

Languages and Compilers

Introduction

Adam Sampson
School of Design and Informatics
Abertay University

Learning outcomes

- “By the end of the module you should be able to:
 - 1. compare and contrast **programming languages** and paradigms
 - 2. explain the execution-time **memory and type operation** of object oriented software
 - 3. interpret language **BNF/EBNF specifications**, defining BNF for small languages
 - 4. explain and exemplify the functional and information **architecture of a compiler**
 - 5. design and implement compilers for small languages using the **recursive-descent methodology**”

Why's this useful?

- You **might** some day end up writing a compiler...
- ... but it's much more likely that you'll:
 - Write a specification or parser for some network protocol or file format
 - Learn a programming language – or some subtle aspect of one – by reading its specification
 - Design a domain-specific language (DSL)
 - Select an appropriate language for a particular programming problem
- In this module, we'll learn to think about **language design and implementation**

A caveat

- I'll take a fairly traditional approach to the theoretical side of this module...
- ... but this is an **introduction** to an area in which there has always been a lot of active research
- There are better (easier, safer) ways of doing some of the things we'll look at – I'll mention these as they come up

Lectures

- In most weeks, there'll be two lectures, both of which you need to attend
- Roughly two groups of content...
- How compilers work (drawing on Allan Milne):
 - Grammars; lexing; parsing; semantic analysis; code generation
- Programming language design issues:
 - History; formal specifications of syntax and semantics; type systems, typechecking and type inference; dynamic languages; functional languages and lazy evaluation; concurrency

Practical work

- Most weeks have a practical exercise
 - You should be spending 2-3 hours per week on this
- Timetabled lab session for support, but please have a go at the exercise **before** the lab so you know what questions you want to ask!
- Reading papers
- Programming (compilers and new languages)
- Compiler tech (closely related to coursework)

Assessment unit 1

- 50% software project – due Tue 5th May
 - Implement a compiler frontend for a simple programming language; I'll provide the spec
 - **Ask me** if anything's unclear about the spec
 - You write the **parser** and **typechecker**
 - If you've worked through the practicals, you will already have seen how to do this
 - Marked by testing your code against my collection of sample programs, and a short report
 - I encourage the class to **work together** on your own shared test suite – set up a repository for this! (This worked really well for the last 3 years.)

Assessment unit 2

- 50% exam
 - Tests your understanding of the theoretical material we've covered in the lectures
 - Covers **all** the material that we've covered in lectures, practicals and reading – compiler technology, language design and implementation
 - See the past papers to get a feel for the kinds of questions I'm likely to ask – 4 years of CMP409 (and CE1021A before that with less content)

Support

- Learning resources are on MyLearningSpace
 - Slides, recordings, exercises, software
 - Frequently asked questions
 - Reading list – no mandatory books!
- Contact me if you've got questions
 - Best approach: ask in lab sessions
 - By email: a.sampson@abertay.ac.uk
 - **Please include CMP409 in the Subject, so I know which module you're asking about!**

Attendance monitoring

- In 2018-9, there was a **very** strong correlation between not coming to the lectures and not passing the module... (<50% attendance = fail!)
- I'll be tracking attendance in lectures and labs
- You must sign in either with the web site, the mobile app, or by ticking the paper register
 - Any of these are fine with me
- If you can't attend a lecture, you **must** let me know as soon as possible

Language classification exercise

Shell
Script.

Interpreted

Python Ruby
PHP Perl

Virtual BASIC

SQL: JavaScript

Lua R Matlab

Dynamic

Swift
Rdgy

Modula

Java
C++

C
C++
Pascal
Fortran

Functional

Static

Scala
Haskell

Squirrel
Caml
Scheme

Scripting

Procedural

Forgiving/Easy

Dynamic types
Friendly syntax

"not so"
Bedwatter

UnTyped

Assembler

BCPL

Compiled

What does “compiler” mean anyway?

“To compile means to compose out of materials from other documents. Therefore, the **compiler method of automatic programming** consists of assembling and organizing a program from programs or routines or in general from sequences of computer code which have been made up previously.”



– Grace Hopper,
UNIVAC, 1951

Any questions?

- In 2015, the BBC did a series of 15-minute radio programmes about the history of programming languages (Fortran, COBOL, BASIC, Java) – not very technical, but worth a listen
 - Podcast downloads here:
<https://www.bbc.co.uk/programmes/b05pmpf5>
- Definitely technical: Donald Knuth, “The early development of programming languages”
http://bitsavers.org/pdf/stanford/cs_techReports/STAN-CS-76-562_EarlyDevelPgmgLang_Aug76.pdf
 - 1952 Autocode: Alick Glennie, Fort Halstead
- Next lecture: the structure of a compiler