Programming Paradigms CSI2120 – Winter 2014

Jochen Lang
EECS, University of Ottawa
Canada

uOttawa
L'Université canadienne
Canada's university

Université d'Ottawa | University of Ottawa

uOttawa.ca

Course Syllabus

- **Complete syllabus at http://www.eecs.uottawa.ca/~jlang/CSI2120/CSI2120.html**
- **Course notes, laboratory exercises and assignments will be made available through Virtual Campus.**
- **Labs and tutorials in two sessions**
  - Make sure to attend your session
  - Attendance at courses of instruction, laboratory periods and discussion groups is mandatory. To be admitted to the final examination in a subject, a student must attend a minimum of 80% of classes.

uOttawa

Summary of Topics

- **An introduction to programming paradigms**
- **Logic Paradigm: Prolog. Syntax and lists, satisfying goals and backtracking, arithmetic, structures, and recursion, input/output, trees and databases, the cut operator, built-ins, grammars and parsing**
- **Functional paradigm: Scheme. Simple expressions, evaluation, arithmetic, lists, conditional expressions, simple recursion, I/O, assignment, vectors, trees, functions as first-class data values**
- **Scripting, dynamic typing paradigms: Python**
- **Other Paradigms: Imperative revisited (Pascal), constraint programming**
- **Other paradigms: Concurrent programming**

uOttawa

Student Evaluation

- **Midterm exam, February 11th (in class) 24 marks**
- **Final exam 40 marks**
- **6 Assignments 30 marks**
- **Lab Quizzes 6 marks (8 Quizzes in total, best 6 count)**

*) If the student's grade in the exam component (midterm and final) is less than 50% then the student's marks for the assignments will be zero.

**) A student who has an official medical certificate (from the University Health Services) for the absence on the day of the midterm will have the final exam mark scaled up accordingly. In those cases the student may not receive 25% of the marks before the drop-date.

uOttawa

Assignments and Academic Fraud

- **Assignments**
  - There will be six programming assignments which must be submitted via Virtual Campus. No other form of submission will be accepted.
  - Late hand-ins will NOT be accepted.
  - All assignments and labs will be posted on Virtual Campus.
- **Academic Fraud and Plagiarism**
  - Any form of plagiarism or fraud on an *assignment* will result in an automatic *zero* for the assignment.
  - For any plagiarism or fraud possible university sanctions still apply.

uOttawa

Recommended Textbooks

- *Allen B. Tucker and Robert E. Noonan, Programming Languages: Principles and Paradigms, McGraw Hill, 2nd ed., 2007.*
- **Maurizio Gabbrielli and Simone Martini, Programming Languages: Principles and Paradigms, Springer, 2010.**
- *William F. Clocksin and Christopher S. Mellish, Programming in Prolog, Springer, 5th ed., 2003.*
- **Patrick Blackburn, Johan Bos, and Kristina Striegnitz, Learn Prolog Now!, College Publications, 2006. on-line .**
- **R. Kent Dybvig, The Scheme Programming Language, MIT Press, 4th ed., 2009. on-line.**
- **Mark Lutz, Learning Python, O'Reilly Media, Inc., 5th ed, 2013.**
- *Kathleen Jensen, Niklaus Wirth, Pascal User Manual and Report: ISO Pascal Standard, 4th ed., Springer, 1991.*

uOttawa

On-Line Resources

- **SWI Prolog Open source prolog compiler available for all major platforms.**
- **MIT/GNU Scheme Full scheme system for Linux, MacOS and Windows.**
- **Free Pascal Open source, modern Pascal compiler available as binary for all major platforms.**
- **Official Python Site Python is available on all major platforms and can even run in Java and .NET virtual machines. Provides the Python tutorial in html.**
- **Go concurrent programming language. Open source with binary distributions for Linux, MacOS and Windows.**

uOttawa

Paradigms and Programming

- **Paradigm:**
  "a theory or a group of ideas about how something should be done, made, or thought about" (Merriam-Webster, accessed Jan, 2014).
- **Computer Programming:**
  " Computer programming … is the comprehensive process that leads from an original formulation of a computing problem to executable programs." (Wikipedia, accessed Jan, 2014).

uOttawa

More Comprehensive View of Programming

- **Programming**

  "… involves activities such as analysis, understanding, and generically solving such problems resulting in an **algorithm**, **verification of requirements** of the algorithm including its correctness and its resource consumption, **implementation** (or coding) of the algorithm in a **target programming language**, **testing**, **debugging**, and **maintaining** the source code, …" (Wikipedia, accessed Jan, 2014).

