



## Selected Overview of Programming Languages

- Fortran (I, II, IV, 77)
  - Dramatic impact on computing in early days
  - Mainly used for numerical computation
  - No recursion
  - Limited data types (no records and no pointers)
  - Limited type checking
  - Very good compilers are available today
- Fortran (90, 95, HPF)
  - Major revisions, eg. recursion, pointers, and records added
  - New control constructs (eg. `while` loop)
  - Extensive set of array operations
  - HPF (High-Performance Fortran) includes parallel constructs
- Lisp
  - The original functional language developed by McCarthy as a realization of an abstract machine: Church's lambda calculus
  - Many dialects, including Common Lisp and Scheme
  - Very powerful for symbolic computation using lists (eg. for artificial intelligence)
  - Implicit memory management (allocate/deallocate) by "garbage collection"
  - Influenced functional programming languages (ML, Miranda, Haskell)

scopes)

- First use of Backus-Naur Form (BNF) to formally define grammar
- All subsequent imperative programming languages are based on it
- No I/O and no character set, not widely used in US
- Algol 68
  - Large and relatively complex
  - Strong influence on Pascal, C, Ada
- Cobol
  - Originally developed by Department of Defense
  - Intended for business data processing
  - Extensive numerical formatting features and decimal number storage
  - Introduced the concept of records and nested selection statements
- Basic
  - Intended for interactive use (intepreted)
  - Goals: easy to learn and use for non-science students
  - Visual Basic is a popular dialect
- PL/I
  - Designed by IBM
  - First exception handling
  - First pointer data type
  - Poorly designed, too large, too complex

## Selected Overview of Programming Languages

- Algol 60
  - The original block-structured language (local variable

## Selected Overview of Programming Languages

- Pascal
  - Designed for teaching "structured programming"
  - Small and simple
- Simula 67
  - Primarily designed for discrete-event simulation
  - Based on Algol 60
  - Introduced concept of coroutines (co-executing routines, kind of communicating threads)
  - Introduced the `class` concept for data abstraction
- Ada
  - Originally intended to be the standard language for all software commissioned by the Department of Defense
  - Very large
  - Elaborate support for packages, exception handling, generic program units, concurrency
- Ada 95
  - Support for object-oriented programming
  - New concurrency features

## Selected Overview of Programming Languages

- Smalltalk-80
  - Developed by XEROX PARC
  - First full implementation of an object-oriented language
  - First design and use of window-based graphical user interfaces (GUIs)
- APL
  - Intended for interactive use ("throw-away" programming)
  - Highly expressive functional language makes programs short, but hard to read
  - Many array operations
- Prolog
  - The most widely used logic programming language
  - Non-procedural (declarative: states what you want, not how to get it)
  - Based on formal logic
- Haskell
  - The leading purely functional language, based on Miranda

## Selected Overview of Programming Languages

- **C**
  - One of the most successful programming languages
  - Designed for systems programming
  - Powerful set of operators, but weak type checking and no dynamic semantic checks
- **C++**
  - The most successful of several object-oriented successors of C
  - Evolved from C and Simula 67
  - Large and complex, because it supports both procedural and object-oriented programming
- **Java**
  - Developed by Sun
  - Based on C++, but significantly simplified
  - Supports only object-oriented programming
  - Safe language (eg. no pointers but references, strongly typed, and implicit garbage collection)
  - Truly machine-independent (?)

Note: More about Java

## So Why is it That There are so Many Programming Languages?

- **Evolution**
  - This course should give you some insight in what constitutes a good or a bad programming construct for language design. Appendix B of the textbook has a long list of historical mistakes
  - Early 70s: "structured programming" in which `goto`-based control flow was replaced by high-level constructs such as `while` loops and `case` statements
  - Late 80s: nested block structure gave way to object-oriented structures
- **Special Purposes**
  - Many languages were designed for a specific problem domain. For example
    - Scientific applications
    - Business applications
    - Artificial intelligence
    - Systems programming
- **Personal Preference**
  - The strength and variety of personal preference makes it unlikely that anyone will ever develop a universally acceptable programming language

## What Makes a Programming Language Successful?

- **Expressive Power**
  - All languages are equally powerful in technical sense
  - Language features have a huge impact on the programmer's ability to read, write, maintain, and analyze programs
  - Abstraction facilities enhance expressive power
- **Ease of Use for Novice**
  - Low learning curve and often interpreted, eg. Basic and Logo
- **Ease of Implementation**
  - Runs on virtually everything, eg. Basic, Pascal, and Java
  - Freely available
- **Excellent Compilers**
  - Fortran has extremely good compilers (because it lacks recursion and pointers) and is therefore popular for numerical processing
  - Supporting tools to help the programmer manage very large projects, eg. Visual C++
- **Economics, Patronage, and Inertia**
  - Powerful sponsor: Cobol, PL/I, Ada
  - Some languages remain widely used long after "better" alternatives because of a huge base of installed software and programmer experience

## Classification of Programming Languages

- **Declarative** ("what the computer is to do")
  - Functional (eg. Lisp, Scheme, ML, Miranda, Haskell)
  - Dataflow
  - Logic (eg. Prolog, Excel)
- **Imperative** ("how the computer should do it")
  - Procedural or "von Neumann" (eg. Fortran, Pascal, Basic, C)
  - Object-oriented (eg. Smalltalk-80, C++, Java)

