Lecture Notes 2

Zuse's Plankalkül

- Designed by Konrad Zuse for Z4 electromechanical computer
- Started in 1943, but not published until 1972. Never fully implemented
- Supported integer, floating-point, array and struct data types
- Had features similar to assertions, iteration and selection

Historical Perspective

1. 1st Generation Programming Languages – Machine languages (1940's)
   - Machine-specific
   - Verbose
   - Modularity: zero
2. 2nd Generation Programming Languages – Assembly languages (1950's)
   - Machine-specific
   - Verbose
   - Modularity: zero
3. 3rd Generation Programming Languages – First high level languages (1960's)
   - Languages are imperative – Programmer defines sequence of actions
   - Portable, concise
   - Modularity better (subroutines, procedures)
   - The transition 2nd → 3rd brought about:
     - *Small Increase* in cost of use as programs became slightly bigger and slower
     - *Large Reduction* in cost of development as programs became much cheaper, more portable, and more reliable.
   - 3rd generation languages have mostly replaced 2nd generation languages.
   - Early Examples: FORTRAN, ALGOL, COBOL
   - Modern Examples: C/C++, C#, Java, Pascal
4. 4th Generation Programming Languages – Higher-level declarative languages (1970's)
   - Declarative Language – Programmer defines what should be done, not how to perform the task.
   - The quest for another similar increase in programmer productivity continues to this day, 4th generation not completely successful in its.
   - Domain Specific – Often languages have specific narrow domain
   - Lines are often blurred between 3rd and 4th generation languages.
   - Examples: SQL, XUL, CSS
5. 5th Generation Programming Languages – Constraint based programming (1980's)

- Constraints Programming – Programmer specifies what is/is not allowed, does not write an algorithm
- Often used in AI research.
- Examples: Prolog, OPS5, Mercury
- Lines are often blurred between 4th and 5th generation languages

Major Advancements/Influences and Programming Languages

- 1950–55:
  - Hardware: Vacuum-tube computers
  - Methods: Assembly languages; foundation concepts: subprograms, data structures;
  - Languages: experimental use of expression compilers
- 1956–60: Small, slow and expensive computers,
  - Hardware: Magnetic tape storage; core memories; transistor circuits
  - Method: Compilers, software interpreters, code optimization, dynamic storage management, linked data structures, BNF grammars
  - Languages: FORTRAN, ALGOL58 and Algol60, COBOL, LISP
- 1961–65:
  - Hardware: Large expensive computers, Magnetic-tape storage systems,
  - Methods: Operating systems, Multi-programming, Syntax-directed compilers,
  - Languages: FORTRAN IV, Algol60-revised, SNOBOL. APL
- 1966-70:
  - Hardware: Microprogramming, minicomputers, ICs
  - Methods: Time-sharing interactive operating systems, Optimizing compilers, Translator writing systems,
  - Languages: PL/I, FORTRAN 66 (Standard), COBOL 65 (Std), Algol 68, Snobol 4, Simula 67, BASIC, APL
- 1971-1975:
  - Hardware: Microcomputers, Small inexpensive storage systems
  - Methods: Proofs of program correctness, Structured programming, Software engineering, Reactions against large complex languages
  - Languages: Pascal, COBOL 74 (Std), PL/I (Std)
- 1976-1980:
  - