GÖTEBORG UNIVERSITY
The Board of the IT Faculty

DIT230, Programming Language Technology, 7,5 higher education credits

Second Cycle/A1N

This syllabus in English is a binding document.

1. Confirmation

The syllabus was confirmed by the Faculty Board of IT Faculty/The Dean on 2006-11-17 to be valid from the autumn semester 2007. It has been revised on 2007-10-23 to be valid from autumn semester, 2008 and on 2009-10-15 to be valid from spring semester 2010.

Field of education: Sciences.

Responsible department: Computer Science and Engineering.

2. Position in the educational system

The course is a part of the Computer Science Master´s programme and a single subject course at the University of Gothenburg. The level for the course in relation to degree requirements is Master´s degree, code A1N. The course has course/courses at first cycle level as entry requirements.

3. Entrance qualifications

The requirement for the course is to have successfully completed a Bachelor degree within Computer Science or equivalent.

4. Course content

The teaching consists of lectures, exercises, and laborations, as well as individual supervision in connection to the laborations.

5. Learning outcomes

The aim of the course is to give understanding of how programming languages are designed, documented, and implemented. The course covers the basic techniques and tools needed to write interpreters, and gives a summary introduction to compilation as well.

After completing the course the student is expected to be able to:

- Define the lexical structure of programming languages by using regular expressions, explain the functioning of finite automata, and implement lexical analyzers by using standard tools;
- Define the syntax of programming languages by using context-free grammars, explain the principles of LL and LR parsing, and implement parsers by using standard tools;
- Define and implement abstract syntax
- Master the technique of syntax-directed translation and its efficient implementation in their chosen programming language;
- Formulate typing rules and implement type checkers;
- Implement polymorphic type checking by using unification;
- Formulate operational semantic rules and implement interpreters for both imperative and functional languages;
- Write simple code generators;
- Be familiar with the basic implementation issues of both imperative and functional languages;
- Design and implement special –purpose programming languages.

6. Required reading

See separate literature list.

7. Assessment

The student is evaluated through Lab work and a final written exam.

8. Grading scale

The grades are Pass, Pass with distinction or Fail.

9. Course evaluation

The course is evaluated through meetings both during and after the course between teachers and student representatives. Further, an anonymous questionnaire can be used to ensure written information. The outcome of the evaluations serves to improve the course by indicating which parts could be added, improved, changed or removed.

10. Additional information

The course is given in English.