Course Outcomes	CSC 4101

CSC 4101: Programming Languages

Credit Hours: 3 hours

Prerequisites:

CSC 3102

Prerequisites by Topics:

Object-oriented programming; recursion; data structures.

Catalog Course Description:

Principles of programming languages design; specification of syntax and semantics; underlying implementation of block structured languages; dynamic memory allocation for strings, lists, and arrays; imperative versus applicative programming; logic programming; modern programming languages.

Course Outcomes

1. Master using syntax-related concepts including context-free grammars, parse trees, recursive-descent parsing, printing, and interpretation.
2. Master analyzing semantic issues associated with function implementations, including variable binding, scoping rules, parameter passing, and exception handling.
3. Master implementation techniques for interpreted functional languages.
4. Master using object-oriented languages.
5. Be familiar with design issues of object-oriented and functional languages.
6. Be familiar with language abstraction constructs of classes, interfaces, packages, and procedures.
7. Be familiar with implementation of object-oriented languages.
8. Be familiar with using functional languages
9. Be exposed to using logic languages.

Texts and Other Course Materials

Programming Language Pragmatics- Michael L. Scott. 1-55860-442-1. HB Latest Morgan Kaufman.

Major Topics

- History of programming languages.
- Compilation versus interpretation.
- Overview of compilation including lexical and syntax analysis, semantic analysis and intermediate code generation.

- Programming in a functional language, such as Scheme and ML.
- Specifying syntax using BNF regular expressions and context-free grammars.
- Recursive descent parsing and parse trees.
- Data abstraction and object orientation.
- Design issues for object-oriented languages.
- Binding times, static and dynamic scoping.
- Object lifetime and storage management including stack-based allocation and heap-based allocation.
- Types, type equivalence, conversion, casting, compatibility, coercion, and inference,
- Implementation of function calls, including static vs. stack vs. heap allocation of activation records, static and dynamic links, parameter passing, and closures.
- Garbage collection.
- Design issues for procedures, including generic procedures and exception handling.
- Logic programming.

Assignments/Projects/Laboratory Projects/Homework

- Recursive descent parser and pretty-printer for Scheme written in C++ or Java.
- Interpreter for a small subset of Scheme written in C++ or Java.
- Library of Scheme built-in functions implemented in Scheme subset.
- Small Scheme, ML, and Prolog programs on homeworks.

Curriculum Category Content (estimated in semester hours)

Area	Core	Advanced	Area	Core	Advanced
Algorithms		2	Data Structures	4	6
Software Design	2	2	Prog. Languages	10	10
Computer Arch.					

Relationship to Criterion 3 Outcomes

A	B	C	D	E	F	G	H	I	J	K
*	*	*		*	*	*		*		*

Math and Fundamentals:

Data Structures:

Implementation and use of parse trees, frames, environments, closures, lists (10 hrs).

Algorithms and Software:

Design and implementation of recursive descent parsing, printing, interpretation (6 hrs).

Computer Organization and Architecture:

Next to none.

Concepts of Programming Languages:

Object-oriented, functional, and logic languages, interpreters (20 hrs).

Social and Ethical Issues:

None.

Oral Communication (presentations)

None.

Written Communication:

4 homeworks, 3 programming assignments.

Course Coordinator: Dr. Gerald Baumgartner

Last Modified: June 10, 2007.