CSE 401: Introduction to Compiler Construction

Professor: Craig Chambers

- contact info: `chambers@cs`, Sieg 309, 685-2094
- office hours: Wednesday 2:30-3:30, Friday 1:30-2:30 in Sieg 309

Teaching Assistant: Vass Litvinov

- contact info: `vass@cs`, Sieg 225, 543-3368
- office hours: Tuesday 11:00-12:00 in Sieg 226a

Teaching Assistant: Zasha Weinberg

- contact info: `zasha@cs`, Chateau 115, 616-1854
- office hours: Monday 3:30-4:30 in Sieg 226a

Textbooks

- *Compilers - Principles, Techniques, and Tools* (the "Dragon Book"), Aho, Sethi, and Ullman

Prerequisites

- CSE 326: Data Structures
- CSE 341: Programming Languages
- CSE 378: Machine Organization and Assembly Language
- recommended: CSE 322: Formal Models

Goals

- learn principles and practice of language implementations
- understand tradeoffs between run-time and compile-time processing
- understand tradeoffs between language features, run-time efficiency, and implementation difficulty
- gain experience working with large systems software, object-oriented design, and C++

Project

This class includes a substantial implementation project. You will modify and extend an existing compiler, written in C++, for a toy language called PL/0. PL/0 is like a severely stripped-down version of Pascal. Initially, PL/0 is barely usable as a programing language. As part of the project, you will add several language features, including arrays, `for` loops, and arguments and results of procedures, making the language much more usable. The project will be broken up into several stages, each stage lasting two weeks, paralleling the sequence of study of compilers in the text and lectures. Some lectures will explore how the PL/0 compiler is organized and how textbook techniques are realized in the PL/0 compiler.

You may optionally form two-person teams to do the project, and you are **strongly encouraged** to do so. Grading on the project will be joint. You may dissolve your group at any time during the quarter, after which you can do the project independently with independent grading. The TAs will provide a matchmaking service for interested students.

Homework

I will assign homework problems periodically (on the order of weekly) throughout the quarter. These problems will focus on general and theoretical material in the textbook, complementing the practical but specific material in the project.

Homework problems are to be done independently.

Tests

We will have a midterm and a final. The midterm is tentatively scheduled for May 1, in class. The final exam is scheduled for June 8 at 8:30am (ouch!), in class.

Grading

- project: 40% (in pieces)
- homework: 20% (in pieces)
- midterm: 15%
- final: 25%

Late Policy

Each member of the class has **three** free late days during the quarter. You can turn in any assignment late, as long as you have late days remaining. Turning in on Monday an assignment due Friday counts as two late days. After you run out of late days, late assignments will be assessed a penalty of 25% of the highest possible grade per calendar day late. So use your late days wisely. Late days are your way to cope with unexpected conflicts, sicknesses, etc.

Tentative Course Outline

week	Monday	Wednesday	Friday
3/27	*Overview of Compilers*	*Lexical Analysis (Scanning)*	**[Compiled PL/0 Programs Due]**
4/3		*Syntactic Analysis (Parsing)*	
4/10	**[Scanner Project Due]**		**[EBNF Project Due]**
4/17	*Semantic Analysis (Symbol Tables, Type Checking)*	**[AST Project Due]**	
4/24	**[Parser Project Due]**		Midterm Review
5/1	**MIDTERM**	*Run-Time Environments*	
5/8	**[Type-Checker Project Due]**		*Intermediate Code Generation*
5/15	**[Storage Layout Project Due]**		*Target Code Generation*
5/22			*Optimizations*
5/29	**MEMORIAL DAY**		Final Review **[Code Generator Project Due]**

Project Description

The course project is to add the following language constructs to PL/0:

- comments ("#" to end of line)
- allow "_" in identifiers
- `if`/`then`/`else` statements
- simple `for` loops, plus `break` statements
- procedures with results, plus `return` statements
- call-by-reference parameter passing
- arrays and array indexing
- booleans as first-class data types, with `true` and `false` constants and `and` and `or` operators
- expressions allowed as constants

Each language addition will require language design and specification work (extending the initial PL/0 language specification to include the new features) as well as implementation work (extending the initial C++ implementation of PL/0 to compile the new features).

Emphasis in the implementation should be placed on making clean, understandable changes that fit in well to the existing compiler structure. This will make reading the changed code easier, both for me, the TA, and you. Gross, low-level hacks, even for better compiler efficiency, will not be appreciated.

You are encouraged to organize yourselves into two-person teams to work on the project. When turning in a part of the project, simply list the names of the members of the project team. The grade for that part of the project will be assigned to all members of the team. Teams can be dissolved at any point, after which the two people will do independent work and receive independent grades. You are not required to be on a team, but you are likely to learn more and work less by being on a team.

All implementation work for this class should be done on `ceylon`, `fiji`, `sumatra`, or `tahiti`. The C++ source for the initial version of the PL/0 compiler is in the directory `/cse/courses/cse401/00sp/pl0`.

Project Due Dates (tentative):

3/31: get PL/0 compiler built, write and compile PL/0 programs

4/10: extend RE description of tokens of PL/0; extend lexical analysis of compiler

4/14: extend EBNF description of syntax of PL/0

4/19: extend AST implementation of compiler

4/24: extend syntax analysis of compiler

5/8: extend type-checker of compiler

5/15: extend run-time storage allocation of compiler

6/2: extend code generator of compiler