Programming Language Concepts

Dr. Imran

Lecture 1

Programming Language Concepts

COURSE AIMS

- The key aims of this course are to:
  - Understand the various concepts that arise in different programming languages.
  - Use these concepts in a practical way to help you understand new programming languages.
- To achieve these we'll look at a variety of concepts from how languages are built to type systems to concurrency.

Programming Language Concepts

COURSE ADMINISTRATION

- Three lecture sessions every week.
- Worksheet of exercises every week.
- Two hours lab session every week.
- You are expected to attend all of the lectures and attempt all of the exercises each week.
- To get the most out of the labs, come prepared with questions about the topics you found difficult or confusing.

Programming Language Concepts

COURSE ADMINISTRATION

- The module is assessed by a mixture of coursework and examination.

- There will be following assessments.
  - **Coursework:** worth 40% of module total
  - **Examination:** worth 60% of module total

- Instructions for the coursework and its submission will be posted on the blackboard.
- The coursework will be based on different programming language design and implementation concepts. You will also need to write a report on your language design.
- The remaining 60% of the module total is provided by the examination at the end of semester 2.

Programming Language Concepts

TOPICS TO BE COVERED

- Introduction to Programming Language Concepts
- From Syntax to Execution - lexing and parsing
- Type Systems
- Semantics of Programming Languages
- Concurrency in Programming Languages
- Reasoning about concurrency
- A look at modern programming languages

- The module doesn’t follow any particular textbook but “Types and Programming Languages” by Pierce.

Programming Language Concepts

Why study programming languages?

Increased capacity for expressing ideas

- Different concepts of programming languages will cause you to solve problems in different ways.
- On the other hand, people find it more difficult to conceptualize structures for which there is no explicit language support.

Be able to choose the right language for the job

- Different languages have different strengths and weaknesses.

Increased ability to learn new languages

- technology continues to change
- knowing a wide range of languages will make it easier to master languages of the future

Programming Language Concepts

Why study programming languages?

Better use of already known languages

- understand parts of language that were mysterious
- use techniques from other language in another

Application Domains

Scientific Computing

- Fortran historically used for mathematical applications
- Mathematica, MatLab

Business Applications

- COBOL is still used - since 1959.
- more recently, languages for business processes: BPEL, web services, etc.

Programming Language Concepts

Application Domains

Artificial Intelligence

- LISP and Scheme (functional languages)
- logic programming (e.g. Prolog)
- Machine Learning in Python, R, Julia

Systems

- C / C++, Rust

Web

- markup: HTML, XHTML, CSS
- scripting: Perl, PHP, Ruby, Javascript, …

Programming Language Concepts

Major Programming Language Families

Imperative Languages

- C, C++, Java, C#, Pascal, Rust, Go, Swift

Declarative Languages

- Functional Programming
  - OCaml, Haskell, ML, Lisp, Scheme, F#, Clojure
- Logic Programming
  - Prolog and its variants
- Domain Specific
  - HTML, Ant, SQL, XSLT, SOAP, etc ..

a. i.e. for Java, add sum of all even numbers (step by step)
vs
Haskell add all even numbers (use 'filter, even')

Programming Language Concepts

b. i,e. Java creates & changes lists step by step
vs
Haskell defines functions & applies them to lists ONCE (i.e. map)

Major Programming Language Families

Difference between Imperative and Declarative Languages

Imperative Programming	Declarative Programming
Describes how to achieve a result.	Describes the result desired
Emphasis on changing program state through steps.	Avoids explicit state changes, often stateless.
Its main goal is to describe how to get it or accomplish it.	Its main goal is to describe the desired result without direct dictation on how to get it.
Its advantages include ease to learn and read, the notional model is simple to understand, etc.	Its advantages include effective code, which can be applied by using ways, easy extension, high level of abstraction, etc.
Its type includes procedural programming, object-oriented programming, parallel processing approach.	Its type includes logic programming and functional programming.

Programming Language Concepts

Software Design Methodologies

Data Oriented

- Abstract Data Types (ADTs)
- Object-oriented design: encapsulates data together with code, concepts of class, object, inheritance, ...

Procedure Oriented

- Emphasizes decomposing code into logically independent actions, often emphasized in concurrent programming.

