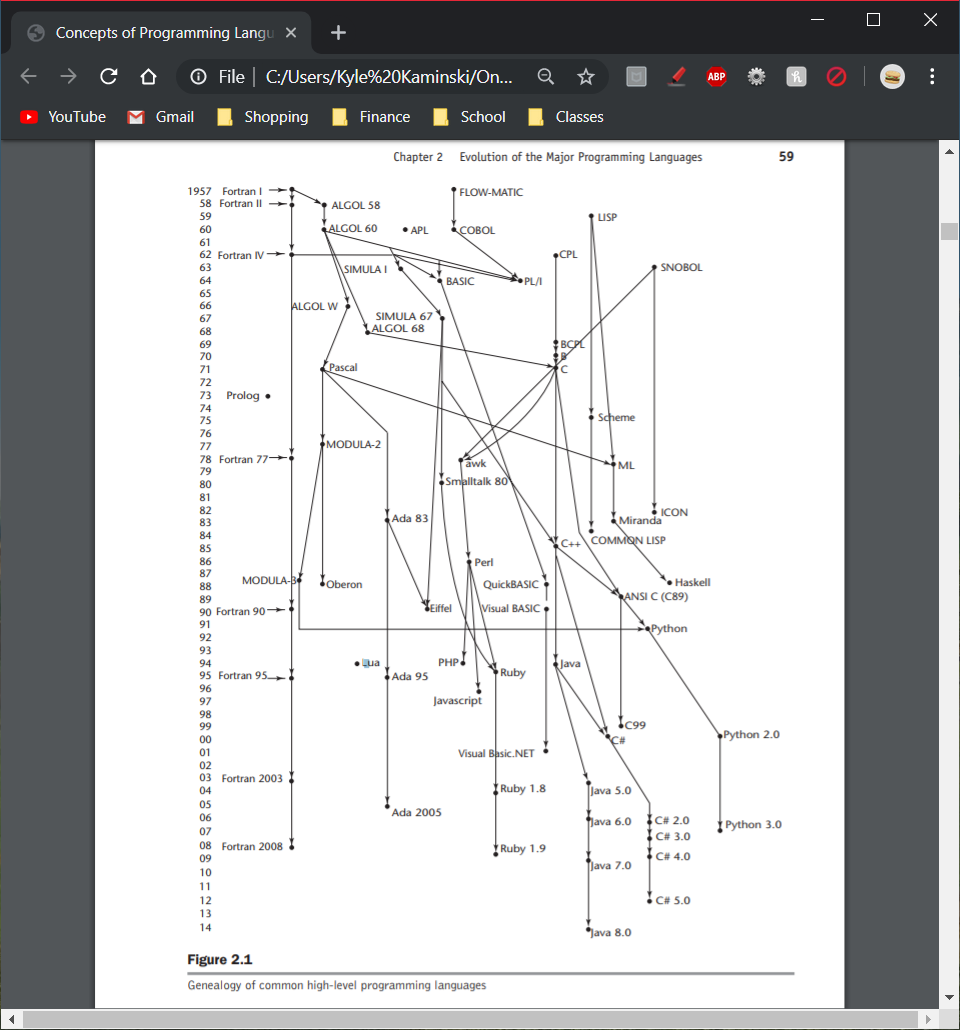
Chapter 2 - Evolution of Major Programming Languages



* Zuse’s Planalkul
  + Never implemented
  + Developed in 1945
  + Historial Background
    - *Program calculus*
    - Used to write algorithms to solve a wide variety of problems
  + Language Overview
    - Used data structures
    - Simplest data type: single bit
    - Included arrays and (nested) records
    - Included an iterative statement like *for*
    - *Fin* command to specify an exit out of a given number of iteration loop nestings
* Pseudocodes
  + Not practical, written in machine code, hard to read
  + Short Code
    - First of these new languages (1949)
    - Translated into machine code with a pure interpreter
  + Speedcoding
    - 1954
    - Speedcoding interpreter effectively converted the 701 to a virtual three-address floating-point calculator
  + The UNIVAC “Compiling” System
    - 1951-1953
    - Expanded a pseudocode into machine code subprograms in the same way as macros are expanded into assembly language
* The IBM 704 and Fortran
  + 1954
  + One of the greatest single advances in computing; its capabilities prompted the development of Fortran (The IBM FORmula TRANslating System)
  + Historical Background
    - Fortran is often credited with being the first compiled high-level language
  + Design Process: the environment in which Fortran was developed was as follows:
    - Computers had small memories and were slow and relatively unreliable
    - Primary use of computers was for scientific computations
    - There were no existing efficient and effective ways to program computers
    - Because of the high cost of computers compared to the cost of programmers, speed of the generated object code was the primary goal of the first Fortran compilers
  + Fortran I Overview
    - 1956
    - Included input/output formatting, variable names of up to 6 characters, user-defined subroutines, *if* statement, and the *do* loop statement
    - No data-typing statements; Integer types must begin with I, J, K, L, M, or N
  + Fortran II
    - 1958
    - Fixed many of the bugs of Fortran I
    - New features: independent compilation of subroutines (previously, any change to the code would require a recompile of the entire thing)
  + Fortran IV, 77, 90, 95, 2003, and 2008
    - Fortran III was never widely distributed
    - Fortran IV was a widely used language during its time
      * Standardized as Fortran 66
      * New features and improvements: explicit type declarations for variables, a logical *if* construct, capability of passing subprograms as parameters to other subprograms
    - Fortran 77 became new standard in 1978
      * Retained most features of Fortran IV and added character string handling, logical loop control statements, and an *if* with an optional *else* clause
    - Fortran 90 additions: dynamic arrays, records, pointers, a multiple selection statement, and modules
    - Fortran 95 new changes: *Forall* was added
    - Fortran 2003 added support for object-oriented programming, parameterized derived types, procedure pointers, and interoperability with the C programming language
    - Fortran 2008 added support for blocks to define local scopes, co-arrays, which provide a parallel execution model, and the *DO CONCURRENT* construct, to specify loops without interdependencies
* Functional Programming: Lisp
  + “LISt Processing”
