Course Description

CS 321 describes formal methods for specifying the syntax and semantics of languages, and uses these methods to help build the compiler's front end. Programming language structure and capabilities are examined with an emphasis on understanding the demands of efficient implementation.

Prerequisites

This class requires CS 201, 202, 300, 311; passed with grades of C or better. These courses provide you with the background that you will need to understand the course material and develop solutions to the assignments that you will be given. It is assumed that you have experience programming in a high-level, object-oriented language (specifically, C++ and Java, from CS 202 and earlier classes). These programming skills are essential because the CS321/322 sequence is centered around substantial programming projects such as implementing a compiler for a realistic language. If you have not completed the prerequisites, then you may not have the background that you need to pass this class. In this case, by default, you will not be able to continue in the class and you should contact the instructor at the earliest opportunity to avoid an administrative drop.

Course Objectives/Student Learning Outcomes

Upon the successful completion of this course students will be able to:

1. Explain the phase structure of a typical compiler and the role of each phase.
2. Describe and apply mechanisms for defining the lexical structure of a programming language.
3. Use context-free grammars to define syntax of a programming language.
4. Describe and apply mechanisms for transforming grammars to make them suitable for predictive parsing, and for building LL(1) parsing tables.
5. Explain the operation of a shift-reduce bottom-up parser.
6. Construct abstract syntax trees for programs in a simple language.
7. Distinguish between inherited attributes and synthesized attributes, and use them to define simple syntax-directed actions.
8. Describe and apply the basic concepts of data abstraction, encapsulation, object-oriented classes, and modules.
9. Describe and apply the basic concepts of type systems, including primitive types, aggregate and recursive types, abstract data types, and type equivalence models.
10. Contrast the main features of different programming paradigms, including procedural, object-oriented, and functional.
11. Starting from a partial implementation, complete a lexical analyzer, parser, and type-checker for a simple but realistic language.

Outline of Course Content
The course will be taught using one lecture and one instructor-led lab session for each week of the term. Lab sessions will be used to provide opportunities for more hands-on exercises and experimentation, and will be used both to introduce new material and to reinforce topics presented in lectures.

<table>
<thead>
<tr>
<th>Week</th>
<th>Lecture Date</th>
<th>Lecture Topic</th>
<th>Lab Topic</th>
<th>Assignments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1/5</td>
<td>Introduction/overview</td>
<td>Compiler phases/using Java</td>
<td>Warmup (5) assigned</td>
</tr>
<tr>
<td>2</td>
<td>1/12</td>
<td>Lexical analysis</td>
<td>Lexer generators</td>
<td>Warmup due, Lexer (10) assigned</td>
</tr>
<tr>
<td>3</td>
<td>1/19</td>
<td>No lecture (MLK Jr Holiday)</td>
<td>Recursive descent parsing</td>
<td>Lexer in progress</td>
</tr>
<tr>
<td>4</td>
<td>1/26</td>
<td>CFGs, trees, ambiguity</td>
<td>CFG transforms, parse trees vs ASTs</td>
<td>Lexer due, Parsing (5) assigned</td>
</tr>
<tr>
<td>5</td>
<td>2/2</td>
<td>Top down parsing</td>
<td>Parser generator pragmatics. Midterm review.</td>
<td>Parsing in progress/midterm review</td>
</tr>
<tr>
<td>6</td>
<td>2/9</td>
<td>Midterm exam</td>
<td>AST assignment help</td>
<td>Parsing due, AST (10) assigned</td>
</tr>
<tr>
<td>7</td>
<td>2/16</td>
<td>Bottom up parsing</td>
<td>Computing over ASTs</td>
<td>AST in progress</td>
</tr>
<tr>
<td>8</td>
<td>2/23</td>
<td>Static analysis</td>
<td>Types and type checking</td>
<td>AST due, Static (15) assigned</td>
</tr>
<tr>
<td>9</td>
<td>3/2</td>
<td>Type systems</td>
<td>Types in Haskell</td>
<td>Static in progress</td>
</tr>
<tr>
<td>10</td>
<td>3/9</td>
<td>Abstract datatypes and modules</td>
<td>Review for final</td>
<td>Static due (end of week)</td>
</tr>
</tbody>
</table>

In accordance with University Guidelines, the final exam is scheduled for Monday, March 16, 5:30-7:20pm in ASRC 230.

