CSCI 395 – Fall 2007
Compilers

Ben Coleman
coleman@cs.moravian.edu
214 PPHAC
Office Phone: 610-625-7781

Office Hours: MWF 9:00 - 10:00
or by appointment
AIM: bjcoleman15

Course Description

Broadly speaking, a compiler is a program that translates input from one representation to another while verifying its form. Although traditionally this translation is from a programming language to machine language, compiler theory is now used in an amazingly wide variety of applications. RTF documents, email headers, XML files, and web server logs are all examples of human-readable formats that can be processed and verified quickly and easily using compiler technology.

We will study the fundamental components of all compilers: lexical analysis, parsing, semantic actions, intermediate representation trees, and code generation. Throughout the semester, we will develop a compiler for a small, but fully function subset of Java. Because this will be a substantial project, a significant portion of the course will emphasize techniques to design, test, and debug the project.

Course Goals

Upon completion of this course, a successful student will be able to:

- Describe the stages of compilation and how they apply to modern, non-traditional applications.
- Describe the process of generating a lexical analyzer from the regular expressions that represent the language tokens.
- Construct lexical analyzers and both top-down and bottom-up parsers using parser generator tools.
- Implement recursive data structures using the composite pattern.
- Generate intermediate representation trees and use the visitor pattern to perform appropriate actions on the tree (semantic analysis, optimizations, and target-language generation).
- Design, implement, and test a symbol table and other supporting data structures.
- Plan and implement pair-programming activities.

Required Text

In addition to the following required text, supplementary readings will be given periodically during the semester.

- *Modern Compiler Implementation in Java – 2nd Edition* by Andrew W. Appel

You should expect to spend about an hour before each class session working through the readings. This means reading the text for detail, studying the syntax for new language features, and working to learn vocabulary – not just skimming through the material before class.

Graded Material

- **Homework** – The goal of homework problems is for you to practice using the current course content and to explore the topics in more detail. During the first half of the semester, homework problems will be assigned nearly every class session and typically will be due the next class period. During the second half of the semester, the programming projects will be more substantial, and therefore less homework will be assigned.
- **Programming Assignments** – The book is structured such that each chapter has its own programming project. The projects (except the first) build on each other so that you ultimately produce a functional compiler at the end of the course.

  The first project is designed to help you gain familiarity with Java, Eclipse, and unit testing, and it will be completed independently. This project will be graded based on both functionality and quality of the code.

  The other six or seven portions of the project have you construct a working compiler, and will be completed in pairs. Grades for these assignments will be based on functionality and quality of testing. At the end of an assignment, I will meet with each group so that the members can demonstrate their progress *on department equipment*. Grades will be assigned immediately, but can be retroactively increased by later demonstrating that additional functionality was added. In theory, it is possible to earn a perfect score on each assignment, but time constraints will likely make significant post-grade improvements impossible.
- **Tests** – There will be two tests during the semester. One will be given after we complete Chapter 3, and the other after we complete Chapter 6.
- **Final** – The final exam will be cumulative, and will be administered orally. You will schedule a mutually agreeable time to meet with me in my office *during or before* the final exam time scheduled by the registrar.

Grade Determination

- (30%) Homework
- (40%) Programming Assignments
- (20%) Tests
- (10%) Final

All grades will be calculated on the standard scale using pluses and minuses.

Responsibilities

Your attendance is expected at each class meeting. You are also responsible for the contents of reading assignments, handouts, class activities, and class email.

If you have a disability that may affect your participation in this course, please contact me immediately to discuss academic accommodations.

Academic Honesty

Except on tests, you are encouraged to discuss the material and work with other students in the course. Specifically, on homework and programming projects you may discuss any portion of the assignment with your fellow students. This policy does not allow you to copy another student's work verbatim – you must produce your own code or write-up of the material. Work together to learn the concepts, but keep in mind that you are ultimately responsible for the material on the tests.

Schedule

My plan for the course is to cover the first nine chapters of the book in order:

Chapter 1 – Introduction	1 week
Chapter 2 – Lexical Analysis	1 week
Chapter 3 – Parsing	2 weeks
Chapter 4 – Abstract Syntax	1.5 weeks
Chapter 5 – Semantic Analysis	1.5 weeks
Chapter 6 – Activation Records	1.5 weeks
Chapter 7 – Translation to Intermediate Code	2 weeks
Chapter 8 – Basic Blocks and Traces	1.5 weeks
Chapter 9 – Instruction Selection	2 weeks

The details of this syllabus are subject to change based on our progress through the material.