CS 313

History of Programming Languages

History of Programming Languages

- Punch cards
  - Jacquard looms
  - Analytical engine (Charles Babbage and Ada Byron Lovelace)
  - US Census data (Herman Hollerith)
- Hand-coded machine language programs

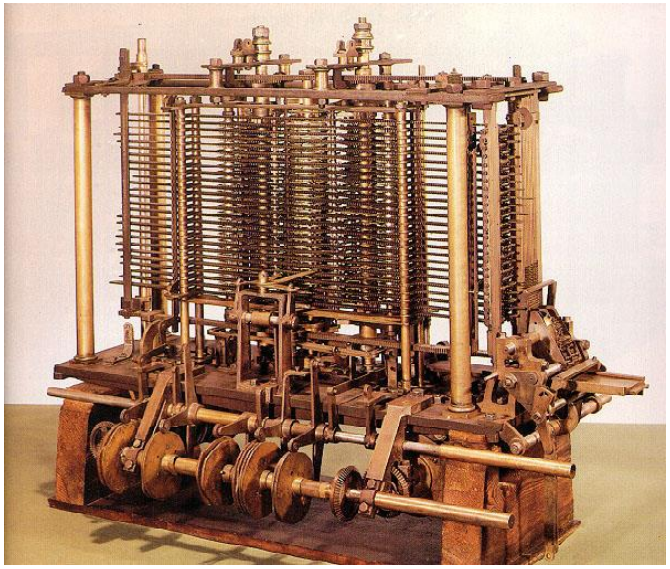
```
10110000 01100001
```

- Assembly language programs

```
movl $3, %eax
```

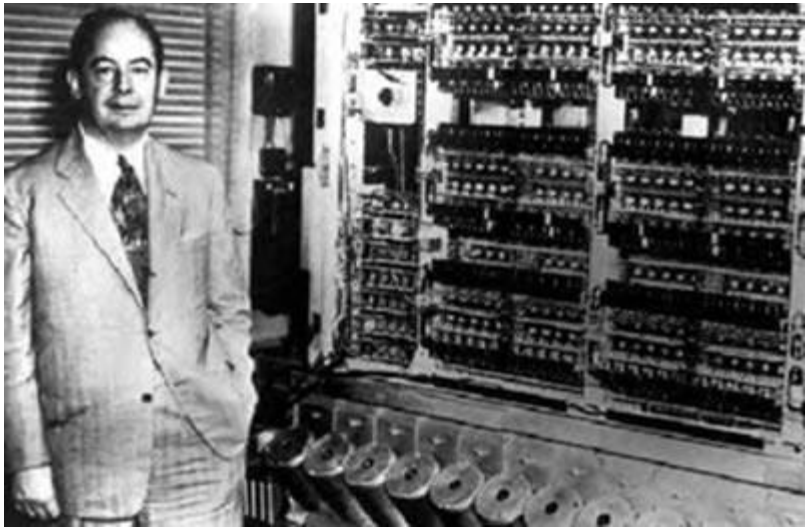
- Modern programming languages

Charles Babbage's Analytic Engine 1834



- Earliest known computer
- Never fully built
- Operations and variables on separate punch cards
- Conditional jumps accomplished mechanically by physically jumping over a band of cards
- Collaborator Lady Ada Byron, Countess of Lovelace.
- Babbage first computer scientist. Ada Byron first computer programmer.

