Pre-Computer

* Ada Lovelace
  + First Programmer, was a mathematician
  + Fascinated with the idea of algorithms being used to solve problems for us
* Zuse’s Plankalkul
  + First ‘language’
  + Designed in 1945, but not published until 1972
  + Never implemented
  + Advanced data structures
    - Floating point, arrays, records
    - IEEE 754 – Standard for representing floating point numbers
  + Invariants – Used to prove the algorithm, correct or not

Pseudocodes

* Speedcoding
  + Pseudo operations for arithmetic and math functions
  + Conditional and unconditional branching
  + Auto-increment registers for array access
  + Slow!

IBM 704 and Fortran

* Fortran 0: 1954 – not implemented
* Fortran I: 1957
  + Designed for the new IBM 704, which had index registers and floating point hardware
  + This led to the idea of new computer programming languages, because there were no others
  + Environment of development
    - Computers were small and unreliable
    - Slow
    - Complex to operate
  + Overview
    - Only types were integers and floating point values
    - No separate compilation
    - Programs for over 400 lines rarely compiled correctly due to poor reliability of the 704
* Fortran Evaluation
  + Highly optimizing all compilers
    - Types and storage of all variables are fixed before run time

Functional Programming: Lisp

* List Processing Language
  + Designed by MIT by McCarthy
* AI research needed a language to
  + Process data in lists rather than arrays
* Only two data types: atoms and lists
* Syntax is based on ‘lambda calculus’
* Evaluation
  + Pioneered functional programming
    - No need for variables or assignment
  + Still the dominant language for AI
  + Common Lisp and Scheme are contemporary dialects of Lisp
  + ML, Haskell, and F# are also functional programming languages, but use very different syntax
* Scheme
  + Developed by MIT in 1970s

The First Step Toward Sophistication: ALGOL 60

* Environment of development
* ALGOL 60 was the result of efforts to design a universal language
* Overview
  + New from 58
    - Block structure
    - Two parameter passing methods
    - Subprogram recursion
    - Stack-dynamic arrays
  + Still no I/O and string handling
  + Successes
    - Was the standard way to publish algorithms for over 20 years
    - Subsequent imperative languages are based on it
    - First machine-independent language
    - First language whose syntax was formally defined (BNF)
  + Failure
    - Never widely used in US
    - Reasons
      * Lack of I/O
      * Too flexible – hard to implement
      * Entrenchment of Fortran
      * Formal syntax description
      * Lack of support from IBM

Computerizing Business Records: COBOL

* Based on FLOW-MATIC
  + Names up to 12 characters
  + English names for arithmetic operators
* Design Process
  + First Design Meeting (Pentagon) – May 1959
  + Goals
  + Look like simple English
  + Easy to use, even if less powerful
  + Broaden the access for base computers
* Contributions
  + First macro facility in a high level language
  + Hierarchical data structures
  + Nested selection statements
  + Long names (30 char)
  + Separate data division

The Beginning of Timesharing: Basic

* Designed by Kemeny & Kurtz at Dartmouth
* Goals
  + Easy to learn and use for non-science students
  + 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 with time sharing

Everything for everybody: PL/I

* Designed by IBM and SHARE
* Computing situation in 1964
  + Scientific computing
* Overview
  + Contributions
    - First exception handling
    - Switch selectable recursion
    - First pointer data type
    - First array cross sections

Early Dynamic Lanugages: APL and SNOBOL

* Dynamic typing and dynamic storage allocation
* Variables are untyped
  + A variable acquires a type when it is assigned a value
* Storage is allocated to a variable when it is assigned a value
* APL
  + Designed as a hardware description language at IBM around 1960
  + Highly expressive
  + Programs are very difficult to read
* SNOBOL
  + Designed as a string manipulation language
  + Powerful operators for string pattern matching
  + Slower than alternative languages
  + Still used for certain text processing tasks

Pascal -1971

* Developed by Wirth
* Designed for teaching structured programming
* Small, simple, nothing really new
* Largest impact was on teaching programming
  + From mid-1970s to late 1990s it was most used language for teaching programming