give emphasis to this aspect.

Main Bibliography

Main textbook

- Compiler Principles, Techniques, and Tools}, 2nd Edition, Addison Wesley, 2007.
Alfred V. Aho, Monica Lam, Ravi Sethi, Jeffery D. Ullman.

Useful reference for the programming project

- Programming Language Processors in Java, 1st Edition, Pearson, 2000.
David Watt, Deryck Brown.

Additional reference

Programming Language Pragmatics, 4th Edition, 2015.
- Michael L. Scott.

Book in portuguese for those not so familiar with english:

Compiladores - da teoria à prática, Editora FCA, 2014
- Pedro Reis Santos, Thibault Langlois.