  + Language Overview
    - Only two kinds of data structures: atoms and lists
      * Atoms: either symbols, which have the form of identifiers, or numeric literals
    - Designed as a functional programming language
    - Lisp is very different because it is both a functional programming language and because the appearance of Lisp programs is so different from those in languages like Java or C++
    - Lisp has completely dominated AI applications
    - Scheme and Common Lisp are two dialects of Lisp that are widely used
      * Scheme is characterized by its small size, its exclusive use of static scoping, and its treatment of functions as first-class entities
      * Scheme functions can be assigned to variables, passed as parameters, and returned as the values of function applications; can also be elements of lists
      * Scheme is well suited to educational applications,
      * Common Lisp was created to unify the different dialects of Lisp being used
* The First Step Toward Sophistication: ALGOL 60
  + Result of efforts to design a universal programming language for scientific operations
  + Early Design Process
    - Syntax should be as close as possible to standard mathematical notation, and programs written in it should be readable with little further explanation
    - Should be possible to use the language for the description of algorithms in printed publications
    - Programs in the new language must be mechanically translatable into machine language
    - ALGOL 58 Overview
      * “ALGOrithmic Language”
      * Descendent of Fortran
      * Formalized the concept of data type, although only variables that were not floating-point required explicit declaration
    - ALGOL 60 Overview
      * Concept of block structure was introduced
      * Two different means of passing parameters to subprograms were allowed: pass by value and pass by name
      * Procedures could be recursive
      * Stack-dynamic arrays were allowed
* Computerizing Business Records: COBOL
  + FLOW-MATIC
  + COBOL Design Process
    - Should have the following characteristics:
      * Should use English as much as possible
      * Must be easy to use
      * Design should not be overly restricted by the problems of its implementation
    - *DEFINE* was the first high-level language construct for macros
* The Beginnings of Timesharing: Basic
  + Basic was easy for beginners to learn, and its smaller dialects could be implemented
  + Goals for the system were as follows:
    - Must be easy for nonscience students to learn and use
    - Must be “pleasant and friendly”
    - Must provide fast turnaround for homework
    - Must allow free and private access
    - Must consider user time more important than computer time
* Everything for Everybody: PL/I
  + “Programming Language I”
  + Represents the first large-scale attempt to design a language that could be used for a broad spectrum of application areas
  + Was developed as an IBM product
  + It included what were then considered the best parts of ALGOL 60, Fortran IV, and COLBOL 60, along with an extensive collection of new constructs
    - ALGOL 60: recursion and block structure
    - Fortran IV: separate compilation with communication through global data
    - COLBOL 60: data structures, input/output, and report-generating facilities
  + First programming language to have the following facilities:
