Concepts of Modern Programming Languages

Peter H. Fröhlich
phf@cs.jhu.edu

Munich University of Applied Sciences
Department of Computer Science and Mathematics

1 Course Goals

We will study a variety of modern (but not necessarily recent!) programming languages to broaden our horizons beyond Java and friends. Our focus will be on concepts first, particular languages second: Instead of covering a language "in depth" we will cover "just enough of it" to develop reading comprehension in service of conceptual understanding. However, we will also write small programs in each language to round out our experience. We will pay particular attention to issues such as scoping and parameter passing, modularity and abstract data types, type systems and type checking, parametric and subtype polymorphism, as well as abstractions for concurrent programming.

Prerequisites: Intermediate programming skills, familiarity with an object-oriented programming language.

Requirements: A laptop running a recent Debian or Ubuntu install, possibly as a virtual machine.

2 Text Books

There are no required text books to purchase but of course you're welcome to buy a text book anyway. Here are a few suggestions (any edition will do):

1. Robert W. Sebesta: *Concepts of Programming Languages*. Pearson, 11th edition, 2015.
2. Michael L. Scott: *Programming Language Pragmatics*. Morgan Kaufmann, 4th edition, 2015.
3. Daniel P. Friedman, Mitchell Wand: *Essentials of Programming Languages*. The MIT Press, 3rd edition, 2008.

Instead of a printed text book we're going to use freely available online resources. Links will be provided in class.

3 Tentative Outline

The plan (!) is to cover the following programming languages in the order given, focussing on the highlighted concepts in particular.

1. Lambda Calculus: Programming as Pure Mathematics.
2. Lisp and Scheme: Implementing the Lambda Calculus.
3. Standard ML and OCaml: Pattern Matching, Type Inference, Parametric Polymorphism, Modules.
4. Io: Prototypical Objects, Actors and Futures, literally *everything* is an object.
5. Google's Go: Systems Programming, Subtype Polymorphism, Concurrency.
6. Python, Jython, Cython: Pragmatic Scripting across Multiple Paradigms.

For each language we'll cover the most important features in lecture and then apply what we learned in lab, both by modifying example code and writing small programs independently. Lab exercises (which may have to be finished as homework after lab) will be graded as your *Studienarbeit* and determine 60% of your final grade.

4 More Languages

Here are a number of languages (and an interesting focus area for each) that can serve as the basis for your presentation on the last day of classes:

1. Clojure (software transactional memory)
2. Erlang (fault tolerance)
3. Forth (embedded systems, firmware)
4. Haskell (lazy evaluation)
5. JavaScript (everything web)
6. Julia (multiple dispatch)
7. Lua (embedded scripting)
8. Nim (hygenic macros)
9. Plankalkül (the first "real" programming language?)
10. Prolog (logic programming)
11. SQL (relational database queries)

Presentations are 15 minutes, 15 slides each and can be done in pairs (provided each student talks for about half the time). Please practice and time (!) your presentations in advance to ensure that things go smoothly. (This is especially important if you plan on live coding.) Presentations will be graded as your *Referat* and determine 40% of your final grade.

5 Other Notes

I live and teach in Baltimore, MD for most of the year, so I am not offering this course in English "as an option" or some such thing: I'll *actually* speak English 99.5% of the time. Please try to do the same, not just as a courtesy to me but to actually practice your English. Hey, it's like getting an English course for free with your Computer Science course.

Which programming language should you learn? Learning a new programming language is a big investment in time, energy, and brainpower.Â I have huge respect in mainstream programming languages. But here I will give you a list of modern programming languages that can improve your productivity, boost your career, and make you a better developer. Also, I will cover a wide variety of domains: system programming, app development, web development, scientific computing. The term â€œModern programming languageâ€ is ambiguous. Many consider languages like Python, JavaScript as modern programming languages. At the same time, they consider Java as an Old programming language. In reality, all of them appeared around the same time: 1995. Since the advent of modern computers, hundreds of â€œhigh-levelâ€ programming languages have been developed. Since the earliest (Fortran â€“ 1957), many different directions have been taken, depending on whether the language designers wished to emphasize features, speed, error handling, pedagogy, theories of computation, correctness, or simplicity. At least since the invention of BASIC (â€œBeginnerâ€™s All-purpose Symbolic Instruction Codeâ€), there have been computer programming languages which have catered to the needs of learnersâ€”few data types, automatic conversion, free form statements, case insens C++ probably is the only modern programming language still supporting the goto statement. In the past, it was the leading cause of bugs, yet C++ has decided to put this dreadful relic of the past in. After all, more features is always better, right?Â Pretty much every modern programming language has generics in one form or another (including the dreaded C#/Java, and even C++ has templates). Generics allow the developer to reuse function implementations for different types. Without generics youâ€™d have to implement the add function separately for integers, for doubles, and for floats, resulting in a lot of code duplication. This item:Concepts of Programming Languages (11th Edition) by Robert Sebesta Hardcover $138.66. Only 12 left in stock (more on the way). Ships from and sold by Amazon.com.Â His professional interests are the design and evaluation of programming languages and Web programming. Tell the Publisher! I'd like to read this book on Kindle. To some people learning a new programming language is an excess best avoided. Why, oh why, they argue, do we need yet another language? Don't we have.Â The overall concept is too complicated to explain in a few words, so feel free to check out the official docs to learn more. The point is, this results in 100% memory safety without the need of a garbage collector, which is a big deal. Rust has taken the system programming world by a storm. Itâ€™s already supported on some platforms, powers browsers and rendering engines are quickly replacing C/C++ code on production systems, and is being used to write operating systems. Sure, itâ€™s not everyoneâ€™s cup of tea to create another browser or device driver, but Rust is already spreading itself to other