Programming Language

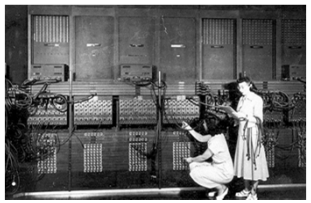
- **A Programming Language**

  "is a formal language designed to communicate instructions to a machine, particularly a computer. Programming languages can be used to create programs that control the behavior of a machine and/or to express algorithms precisely. "(Wikipedia, accessed Jan, 2014).

Programmer (Software Engineer)

- **A Programmer,**

  computer programmer, developer, coder, or software engineer is a person who writes computer software. " (Wikipedia, accessed Jan, 2014).



Source: US Army

Source: German Bundesarchiv

Best Programming Language Implementing the Best Paradigm

- **There is no single best paradigm for all applications.**
- **Problems can be solved based on one or the other paradigm.**
- **Many Paradigms exist:**
  - object-oriented programming,
  - logic programming,
  - functional programming,
  - imperative programming,
  - scripting or dynamic programming,
  - declarative programming,
  - aspect-oriented programming,
  - concurrent programming, and others …

Familiar Programming Paradigms

- **Imperative paradigm**
  - views program as a sequence of commands which change the state of the program
  - Prominent example language (likely) familiar to you: C
- **Object-oriented paradigm**
  - Solution is described by a set of classes providing encapsulation.
  - Class relationships include inheritance and message passing.
  - Polymorphism
  - Prominent example language familiar to you: Java

uOttawa

Logic Programming Paradigm

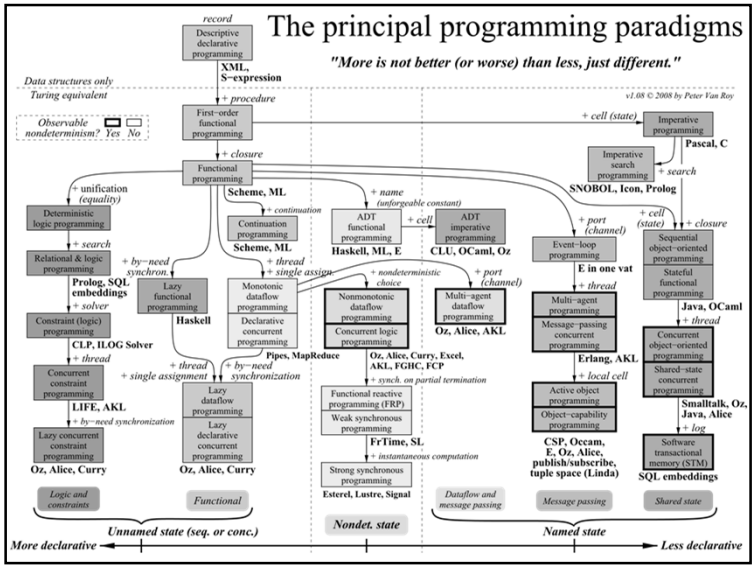
- **Describes the facts and rules about a problem and the desired program outcome and not the individual steps of the solution.**
- **Logic programming paradigm**
  - views programs as a set of constraints,
  - requires programs to produce different (or even all) solutions,
  - and considers programs non-deterministic
- **Prominent example language covered in this course**
  - Prolog

uOttawa

Functional Programming Paradigm

- **Functional programming is inspired by mathematical function as a mapping from an input domain to an output range.**
- **Functional paradigm**
  - compose different function to achieve a solution
  - use often recursive solutions
- **Prominent example language covered in this course:**
  - Scheme