Von Neumann architecture 1945



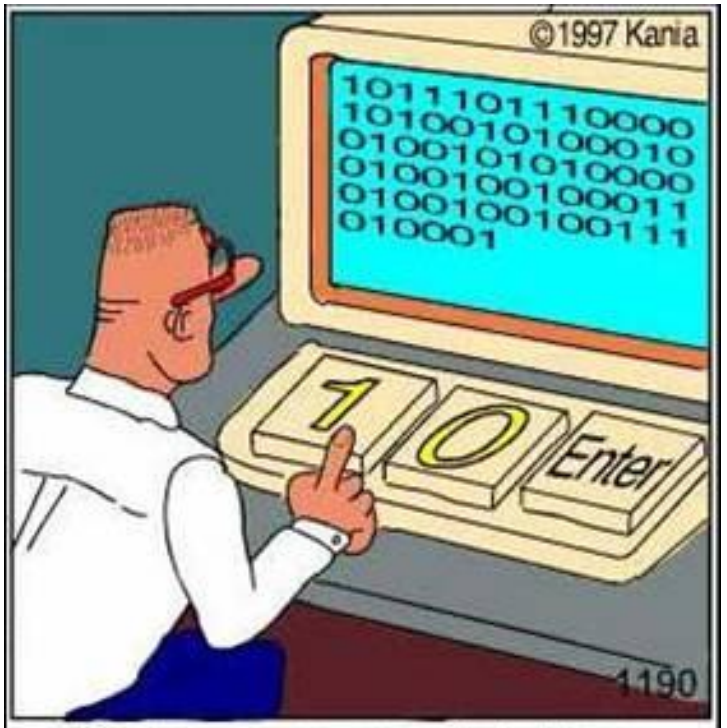
- Mathematician John von Neumann. Part of design of ENIAC, one of first electronic computers.
- Computer in his design consists of small CPU, larger main memory, bus
- Single CPU architecture still referred to as von Neumann machines.
- EDVAC report (Electronic Discrete Variable Arithmetic Computer) describes the first stored program computer.

Programming Language Generations

- First Generation (late 1940s):

  Machine-level programming languages

  - Fast and efficient, executed directly on the CPU
  - Consists only of 0s and 1s
  - Difficult for humans to read, write, and debug



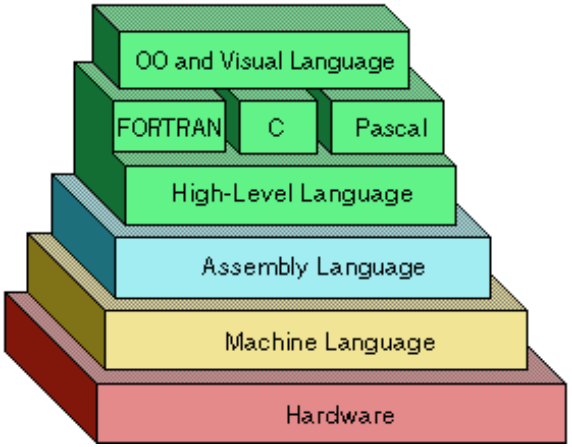
Real programmers code in binary.

Programming Language Generations

- Second Generation

  (early 1950s):

  - Symbolic assemblers
  - Interpreting routines
  - Very early compilers



Assembly languages

- Simple mnemonic instructions *<opcode> <operands>*
- **Assembler** translates into machine code
- Handcoding in assembly only for low-level needs

Programming Language Generations

- Third Generation
  (mid 1950s - present):

  High level, general-purpose

  - FORTRAN, LISP, COBOL, ALGOL
    (Ada, Basic, C, C++, Java, Pascal, Smalltalk, …)

  - Easier for humans to read, write, debug
  - **Compiler** translates into machine code before running
  - **Interpreter** translates into machine code at runtime

Programming Language Generations

- Fourth Generation (1970s -):

  Specification languages, query languages, report generators, systems engineering

  - Maple, Mathematica, Postscript, SPSS, SQL

- Fifth Generation (1980s -):

  Solve problems using constraints rather than algorithms, used in Artificial Intelligence

