CSE 325/CSE 425: Concepts of Programming Language

Evolution of Programming Languages

Dr. Kamruddin Nur

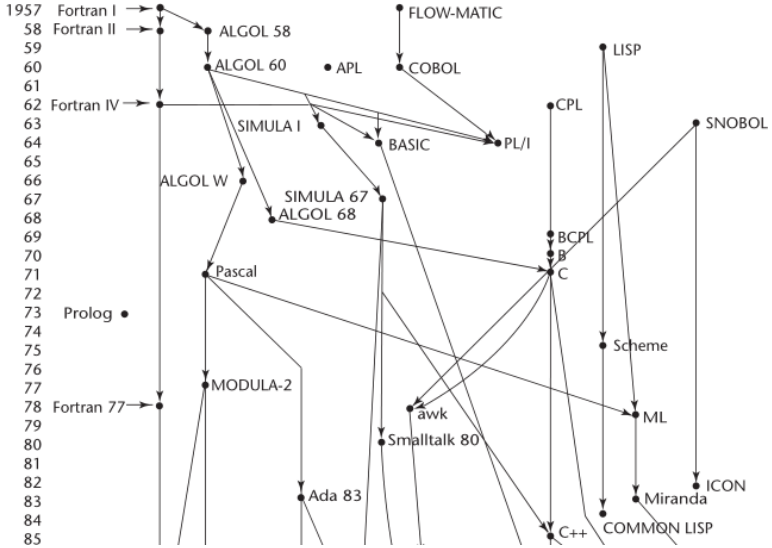
Adjunct Associate Professor, NSU

kamruddin.nur@northsouth.edu

January, 2018

Contents

Genealogy of Common Languages



Genealogy of Common Languages *(Cont'd)*

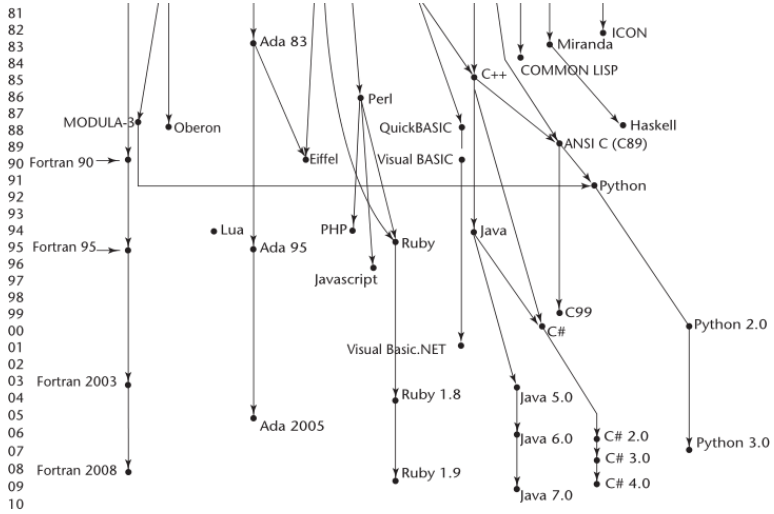


Figure 1: **Genealogy (generation / family history) of common high-level programming languages**
(Figure source: Concepts of Programming Languages by Robert W. Sebesta, 10th Edition, Pearson, p. 37)

2 Major Programming Languages
- Fortran
- LISP
- ALGOL 60
- COBOL
- BASIC
- PL/I
- APL - A Programming Language
- SNOBOL
- SIMULA 67
- Others

Fortran

Fortran is a general-purpose, imperative programming language suitable for numeric and scientific computing.

- **Fortran I: 1957**
  - Designed for the IBM 704, which had index registers and floating point hardware
  - This led to the idea of compiled programming languages, because there was no place to hide the cost of interpretation (no floating-point software)
  - Environment of development
    - Computers were small and unreliable
    - Applications were scientific
    - No programming methodology or tools
    - Machine efficiency was the most important concern
  - Impact of environment on design of Fortran I
    - No need for dynamic storage
    - Need good array handling and counting loops
    - No string handling, decimal arithmetic, or powerful input/output (for business software)

Fortran *(Cont'd)*

- First implemented version of Fortran
  - Names could have up to six characters
  - Post-test counting loop (DO)
  - Formatted I/O
  - User-defined subprograms
  - Three-way selection statement (arithmetic IF)
  - No data typing statements
  - No separate compilation
  - Compiler released in April 1957, after 18 worker-years of effort!
  - Programs larger than 400 lines rarely compiled correctly, mainly due to poor reliability of 704
  - Code was very fast
  - Quickly became widely used

- **Fortran II: 1958**
  - Independent compilation
  - Fixed the bugs of Fortran I
- **Fortran III**
  - Was developed but never widely distributed!

Fortran *(Cont'd)*

- **Fortran IV: 1960-62**
  - Explicit type declarations
  - Logical selection statement
  - Subprogram names could be parameters
  - ANSI standard in 1966
- **Fortran 77: 1978**
  - Became the new standard
  - Character string handling
  - Logical loop control statement
  - IF-THEN-ELSE statement
- **Fortran 90: 1990**
  - Most significant changes from Fortran 77
    - Modules
    - Dynamic arrays
    - Pointers
    - Recursion
    - CASE statement

Fortran *(Cont'd)*

- Parameter type checking

- **Fortran 95**
  - relatively minor additions, plus some deletions
- **Fortran 2003**
  - support for OOP, procedure pointers, interoperability with C
- **Fortran 2008**
  - blocks for local scopes, co-arrays, Do Concurrent

Dramatically changed forever the way computers are used!

