Linnæus University

Jnr: 2018/2843-3.1.2.2

Course syllabus

Faculty of Technology
Department of Computer Science and Media Technology

4DV507 Kodtransformationer och interpretation, 5 högskolepoäng
Code transformation and interpretation, 5 credits

Main field of study
Computer Science

Subject Group
Informatics/Computer and Systems Sciences

Level of classification
Second Level

Progression
A1N

Date of Ratification
Approved by Faculty of Technology 2018-10-08
The course syllabus is valid from spring semester 2019

Prerequisites
90 credits in Computer Science (including a degree project at Bachelor level).

Objectives

After completing the course the students should be able to:

- describe the different phases in the compilation process
- describe different parsing techniques
- explain what happens during semantic analysis
- explain how type systems for common programming languages work
- explain how a stack machine works
- design finite automata-based lexical analysis and an LL(1) parser for a simple programming language
- design and develop a semantic analysis that includes error handling and simple type inference, and that decorates the syntax tree with type information
- implement a parser using a given parser generator
- design and develop a stack maschine-based virtual machine
- judge the difficulty of implementing various programming language constructs
- select an appropriate formal notation to describe a given formal language.

Content

The course presents techniques, theories, and tools used to constructing a compiler. The course also discusses how these ideas can be used to define and interpret domain-specific languages within model-driven software engineering. As a result, the course focuses on thecompiler front-end and runtime interpretation of intermediate program representations.

The following topics are covered:

- Different compilation phases
- Object-oriented compiler design
- Lexical analysis based on finite automata and regular languages
- Context-free grammars and languages
- Different parsing techniques for context-free languages
- Type systems and type inference
- Attributed grammars
- Semantic analysis
- Intermediate program representations
- Code generation
- Stack-based execution

Type of Instruction

The types of instruction for this course encompass traditional lectures for teaching the majority of the course content. In addition, the content is exercised and deepened in context of tutoring sessions related to the practical and written assignments. The written assignments are carried out individually, the practical assignments are carried out in groups of two students.

Examination

The course is assessed with the grades A, B, C, D, E, Fx or F.
The grade A constitutes the highest grade on the scale and the remaining grades follow in descending order where the grade E is the lowest grade on the scale that will result in a pass. The grade F means that the student's performance is assessed as fail (i.e. received the grade F). Assessment of student performance is made through theoretical assignments, programming assignments, and a written exam. Students who do not pass the regular examination will be offered retrials close to the regular examination.
To pass the course, grade E or higher is required for all parts. The final grade is decided from: theoretical assignments (20%), programming assignments (40%), and written exam (40%).

Course Evaluation

During the course or in close connection to the course, a course evaluation is to be carried out. The result and analysis of the course evaluation are to be communicated to the students who have taken the course and to the students who are to participate in the course the next time it is offered. The course evaluation is carried out anonymously. The compiled report will be filed.

Other

Grade criteria for the A–F scale are communicated to the student through a special document. The student is to be informed about the grade criteria for the course by the start of the course at the latest.

The course is conducted in such a way that the course participants' experiences and knowledge are made visible and developed. This means, for example, that we have an inclusive approach and strive for no one to feel excluded. This can be expressed in different ways in a course, for example by using the gender neutral example.

Required Reading and Additional Study Material

Alfred V. Aho, Monica S. Lam, Ravi Sethi, och Jeffrey Ullman, *Compilers: Principles, Techniques, and Tools*, Pearson Education, latest edition. Pages: 510 av 986.