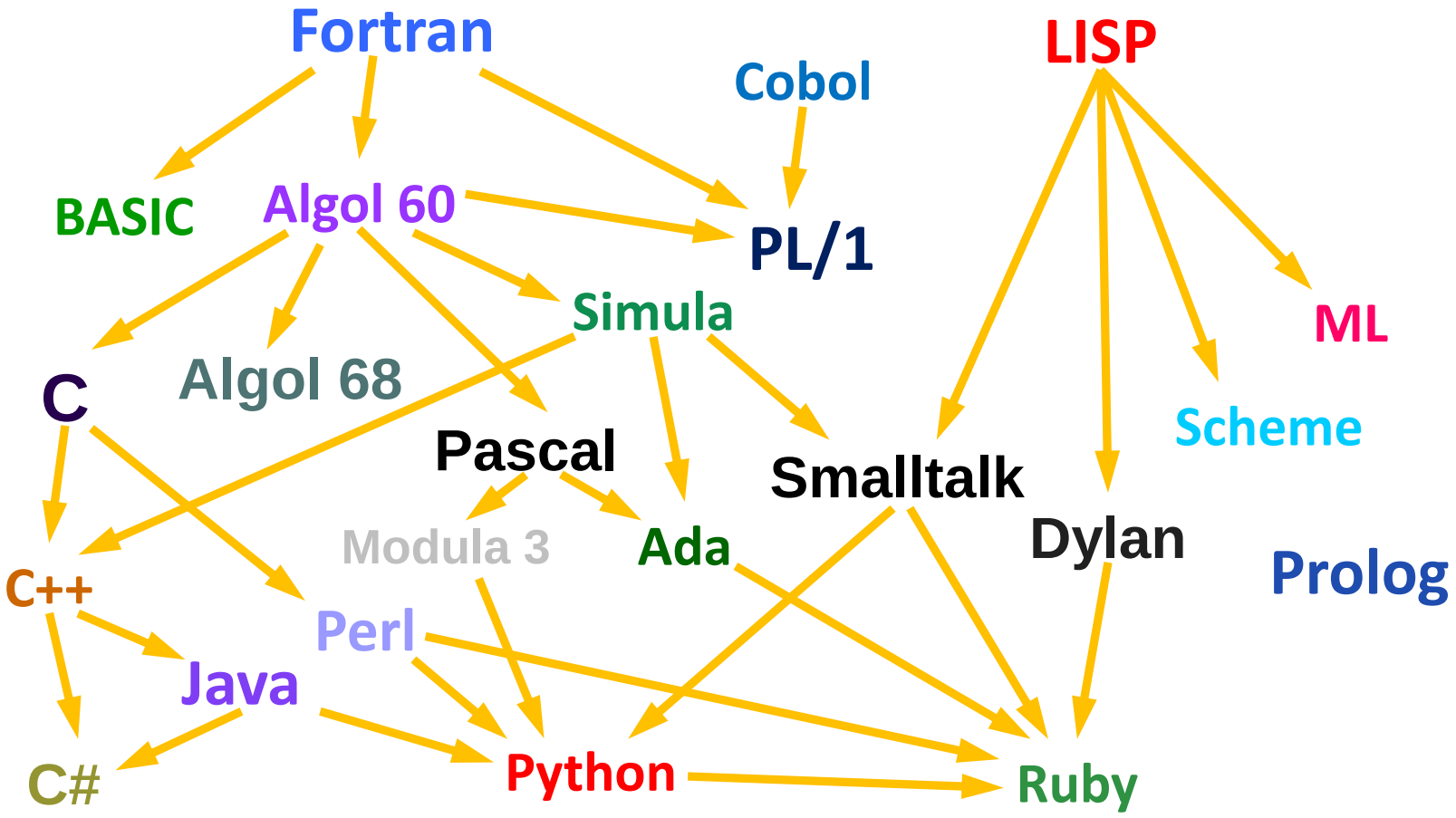
  - Prolog

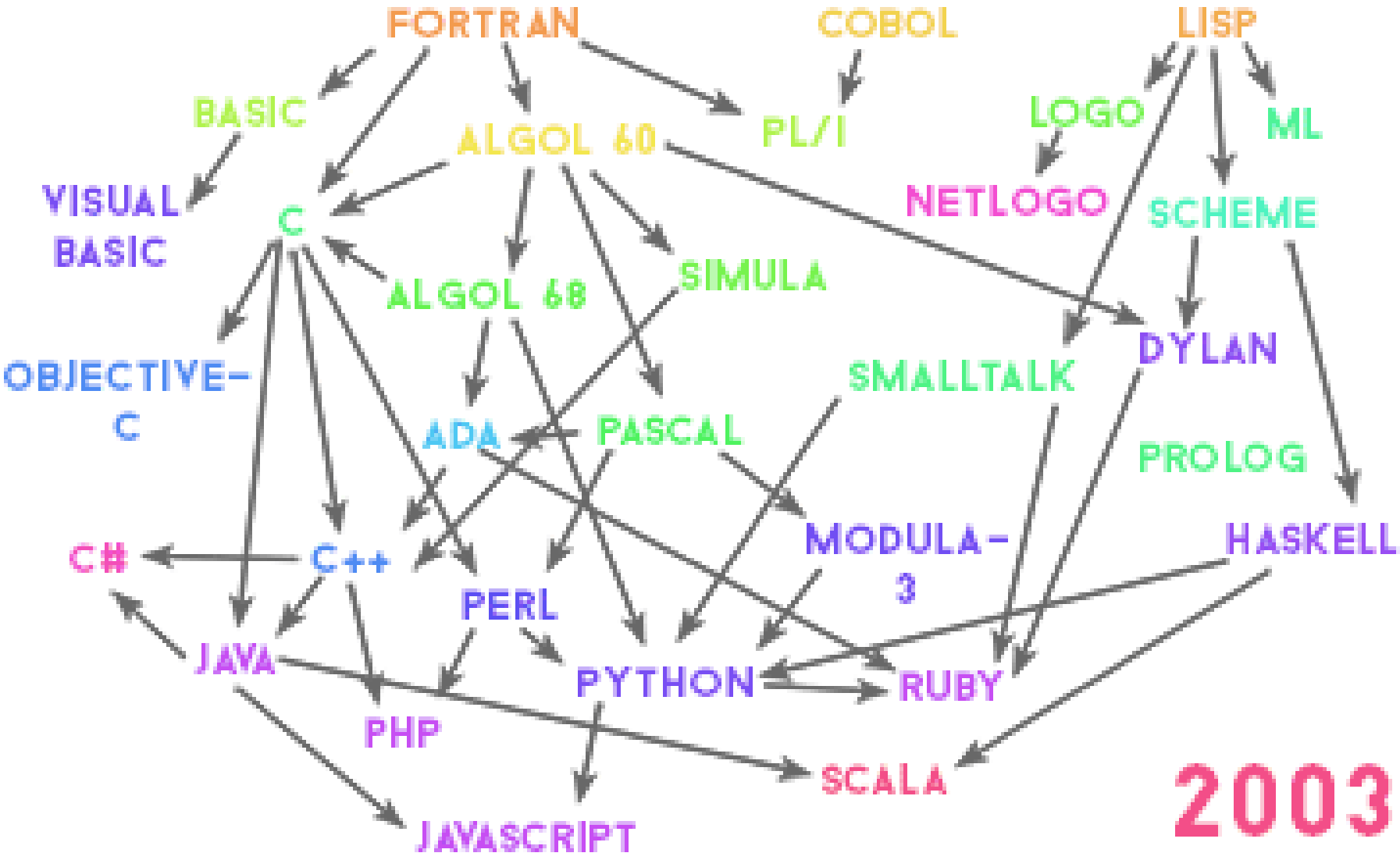
Konrad Zuse's Plankalkül 1945

K. Zuse

- Language for expressing computations
- Not published until 1972
- Anticipated many developments of programming languages
  - Arrays, records
  - Assertions
  - Algorithms for sorting, numerical computations, syntax analysis, and chess

A family tree of languages





© Camille Kim 2015

Evolution of third-generation Languages

- Begins with FORTRAN in 1954
- Generation of high-level programming languages
- Languages stress expressivity and machine independence
- Programming is procedural
- Includes imperative, functional, compiler languages

FORTRAN (1954)

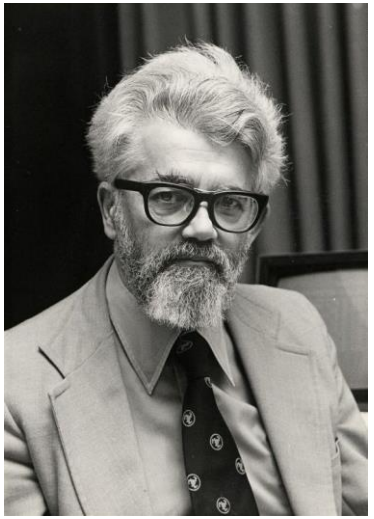
John Backus

- Designed at IBM to efficiently translate mathematical formulas into IBM 704 machine code. Wanted code at least as efficient as hand-coded.
- Language design was secondary to compiler design for optimization
- 1954 Report for a proposed Formula Translating System
- 1957 FORTRAN language manual published
- Translator produced code that in some cases was more efficient than the equivalent hand-coded program.

Innovations of Fortran

- language based on variables, expressions, statements
- the form of the arithmetic-assignment statement
- conditional and repetitive branching control structures
- arrays with maximum size known at compile time
- provision for comments

LISP (1958)



John McCarthy

- Interactive functional language
- Designed for IBM 704 by John McCarthy at Dartmouth 1956-1958
- Implemented at MIT. First reference manual published in 1960.
- Language based on lambda calculus. (Mathematical notation for expressing functions.)
- LISP was designed for symbolic formula manipulation. Stands for LISt Processor.
- Has become standard language of the AI community

Innovations of LISP

- the function as the basic program unit
- the list as the basic data structure
- dynamic data structures
- facilities for "garbage collection" of unused memory
- use of symbolic expressions as opposed to numbers
- recursion and the conditional expression as control structures
- the "eval" function for interactive evaluation of LISP statements

ALGOL (1958)

Peter Naur

- Designed by international team
- ALGOrithmic Language
- Several revisions:
  - ALGOL58
  - ALGOL60
  - ALGOL68
- ALGOL60 had profound influence on programming language design and on computer science. Pascal carries on tradition.
- ALGOL68 was a huge, general purpose language, not widely accepted.
- Language description published in ALGOL60 report
  - First appearance of Backus-Naur Form for programming language definition
- Widely used as a publication language for algorithms

Innovations of ALGOL60

- block structure and localized data environments
- nesting of program units
- free format program code
- explicit type declarations
- dynamic memory allocation
- parameter passing by value and by name

Cobol (1960)

Grace Hopper

- US Dept of Defense wanted "common" PL for data processing
- CODASYL committee (Conference on Data Systems Languages)
- Result was COBOL in 1960 (COmmon Business-Oriented Language)
- Grace Hopper was involved in development and wrote 1st compiler
- Designed to be machine independent, unlike FORTRAN.
- Influenced by Fortran, ALGOL58, and English.
- Example:

```
Multiply A by B giving C
Perform <loop body>
      Varying J from 2 by 1
      Until J > N.
```

- Major revisions standardized and released in 1968, 1974, and 1985.

Innovations of COBOL

- the record data structure
- file description and manipulation facilities
- machine independence of data and program descriptions
- influence of English
- relatively natural language style, including extra words for readability
- effort toward a language that would produce self-documenting program code

APL (early 1960s)

- A Programming Language
- Based on notation developed by Ken Iverson at Harvard 1957-1962.
- Functional, interactive, science-oriented language that assumes the array as the default data structure.
- Suitable for applications with a heavy use of numerical data in large multi-dimensional arrays.
- Used special symbols requiring special keyboard / printer

```
life←{↑1 ⍵∨.∧3 4=+/,¯1 0 1∘.⊖¯1 0 1∘.⌽⊂⍵}
```

BASIC (1964)