Course Requirements and Method of Evaluation

There will be five projects/assignments during the term (Warmup, Lexer, Parsing, AST, and Static). Details of these assignments will be made available during either lectures or lab sessions in the weeks indicated by the schedule below. (Assignments will typically not be released until after all of the lab sessions for the week.) Most assignments will involve significant programming work that you will be required to complete using the Java programming language. There will be a minimum passing grade on each assignment, and students must pass every assignment to avoid receiving an F or X grade.

There will be one mid-term exam and one final exam, both of which will be closed-book. Exams will cover topics from lectures, emphasizing material that is not directly covered by or relevant to programming assignments, so lecture attendance is important. Exams are scheduled in advance and, unless a prior arrangement is made, a grade of zero will be recorded for missed exams.

There may be some assignments that are not formally assessed. This might include written exercises, additional reading, or watching prepared videos, etc. These will be designed to help you master the subject, to keep pace with the lectures, and to prepare for the exams, so it will be very important for you to keep up to date with those exercises.

The weightings that will determine your final grade are as follows: graded projects/assignments: 45%; midterm: 25%; final: 30%.
Required Texts and/or Required Reading List

The slides that are presented in lectures and all of the materials and notes that are provided in labs are an essential part of the course and are required reading.

There is no required textbook for this class, but all students are strongly encouraged to do background reading in parallel with our coverage of topics in the class. There are lots of good textbooks on this subject and we list some of our favorites below. However, we do not expect you to buy or have access to any particular text: for example, we will not set exercises that come from a text book, or require you to have read specific sections of any particular book. So if you find a compiler text in the library, or borrow one from a friend, that will probably still be a good choice for you.

Of course, there are also lots of good resources on the Internet. In particular, we mention "Basics of Compiler Design" by Torben Mogensen, which is available for free in pdf form from http://www.diku.dk/~torbenm/Basics/. You can also find lots of good information on the topics of these classes in places like Wikipedia, or simply by searching on Google.

If you are really looking for a hard copy book to add to your library, our personal favorites (in no particular order) are as follows:


Computing Facilities

This course will be taught using the Java programming language, and all programming assignments for this course must be completed using Java. You may develop your solutions on any machine and operating system you like, so long as it supports Java (JDK Version 1.6 or later). However, if you want assistance from the course staff, then you should do your development on Unix or Linux; help for Windows may be limited.

Academic Integrity

We follow the standard guidelines for academic integrity. It is permissible to discuss assignments with other
students, but you must develop the solution yourself (although you can consult the tutors for help in debugging). Do not, under any circumstances, copy any part of another person's solution and submit it as your own. Writing code for use by another, or using another's code in any form (even with their permission) will be considered cheating. Cheating on an assignment or exam will result in an automatic zero grade for that piece of work, and the initiation of disciplinary action at the University level. Please refer to http://www.pdx.edu/dos/codeofconduct for details of the general PSU Student Code of Conduct.

Disabilities and Accommodations

If you are a student with a documented disability who is registered with the Disability Resource Center, please contact the instructors immediately to arrange any needed academic accommodations, and let us know as soon as possible if you feel that your needs are not being met. If you have accommodations that include taking tests at the University test center, you should take steps to make the necessary reservations at the earliest possible opportunities; the times and dates for both the midterm and final exams are already fixed for 2/9 and 3/16, respectively, as described above.

Subjective Matters

To ensure consistency, all questions about matters that are potentially subjective—such as possible grading errors or determining whether an exceptional circumstance warrants an extension of a deadline—should be addressed to the primary instructor for this class.