Functional Programming: LISP

- The name LISP derives from "LISt Processor"
- Designed at MIT by McCarthy
- AI research needed a language to Process data in lists (rather than arrays) and Symbolic computation (rather than numeric)
- Only two data types: atoms and lists
- Syntax is based on lambda calculus
- Pioneered functional programming
  - No need for variables or assignment
  - Control via recursion and conditional expressions
- Dialects: **Scheme** (simple syntax), **COMMON LISP** (complex syntax, used in industry for some large applications)
- Still the dominant language for AI
- Other Functional Language: **ML** ('Meta Language'), **Haskell**, and **F#**

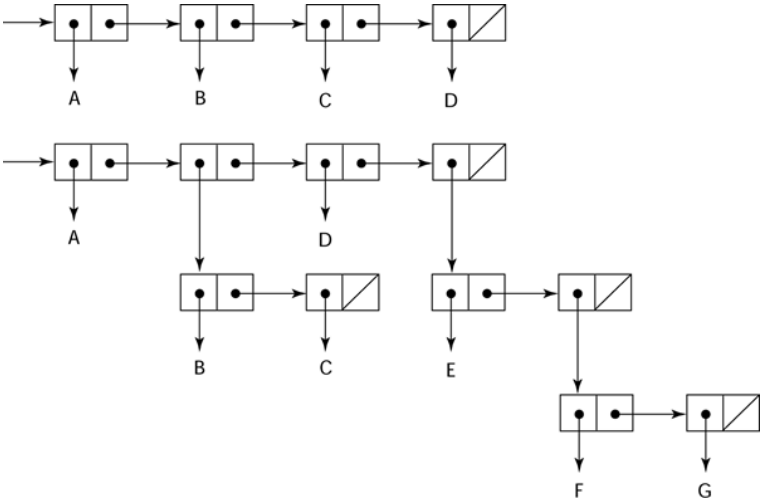


Figure 2: **LISP representing the lists (A B C D) and (A (B C) D (E (F G)))**

ALGOL 60

- Context: Why ALGOL 60?
  - FORTRAN had (barely) arrived for IBM 70x
  - Many other languages were being developed, all for specific machines
  - No portable language; all were machine-dependent
  - No universal language for communicating **algorithms**
- Goal:
  - Close to mathematical notation
  - Good for describing algorithms
  - Must be translatable to machine code
- Features of ALGOL 60:
  - Block structure (local scope)
  - Two parameter passing methods
  - Subprogram recursion
  - Stack-dynamic arrays
- Lacking of ALGOL 60:
  - No I/O and no string handling

ALGOL 60 *(Cont'd)*

- Success:
  - It was the standard way to publish algorithms for over 20 years
  - All subsequent imperative languages are based on it
  - First machine-independent language
- Failure:
  - Never widely used, especially in U.S.
  - Reasons include - Lack of I/O and the character set made programs non-portable

COBOL

- Context: Why COBOL?
  - Design Goals: Design meeting at Pentagon - May 1959
    - Must look like simple English
    - Must be easy to use, even if that means it will be less powerful
    - Must broaden the base of computer users
    - Must not be biased by current compiler problems
- Contributions:
  - First macro facility in a high-level language
  - Hierarchical data structures (records)
  - Nested selection statements
  - Long names (up to 30 characters), with hyphens
  - Separate data division

BASIC

- Designed by Kemeny & Kurtz at Dartmouth
- Design Goals:
  - Easy to learn and use for non-science students
  - Must be “pleasant and friendly”
  - Fast turnaround for homework
  - Free and private access
  - User time is more important than computer time
- Current popular dialect: Visual BASIC
- First widely used language

PL/I

PL/I is a procedural, imperative programming language designed for scientific, engineering, business and system programming.

- Designed by IBM
- Background: By 1963,
  - Scientific users began to need more elaborate I/O, like COBOL had; business users began to need floating point and arrays for MIS
  - It looked like many shops would begin to need two kinds of computers, languages, and support staff too costly
- Computing situation in 1964 (IBM's point of view)
  - Scientific computing
    - IBM 1620 and 7090 computers
    - FORTRAN
    - SHARE user group
  - Business computing
    - IBM 1401, 7080 computers
    - COBOL

PL/I *(Cont'd)*

- GUIDE user group

- The obvious solution:
  - Build a new computer to do both kinds of applications
  - Design a new language to do both kinds of applications
- Contribution:
  - First unit-level concurrency
  - First exception handling
  - Switch-selectable recursion
  - First pointer data type
  - First array cross sections
- Concerns:
  - Many new features were poorly designed
  - Too large and too complex

APL

- Designed as a hardware description language at IBM by Ken Iverson around 1960
  - Highly expressive (many operators, for both scalars and arrays of various dimensions)
  - Programs are very difficult to read
- Still in use; minimal changes

SNOBOL

- Designed as a string manipulation language at Bell Labs by Farber, Griswold, and Polensky in 1964
- Powerful operators for string pattern matching
- Slower than alternative languages (and thus no longer used for writing editors)
- Still used for certain text processing tasks

SIMULA 67

- Designed primarily for system simulation in Norway by Nygaard and Dahl
- Based on ALGOL 60 and SIMULA I
- Primary Contributions:
  - Coroutines - a kind of subprogram
  - Classes, objects, and inheritance
- **The Beginning of Data Abstraction!!**

Others

- Pascal - 1971
- C - 1972

Thanks

Thanks for your time and attention!

kamruddin.nur@northsouth.edu
researchgate.net/profile/Kamruddin_Nur