    - Programs could create concurrently executing subprograms (however, was poorly developed in PL/I)
    - Possible to detect and handle 23 different types of exceptions, or run-time errors
    - Subprograms were allowed to be used recursively, but the capability could be disabled, allowing more efficient linkage for non-recursive subprograms
    - Pointers were included as a data type
    - Cross-sections of arrays could be referenced
* Two Early Dynamic Languages: APL and SNOBOL
  + “A Programming Language”
  + APL was intended to be a vehicle for describing computer architecture
  + SNOBOL was designed specifically for processing
* The Beginning of Data Abstraction: SIMULA 67
  + SIMULA I was designed exclusively for system simulation and was first implemented in late 1964
* Orthogonal Design: ALGOL 68
  + Was the source of several new ideas in the language design, some of which were subsequently adopted by other languages
* Some Early Descendants of the ALGOLs
  + Simplicity by Design: Pascal
  + A Portable Systems Language: C
    - Originally designed for systems programming
* Programming Based on Logic: Prolog
  + Consists of collections of statements
  + Can be used as a kind of intelligent database
* History’s Largest Design Effort: Ada
  + The result of the most extensive and expensive language design effort ever undertaken
  + Ada 95 and Ada 2005
    - Ada 95: type derivation mechanism of Ada 83 is extended in Ada 95 to allow adding new components to those inherited for a base class
    - Ada 95: Rendezvous mechanism of Ada 83 provided only a cumbersome and inefficient means of sharing data among concurrent processes
* Object-Oriented Programming: Smalltalk
  + First programming language that fully supported object-oriented programming
* Combining Imperative and Object-Oriented Features: C++
  + Supports both procedural and object-oriented programming
* An Imperative-Based Object-Oriented Language: Java
  + Provides much of the power and flexibility of C++, but in a smaller, simpler, and safer language
  + Does not have pointers
  + Has primitive *Boolean* type
* Scripting Languages
  + Script- a list of commands
  + Perl was originally a combination of *sh* and *awk*
  + JavaScript is most commonly used in web browsers and is interpreted by the browser when the documents are displayed
  + Java and JavaScript are only similar through their syntaxes
  + PHP (“Personal Home Page tools”) was to provide a tool to help track visitors to a web site
  + Python is used for system administration, CGI programming, and other relatively small computing tasks
  + Ruby is a pure object-oriented language; every data value is an object and all operations are via method calls
  + Lua is a scripting language that supports procedural and functional programming with extensibility as one of its primary goals
* The Flagship .NET Language: C#
  + Based on C++ and Java
  + Purpose is to provide a language for component-based software development, specifically for such development in the .NET Framework
* Markup-Programming Hybrid Languages
  + Markup language in which some of the elements can specify programming actions, such as control flow and computation
  + JSP (Java Server Pages)
  + Servlet- an instance of a Java class that resides on and is execute on a Web server system
  + Servlet container- web server process that controls the execution of servlets