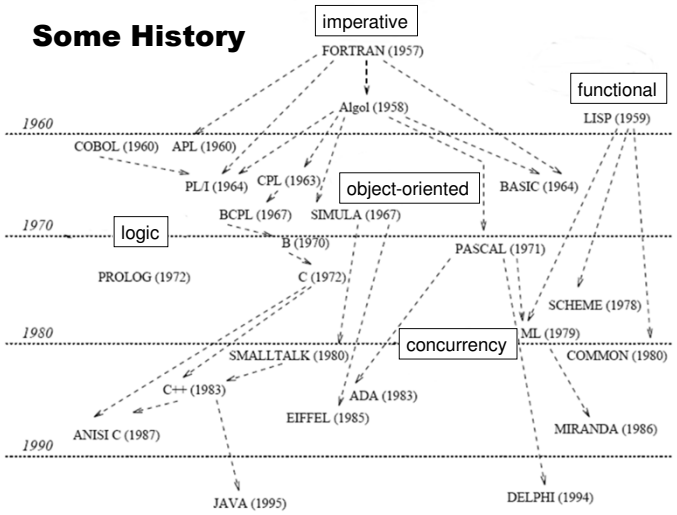
uOttawa



Why study different paradigms?

- Understanding programming at the abstract level will lead to a better overall understanding of programming
- Different problems, different challenges benefit from different approaches
- Knowing many languages helps to learn new languages
- To be able to chose the best paradigm and language for a new task

Some History



Influential Programming Languages

(until 1990, according to Bill Kinnersley)

1957 FORTRAN	1977 OPS5	1988 Oberon
1958 ALGOL	1978 CSP	1989 HTML
1960 LISP	1978 FP	1990 Haskell
1960 COBOL	1980 dBASE II	
1962 APL	1983 Smalltalk-80	
1962 SIMULA	1983 Ada	
1964 BASIC	1983 Parlog	
1964 PL/I	1984 Standard ML	
1966 ISWIM	1986 C++	
1970 Prolog	1986 CLP(R)	
1972 C	1986 Eiffel	
1975 Pascal	1988 CLOS	
1975 Scheme	1988 Mathematica	

Alphabetical High-level Language Overview

- ADA: created at Honeywell Bell for the American DoD, supports concurrency, synchronous message passing
- Algol: ACM, GAMM, close to mathematical notation, machine-independent description of algorithms
- ANSI C: standardized C increasing portability and prototyping
- APL: K. Iverson, mathematical programming (matrices, vectors) but difficult to read
- B: K. Thompson, simple compilation, original language for UNIX
- BASIC: J. Kemeny, T. Kurtz, Simple language, made programming widely accessible to engineering
- C: B.W. Kernigham, D.M. Ritchie, system programming, non-restrictive language, close to machine instructions
- C++: B. Stroustrup, object-oriented C, hybrid language, abstract data types, templates.

Alphabetical High-level Language Overview

- C#: hybrid between C++ and Java, used for Microsoft platform .Net
- COBOL: COmmon Business-Oriented Language (created by Grace Hopper), business, finance, management; verbose to ease general understanding.
- COMMONLISP: LISP dialect that unifies the different LISP dialects, tentative standard.
- CPL: North American equivalent of Algol, but closer to machine instructions.
- Eiffel: B. Meyer, object-oriented language, created to support "design by contract"
- FORTRAN: created by J. Backus for the IBM704, effective coding.
- Java: designed as a "better" C++ with closer adherence to OO paradigm; created by J. Gosling at Sun.

Alphabetical High-level Language Overview

- LISP: J. McCarthy, artificial intelligence, theorem proving, (emacs); based on a single data structure, lists.
- Miranda: D. Turner: first commercial purely functional language, lazy evaluation.
- ML: R. Milner, strongly typed, functional language with inference
- Pascal: functional, originally created by N. Wirth for teaching.
- PL/I: IBM and Share 3by3, universal language, originally developed for mainframe computers.
- Prolog: Created by Alain Colmerauer with Philippe Roussel, mainly expert systems, artificial intelligence and data bases, now ISO standard.
- Scheme: MIT, LISP dialect, originally created for teaching.
- Simula: K. Nygaard, O.J. Dahl, simulation, introduced the notion of a class.
- SmallTalk: A. Kay, first object-oriented language, objects, message passing. Designed for graphical environments (windows).

Conclusion

- Why are there so many similar languages?
- Isn't Java (or C++, or your favorite) a programming language for everything?

➢ Analyze new programming paradigms

➢ Learn new languages

➢ Get an overview of other paradigms.