Programming Languages Q-Exam Syllabus
Fall 1997

Language Principles, Paradigms, and Features:

This material is covered in an undergraduate programming language course, such as CS411. Students should have a good working knowledge of each language paradigm and the principles that distinguish this paradigm from others. Students should also be familiar with the operational semantics of at least one language from each category. Finally, students should have a fairly complete working knowledge of core concepts such as scoping, types, encapsulation mechanisms, and threads.

1. Imperative languages: assignment, while/for/do/repeat loops, arrays, etc. (e.g., FORTRAN)
2. Procedural languages: procedures, functions, parameter-passing mechanisms, lexical scope, etc. (e.g., Pascal, C)
3. Abstract types: modules, interfaces, data abstraction, polymorphism. (e.g., Modula, Clu, Eiffel, Ada, SML)
4. Functional languages: a basic understanding of typed lambda-calculus, higher-order functions, lexical vs. dynamic scope. (e.g., ML, Haskell, Lisp, Scheme)
5. Logic and relational languages: a basic understanding of propositional and first-order predicate calculus, Horn clauses, unification. (e.g., Prolog, Mercury)
6. Object-oriented languages: objects and methods,, subtypes and the "subtypes as subsets" principle, classes, inheritance and method override. (e.g., Smalltalk, Java, C++)
7. Concurrent and parallel languages: threads, synchronization primitives (e.g., mutex locks, condition variables, monitors, fork/join, channels, rendezvous). (e.g., Modula-3, Java, Ada,Concurrent ML)

General References:

G.Winskell. *The Formal Semantics of Programming Languages: An Introduction.* MIT Press, 1994.

R.Sethi. *Programming Languages: Concepts and Constructs,* Second Edition, Addison-Wesley, 1996.

A.J.Field and P.G. Harrison. *Functional Programming.* Addison-Wesley, 1988.

Some Specific Language References:

L.C.Paulson. *ML for the Working Programmer.* Cambridge University Press, 1991.

L.Sterling and E.Shapiro. *The Art of Prolog.* MIT Press, 1986.

A.Goldberg and D.Robson. *Smalltalk-80: The Language and Its Implementation.* Addison-Wesley, 1991.

B.Stroustrup. *The C++ Programming Language*, Second Edition. Addison-Wesley, 1991.

K.Arnold and J.Gosling. *The Java Programming Language.* Addison-Wesley, 1996.

Implementation:

This material is covered in an undergraduate compiler course, such as CS412. Students should be able to apply the ideas in other situations where program translation needs to be done, e.g. in typesetting systems.

1. Lexical analysis, context-free grammars; top-down and bottom-up parsing; LL, LR, and LALR grammars; generation of parsers and scanners.

2. Syntax-directed translation.

3. Run-time environments: stacks, static and dynamic links, displays; dynamic storage allocation (e.g., malloc/free); garbage collection; symbol tables; parameter-passing mechanisms (e.g., call-by-value, copy-in-copy-out, etc.)

4. Basic code generation and code optimization: register allocation, copy-propagation, constant-folding, strength-reduction, induction-variable elimination, jump-threading, etc.

5. Program analysis: type-checking; dataflow analysis

Example References:

A.V.Aho, R. Sethi, and J.D.Ullman. *Compilers: Principles, Techniques, and Tools.* (The "Dragon Book".) Addison-Wesley, 1988.

S.S.Muchnick. *Advanced Compiler Design & Implementation.* Morgan Kaufmann, 1997.

C.N.Fischer and R.J.LeBlanc,Jr. *Crafting a Compiler.* Benjamin/Cummings, 1988.

T.W.Pratt and M.V. Zelkowitz. *Programming Languages: Design and Implementation.* Prentice-Hall, 1996.

Programming Methodology:

1. Partial and total correctness of sequential programs: weakest preconditions and Hoare-style proof rules for imperative languages.

2. Formal development of sequential programs.

Example References:

D.Gries. *The Science of Programming.* Springer-Verlag, 1981.