- Developed at Dartmouth in 1960's by Tom Kurtz, John Kemeny, and a succession of undergraduates; first ran in 1964.
- Beginner's All-purpose Symbolic Instructional Code
- Intended to introduce students in non-scientific disciplines to computing.
- Influenced by FORTRAN and ALGOL.
- Major goal to simplify user interface:
  - Simplicity chosen over efficiency
  - Time sharing over punched cards
  - Distinctions such as int vs real eliminated
  - Automatic defaults for declarations, values, arrays, output format, etc.
  - Clear error messages
  - Students had access to computers at all times
- No universal BASIC standard:
  - **ANSI** (American National Standards Institute) is a minimal standard.
  - **True Basic** – Kemeny's company

PL/1 (1964)

- Planned and designed by IBM as an extension to FORTRAN
- “Extension” departed from FORTRAN specs and was first released as NPL. Renamed PL/1 (Programming Language 1)
- Of interest in academic community because it had every element of language design. Too big and complicated.
- Compiler sold separately from machine
- COBOL and FORTRAN already had huge user bases

Innovations of PL/1

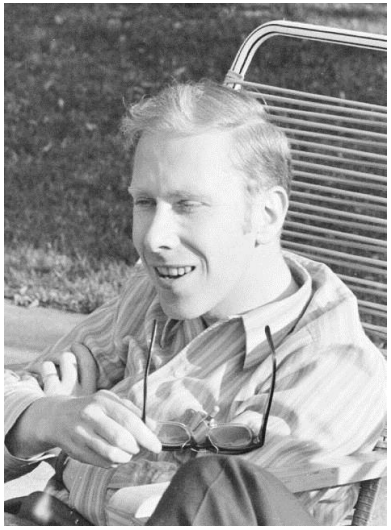
- multitasking
- programmer-defined exception handling
- explicit use of pointers and list processing
- wide variety of alternatives for storage allocation (static, automatic, controlled)
- consideration of problems arising from interacting with operating system

ALGOL68

- ALGOL committee produced considerably revised and extended version of ALGOL in 1968
- Huge, general-purpose language, very different from ALGOL60
- Not widely accepted, but influenced many other languages
- ALGOL68 introduced:
  - User-defined data type
  - Pointer type

  (Both significant features of Pascal)

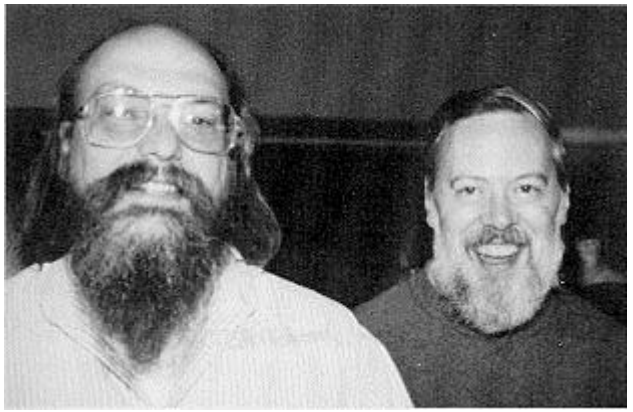
Pascal (1970)



Niklaus Wirth

- Designed by Niklaus Wirth
  - (member of ALGOL committee; he proposed a revision known as ALGOL-W in 1965)
- Pascal first implemented in 1970.
- In opposition to trend of PL/1 – ALGOL68 – Ada
- Named after 17th century French philosopher and mathematician Blaise Pascal.
- Simple and elegant
- Widely used in academic community
- Interesting features:
  - Case statement
  - Facility for user-defined data types
  - Record structure

C (1972)



K. Thompson and D. Ritchie

- Designed by Kenneth Thompson and Dennis Ritchie at Bell Labs in 1972.
- Designed for coding the routines of the UNIX operating system.
- “High level” systems programming language which created the notion of a portable operating system
- Concise syntax – programs somewhat hard to read, understand, debug, maintain
- No built-in operations for handling composite data types such as strings, sets, and lists.
- Not strongly typed. No run-time type checking. Easily leads to programming errors.
- Provides ability to code low-level operations in a high-level language.

