520 — Principles of Programming Languages

0: Administrivia

Christian Collberg
collberg@cs.arizona.edu

Department of Computer Science
University of Arizona

Contact Information

Class : 520 PRINCIPLES OF PROGRAMMING LANGUAGES
Lecturer : Christian Collberg
Email : collberg@cs.arizona.edu
WWW : http://www.cs.arizona.edu/~collberg
Office : 758
Office Hours : Open door policy
Phone : 621-6612
Lectures : 10:30-11:45, MW, GLD-S 701
Book : Programming Language Pragmatics. Michael Scott
TA : TBA

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.
We will evaluate and compare languages both from the user’s and implementor’s view.
We will develop precise mechanisms for specifying the semantics of programming languages.

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 (50%), Friday, May 15, 08:00–10:00.
   (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%). Homeworks will require programming and/or theoretical work.
3. One paper presentation (10%). You will prepare a presentation of a research paper. This will be done in teams of two. The presentation will take ca 15 minutes.

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

$$
\max(0, g(1 - 0.1k)) \quad \text{if } k \leq 5
$$

0 \quad \text{otherwise}

Late lecture slides will not be accepted!

Presentations

Towards the end of the semester there will be presentations of research papers relevant to programming languages.

1. I will make a list of papers available in my office.
2. Papers will be assigned on a first-come-first serve basis.
3. Presentations are expected to last 15 minutes.
4. You must prepare a comprehensive set of slides that covers the topic. You must use LaTeX/Prosper. A template will be provided.
5. Two days prior to your presentation you must have submitted the slide sources.

6. You must give a practice talk 2 days before the class. Draft either your team buddy or a friend from the class. The idea is to make sure that you know what you’re talking about and that the presentation is of appropriate length.
7. You are responsible for printing out copies of the slides (the file handout.ps) prior to the class to hand out to the students.
Course Organization (Subject to Change)

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 Smalltalk.
   - 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.

Course Organization...

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 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.
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.

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.

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.

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.
3. Various information regarding the course (including postscript files of the handouts) can be found on the info-bahn:
Free Compilers and Interpreters

Ada:


Modula-2:


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

Reading a Research Paper

1. Read the abstract, the introduction, and the conclusion.
2. Do you know what’s going on? If not, you need to do some background reading:
   (a) Check the paper’s reference (bibliography) section.
   (b) Look for tech-reports by the same author; they often contain details left out in a published article.
   (c) Journal articles are more detailed than conference articles.
   (d) Textbooks are sometimes helpful, but not for “new” topics.

3. Read the rest of the article, but skip proofs and other technical detail.
4. Read it again, but this time make up and work some examples to make you understand the details better.
5. Read some other articles in the same area but by different authors to give you a different perspective.
Preparing a Presentation

1. Think of good lectures you’ve been to: how did the lecturer organize the presentation?
2. Decide what audience you are talking to:
   - Are they familiar with the topic? If not, spend a lot of time on background information!
   - Are they mathematically minded? If not, avoid \( \lambda \sum_1^n (\Gamma(n))! \)
   - Are they awake? If not, make sure everything’s on the handout!
3. Avoid too much technical detail!!!
4. Avoid too much technical detail!!!

Preparing a Presentation...

4. Decide what the audience should remember after the talk. E.g.:
   - What problem is the authors trying to solve?
   - What previous attempts have there been at solving this problem?
   - What’s the authors’ principal idea?
   - Is this a solved problem now, or does the paper leave some open problems?
   - Is it, in your opinion, a good or a bad paper? Why?

Giving the Presentation

1. Make sure you have adequate notes:
   - If you’re likely to be very nervous, write a script.
   - Otherwise, topic headings will suffice.
2. Talk slowly and clearly.
3. Rehearse the presentation:
   - Give the presentation to yourself. Read it out loud! It takes time to get used to hearing your own voice.
   - Time yourself. Do you have enough material; too much material?
   - Give the presentation to a friend. Not your boy/girlfriend; you’ll want someone who dares to criticize you...