Course Outline (Subject to change)

This course will define, analyze and evaluate important concepts found in current programming languages. Its goals are to build an ability to evaluate and compare programming languages, both from the user’s and implementor’s view, and to develop precise mechanisms for specifying the semantics of programming languages.

Course Outline (Subject to change)...

In particular, we will cover the following topics:

1. scope of objects and time of binding
2. module mechanisms (e.g., blocks, procedures, coroutines)
3. data abstraction, datatypes
4. control structures
5. storage management and runtime support
6. operational, denotational, and axiomatic semantic specification; attribute grammars
7. applicative and object-oriented languages
Grading (Subject to change)

1. One final exam (40%), Friday, May 9, 11:00–13:00.
   (a) The exam is closed book.
   (b) Without prior arrangement, missed exam ⇒ grade of zero.
   (c) Fail the exam ⇒ you might fail the course.
2. “Several” homework assignments (40%). Homewors will require programming and/or theoretical work.
3. One final project (20%). The project should be done in teams of two.

Grading (Subject to change)...

If your graded score for an homework is $g$ and you handed in $k$ days late, then your computed score for this assignment will be

$$
\begin{cases}
\max(0, g(1 - 0.1k)) & \text{if } k \leq 5 \\
0 & \text{otherwise}
\end{cases}
$$

Late project reports/synopsis will not be accepted!
Suggested Final Project Topics...

- Other languages: Forth, APL/J, SETL.
- Parallel programming languages: Occam, Linda.
- Obfuscated languages: Brainf*ck, Befunge, Intercal.
- Scripting languages: Perl, Python, Tcl, Ruby, SNOBOL, JavaScript.
- Semantic description languages: Action semantics, Z.

Suggested Final Project Topics...

- Novel paradigms: visual programming languages, multiparadigm programming languages (Leda), constraint programming languages, literate programming languages, mobile programming languages, aspect-oriented programming languages (Aspect.J), dataflow programming languages, functional-logic programming.
- Language features: macros (Scheme), futures, module systems (ML), persistence (Napier88), input/output facilities.
- Implementation techniques: runtime code generation.
- Theory: Typed assembly language, proof-carrying code.

Final Project...

- Read 2-3 theoretical programming language papers on a topic that interests you. Summarize and critique the papers.
- Study how some language feature not covered in class has been realized in a few programming languages. Write programs in each language to show how the feature impacts real programs. Compare, contrast, and critique.

Suggested Final Project Topics...

- OO languages: Cecil, BETA, Eiffel, Obliq, Pizza, Sather, Simula, C#, Objective-C, Smalltalk.
- FP languages: Erlang, Hope, Sisal.
- LP languages: Mercury.
- Special purpose languages: compiler construction (Gentle), symbolic mathematics (Macsyma, Matlab, Mathematica), web programming (WebL, PHP), database programming (SQL, Mumps, RPG, COBOL), linking languages (Mesa), teaching languages (Pascal, Turing, Logo, ABC), typesetting languages (\TeX/\LaTeX, Postscript, PDF), architecture description languages (VHDL), computer gaming languages, robotic control languages.
1. Introduction:
   - Language design goals.
   - Language translation systems.

2. A survey of important programming languages and paradigms:
   - Functional programming: Scheme and Gofer/Haskell.
   - Logic programming: Prolog.
   - Procedural programming: Pascal, Modula-2, Ada.
   - Object-Oriented programming: Modula-3 and C++.
   - String programming: Icon.

I will lecture on each language. You will be responsible for reading up on the details of the languages. This will involve getting out books from the library, searching the web for relevant tutorials, etc. There will be one small programming assignment for each language.

3. A study of programming language constructs:
   - Names and scope.
   - Expressions.
   - Control structures (loops, iterators, short-circuit evaluation).
   - Data types (type equivalence and inference, records, arrays, strings, pointers).
   - Runtime organization (calling sequences, exceptions, nested procedures, coroutines, parameter passing, garbage collection).
   - Object-oriented programming (single/multiple inheritance, dynamic dispatch, run-time typechecking).

To study each feature you will write small interpreters (in Prolog) and/or small test-programs in several languages.

4. Formal description of programming languages:
   - Lexical/syntactic specification (regular expressions, context free grammars, EBNF).
   - Concrete vs. abstract syntax.
   - Lambda calculus.
   - Semantic specification (denotational, axiomatic, and operational specification, attribute grammars).

You will write formal specifications of small languages.
**Handouts & Other Material**

1. I always make copies of my transparencies available to students. Note that
   - I do this to relieve you of having to take notes during lectures,
   - they are not substitutes for reading the textbook,
   - their primary purpose is to remind you of what you need to study for the exam.

2. Various manuals and papers will be handed out during class. Extra copies can be picked up from the boxes outside my office.

3. Various information regarding the course (including postscript files of the handouts) can be found on the info-bahn: [http://www.cs.arizona.edu/~collberg/Teaching/520/2003/index.html](http://www.cs.arizona.edu/~collberg/Teaching/520/2003/index.html)

**Academic Integrity**

- You will not
  1. turn in another student’s work as your own.
  2. use material from the web, textbooks, journals, etc. without giving the appropriate attribution.
  3. accept solutions from other students.
  4. give solutions to other students.
  5. tamper with graded papers or exams.
  6. collaborate with students outside your own team.

- Sanctions typically include:
  - grade reduction, course failure, suspension, expulsion.

- *I take this stuff seriously.*

**Handouts & Other Material...**

2. Various manuals and papers will be handed out during class. Extra copies can be picked up from the boxes outside my office.

**Academic Integrity ...**

- Students who violate the Code are also subject to possible sanctions imposed by the Dean of Students office.

- Submitted solutions will be compared with each other, as well as with solutions from previous semesters.

- All students involved in collusion are equally culpable:
  1. Do not give another student access to your account.
  2. Do not leave printouts in the recycling bin.
  3. Pick up your printouts promptly.
  4. Do not leave your workstation unattended.

If you suspect that your work has been compromised notify me immediately.
Free Compilers and Interpreters

Scheme: http://www.drscheme.org
Modula-2: http://floppsie.comp.glam.ac.uk/Glamorgan/gaius/web/GNUModula2.html
Modula-3: http://www.m3.org

Also see http://www.idiom.com/free-compilers.

Installed Translators

- Compilers and interpreters available on lectura:
  Pascal: gpc
  Scheme: scheme and scheme48.
  C,C++,Objective-C: gcc.
  ML: sml.
  Icon: icont.
  Prolog: prolog.
- Compilers and interpreters available on linux:
  Gofer: /home/cs520/2003/bin/linux/gofer