Programming languages: Introduction and overview

- Languages and the software development process
- Languages and software development environments
- Languages and software design methods
Language paradigms

- **Procedural programming**
  - Programs decomposed into computation steps
  - Routines used as decomposition units

- **Functional programming**
  - Computation of values via expressions + functions
  - Functions are primary building blocks

- **Abstract data type programming**
  - Abstract data types as the unit modularization
Language paradigms

- **Module-based programming**
  - modularization = group of entities (vars, procs, types, etc.) + export interface

- **Object-oriented programming**
  - modularization via class definitions
  - instances created during program execution

- **Generic programming**
  - generic modules to instantiate (compile/run-time)

- **Declarative programming**
  - declarative problem, not algorithm decomposition
Language paradigms: a more orthogonal view

Von Neumann languages
(procedural, operational, state machine-based...)

Non-Von Neumann languages
(go back to the mathematical notion of variable, no assignment, ...)

Functional languages (LISP, ML, ..)

Logic languages (PROLOG, ...)
Abstraction mechanisms:

- High level constructs
- Modularization
- Object-orientation
- ...

- Can be applied to anyone of such languages:
  - From machine languages ...
  - ... to OO LISP (CLOS),
  - ... OO logic languages, ...
• In this course we will mainly focus on von Neumann languages
• With only hints to non von Neumann ones
• These are treated in depth in other courses (ICSE, AI)
• We will refer to a few “sample languages” (C++, Java, Ada, …) with the goal of creating
  – flexibility in learning new languages
  – critical and comparative attitude
Languages and architecture

A von Neumann computer architecture
von Neumann Languages

- also called imperative, statement-based state-based
  - variables as abstraction of cells
  - one-at-a-time instruction execution
  - execution changes state
Requirements and constraints on a language

Programmer’s needs

requirements

Programming language

higher level of abstraction

constraints

von Neumann architecture
Language qualities

Software must be

- reliable
  - writability, readability, simplicity, safety, robustness

- maintainable
  - factoring, locality

- efficiently executable
  - but efficiency of programming is even more important
Historical view

• Early software process: simple programming
  – Late 50s - early 60s
  – FORTRAN: separate compilation, formulas
  – ALGOL60: block structure, recursion, data structure
  – COBOL: Files, I/O

• Early nonconventional languages
  – LISP: lists, simple operations, uniform code/data
  – APL: arrays, lots of operations
  – SNOBOL4: strings, pattern-matching, backtracking
Historical view - 2

• Late 60s
  – Algol 68: purity, orthogonality
  – Simula 67: simulation
  – Pascal: Simplicity
  – BASIC: interactivity

• 70s
  – Experimentation with paradigms
  – Object orientation: Smalltalk, Eiffel, ...
Historical view - 3

– Logic: PROLOG
– Concurrency and modularity: Mesa, Modula-2
– Security: Euclid, Gypsy

• 80s
  – Object-orientation: C++, Ada, Eiffel

• 90s
  – Visual interfaces: Visual basic
  – Parallelism: FORTRAN 90
  – Network programming: Java