A Brief History of Programming Languages

- **Wegner, "Prog. Langs-The First 25 Years", IEEE Transactions on Computers, 12/76, 1207-25**

- ***1950's "Discovery and description"***

  → **assembly**

  → **FORTRAN, ALGOL60, COBOL, LISP**

  → **basic implementation techniques**

  symbol tables

  stack evaluation of arithmetic

  activation records

  garbage collection

  → **languages as tools**

  → **late 1950's: first compilers (Hopper, etc.)**

  grammars and automata (Chomsky and Miller)

A Brief History of Programming Languages

- ***1960's "Elaboration and analysis"***

  → **theories of programming languages**

  → **more formal development**

  formal languages

  automata

  formal semantics

  verification

  → **bigger, more complex languages**

  → **PL/I, Simula, ALGOL68**

  → **late 1960's: theoretical work on compilers, program optimization**

A Brief History of Programming Languages

- ***1970's "Technology"***
  - → **practical issues**
  - → **applications of computer science**
  - → **hardware cheaper, faster**
  - → **software complexity increased**
  - → **programming methodologies**
    - structured programming
    - program verification
  - → **Pascal, C, Modula, Clu**

A Brief History of Programming Languages

- ***1980's***

  → **parallel hardware => parallel language**

  → **very high-level languages**

  functional

  → **logic**

  → **(object-oriented)**

Specific Milestones

- ***1944*: EDVAC (Electronic Discrete Variable Automatic Calculator) Report (von Neumann)**
  - → **first description of a stored-program computer**
- ***1950*: First Assemblers**
- ***1954-57*: FORTRAN ("FORmula TRANslating system")**
  - → **Backus et al @ IBM**
  - → **Goals:**

    efficiency -- less than twice as slow as assembler

    solve economic problem -- design, coding, debugging too expensive in assembly
  - → **elegance of design secondary**
  - → **Versions I, II, III, IV**
  - → **introduced separate compilation with II because programs were getting too large to compile without hardware errors (300 - 400 SLOCS)**
  - → **"An existence proof for higher-level languages..."**

Specific Milestones

- ***1958-60*: ALGOL 60 ("ALGOrithmic Language")**
  - → **by committee, including Backus**
  - → **Goals:**

    elegant, universal language (FORTRAN was for IBM)

    standard mathematical notation

    major contributions:

    BNF
    block structure
    recursion
    call-by-value/name
    stack model of evaluation
    semi-dynamic arrays
    but no formatted I/O -- too machine-dependent

- ***1956-62*: LISP ("LISt Processing")**
  - → **McCarthy @ MIT**
  - → **for symbolic computation in AI**
  - → **free of von Neumann concepts**
  - → **(roughly) based on lambda-calculus**

Specific Milestones

- ***1956-62*: APL ("A Programming Language")**
  - → **Iverson @ Harvard**
  - → **array processing**
  - → **functional flavor, fairly non-von Neumann**
  - → **didn't catch on until 1970's**
- **1960: COBOL ("COmmon Business Oriented Language")**
  - → **at U Penn by representatives of computer manufacturers**
  - → **alienated from CS community**
    - developed by commercial community; didn't ask CS'ers
    - no interest in scientific or research implications
    - no BNF definition
    - no good books
    - commercial applications thought trivial by CS'ers
  - → **main contribution: <u>file/record structure</u>**
  - → **syntax wordy, <u>English-like</u>**
  - → **very slow at first, but survived because use mandated by DoD**
  - → **ref: Schneiderman, *Annals of the History of Computing*, 10/85**

Specific Milestones

- ***1960's*: BASIC**
  - → **Kemeny and Kurtz @ Dartmouth**
  - → **for teaching**
  - → **access through terminals**
  - → **novel idea: <u>user time more important than machine time!</u>**
  - → **commercial success a surprise -- intended for their students**
  - → **no real contributions**

- ***1962-67*: SNOBOL4 ("StriNg Oriented symBOlic Language")**
  - → **Griswold @ Bell Labs**
  - → **string processing**
  - → **introduced pattern-matching**

- ***1964-69*: PL/I ("Programming Language I")**
  - → **by committee @ IBM**
  - → **tried to unify commercial and scientific features**
  - → **very large; programmers learn a subset**

Async tasks, except. handling, pointer data types, array slices

Specific Milestones

- ***1963-68*: ALGOL68**
  - → **by committee**
  - → **small number of orthogonal constructs**
  - → **hard to learn -- too general and too flexible**
  - → **poor implementations/documentation**
- ***1967-71*: Pascal**
  - → **Wirth**
  - → **small, simple -- for teaching**
  - → **structured programming, fairly rich data structures**
- **1967: Simula 67**
  - → **Data Abstraction**
  - → **Class Concept**
  - → **Data and operations packaged together**

Specific Milestones

- ***~1973*: C**
  - → **Kernighan and Ritchie @ Bell Labs**
  - → **low level, for systems programming**
  - → **fairly small, fast**
  - → **hard to read and maintain**
- ***mid 1970's*: Modula-2**
  - → **Niklaus Wirth**
  - → **Pascal and modules**
  - → **better for systems programming and large projects**
- ***mid 1970's*: PROLOG ("PROgramming in LOGic")**
  - → **Kowalski and Colmerauer @ Edinburgh and Marseilles**
  - → **non-von Neumann, based on first-order logic (but impure)**
  - → **most applications in AI**
  - → **Japanese 5th generation computing project chose it**

Specific Milestones

- **mid 1970's: SMALLTALK**
  - → **Xerox**
  - → **object-oriented: shift in focus**
  - → **not just a language; a whole system**
- **mid 1970's - 80: Ada (after Ada Augusta, associate of Babbage -- "the first programmer")**
  - → **DoD**
  - → **requirements developed slowly:**

    Strawman, Woodman, Tinman, Ironman, Steelman
  - → **design contract won by CII-Honeywell Bull (Jean Ichbiah)**
  - → **based on Pascal**
  - → **large, complex**
  - → **features:**

    packages, tasks, real-time capabilities, exception handling

Specific Milestones

- ***1980's*: C++**
  - → **Bjorne Stroustrup**
  - → **C + classes**
  - → **OOP in a popular language**
- ***1980's*: Hope, Miranda, LML, Haskell**
  - → **purely functional**
  - → **based on lambda-calculus**
  - → **higher-order functions, pattern matching, type inferencing**
  - → **good for parallel machines?**