Programming Language Concepts

Evaluation Criteria

A language ought to be:

- easy to write programs in,
- result in readable code,
- help the programmer to avoid bugs,
- provide an appropriate level of abstraction,
- make the code run fast, ...

Programming Language Concepts

A brief history of programming languages

Fortran (1957)

- Designed by John Backus as an alternative to assembly language for the IBM704
- The first high-level programming language!
- Aimed at scientific programming (FORTRAN = Formula Translating System)
- Extremely primitive type system

LISP (1958)

- Developed by John McCarthy at MIT
- First functional programming language
- Dominated in AI applications, particularly in the US
- Scheme (a LISP dialect) is still popular

Programming Language Concepts

A brief history of programming languages

ALGOL (1958)

- Designed by a committee of US and European computer scientists
- ALGOL 60 was the most influential dialect (first implementation by Edsger Dijkstra)
- Led to Backus-Naur Form, procedural design, orthogonal design of language features
- Influenced Pascal, Ada, Modula, C, Simula, Java, etc.

COBOL (1959)

- Designed by CODASYL (Committee of Data Systems Languages)
- Based in part on Grace Hopper's FLOW-MATIC language (developed for the UNIVAC I)
- Aimed at business applications – in widespread use from the 1960s onwards
- Emphasizes data processing as opposed to control flow.

(other 1960s languages: SNOBOL, Simula, PL/I, MUMPS)

Programming Language Concepts

A brief history of programming languages (1970s)

C (1972)

- Developed by Dennis Ritchie at Bell Labs, used to re-implement the UNIX kernel in 1973

BASIC

- Originally developed in 1964 at Dartmouth College
- Became widespread on microcomputers in the 1970s (i.e. Microsoft BASIC in 1975)

PROLOG (1972)

- Logic programming, based on Horn clauses
- Favoured by European AI developers

Ada (1979)

- Driven by need to simplify software development in defense industry
- Emphasized security and reliability; exception handling

(other 1970s languages: Pascal, Forth, SQL)

Programming Language Concepts

A brief history of programming languages (1980s)

Smalltalk (1980)

- Developed at Xerox Palo Alto Research Center in 1972, first distributed as Smalltalk-80
- Object-oriented, dynamically typed
- Influenced by Simula

Standard ML (1983)

- ML developed in 1973 by Robin Milner at the University of Edinburgh
- Revolutionary type system types everything at compile time!
- Influenced Miranda, OCaml, Haskell and Rust

Objective C (1984)

- Popularised by its use in NeXTStep (which begat OS X/macOS)

C++ (1985)

- Developed by Bjarne Stroustrup at Bell Labs, based on his earlier C with Classes

(other 1980s languages: Erlang, Perl)

Programming Language Concepts

A brief history of programming languages (1990s)

Python (1990)
- General purpose scripting language developed by Guido van Rossum

Haskell (1990)
- Lazy pure functional language developed as a non-proprietary alternative to Miranda

Java (1995)
- Developed by James Gosling at Sun Microsystems
- Virtual machine for portable execution – write once, run anywhere

JavaScript (1995)
- Client-side Web scripting language developed by Brendan Eich of Netscape

PHP (1995)
- Server-side Web scripting language
- Interleaves HTML and PHP code

Programming Language Concepts

A brief history of programming languages (2000 and beyond)

C# (2000)
- Developed by Microsoft, largely as a response to Java

F# (2005)
- Developed by Microsoft Research in Cambridge, functional language targeting the C# CLR

Scratch (2007)
- Visual programming language for education, developed by the MIT Media Lab from 2003

Go (2009)
- C-like language developed at Google, designed for memory safety and concurrency

Clojure (2007)
- Lisp on the Java virtual machine

Rust (2010)
- C++-like language developed at Mozilla, designed for memory safety and concurrency

Programming Language Concepts

What's next for programming languages (2000 and beyond)

- Big Data computing (e.g. map/reduce) with language support for this (Google Go)
- Component-based programming and frameworks - bridging the gap between software and hardware design
- Libraries are hugely important, but due a rethink - scalability, orthogonality issues
- Concurrency is a major challenge to make it compatible with other language features
- Types for reliable, safe and secure code are becoming more prominent