Ada

- Designed according to specifications developed by US Dept of Defense
- Requirements stressed structural programming methodology and readability over writability
- Development period 1975 – 1985
  - 1975: first requirements documents
  - 1980: complete language proposed
  - 1983: final standardized version
  - 1985: working usable compilers appeared
- Contains virtually all elements of PL design
  - Exception handling
  - Parallel processing
  - Abstract data types

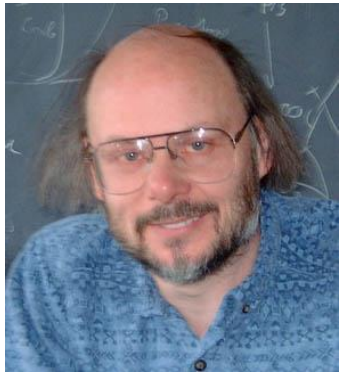
Programming Language Paradigms

- Procedural: procedures, sequential execution of code are basic building blocks of program
  - FORTRAN (FORmula TRANslating; John Backus, IBM, 1950s)
  - ALGOL (ALGOrithmic Language, 1958)
  - COBOL (COmmon Business Oriented Language, 1960)
  - BASIC (Beginner's All-purpose Symbolic Instruction Code, John Kemeny and Thomas Kurtz, Dartmouth, 1963)
  - Pascal (Niklaus Wirth, 1970)
  - C (Dennis Ritchie, Bell Labs, 1972)

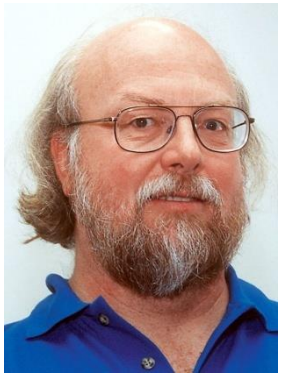
Programming Language Paradigms

- Object-Oriented: Program is designed around the *objects* required to solve the problem
  - Smalltalk (Alan Kay, Xerox PARC, 1971)
  - Ada (US Dept of Defense, 1975)
  - C++ (Bjarne Stroustrup, Bell Labs, 1983)
  - Java (James Gosling, Sun Microsystems, 1995)
  - C# (Microsoft, 2000)

Alan Kay



B. Stroustrup



J. Gosling

Programming Language Paradigms

- Functional: Program is designed around the evaluation of *functions*, rather than modifying state
  - LISP (John McCarthy, MIT, 1958)
    - Common Lisp
    - Dylan
    - Logo
    - Scheme
  - ML (Robin Milner et al, Edinburgh, 1970s)
  - Haskell (purely functional language, 1990)

Programming Language Paradigms

- Logic: Program is declarative, based on *mathematical logic*
  - Prolog (1972)

    A program lists *facts* and *rules*, program execution is controlled deduction to answer a *query*.

Programming Language Paradigms

- Scripting and multi-paradigm languages
  - awk (Aho, Weinberger, Kerningham, Bell labs, 1978)
  - Perl (Larry Wall, NASA, 1987)
  - Tcl/Tk (John Ousterhout, 1988)
  - Python (Guido van Rossum, CWI, 1991)
  - PHP (Rasmus Lerdorf, 1995)
  - Ruby (Yukihiro Matsumoto, 1996)


COBOL
ALGOL
PL/1
ADA
DOD
FORTRAN
PASCAL
MODULA 2
C
UNIX
N.W.
LISP
BASIC
APL
FORTH
A⇒B
PROLOG
MACHINE LANGUAGE
PILOT
ASSEMBLY
LOGO
A
B
C
W
'85

Sources

- Sethi, *Programming Languages*, 2nd edition
- Sebasta, *Concepts of Programming Languages*, 8th edition
- Wikipedia (most images)
- Old CS 101 and CS 313 lecture notes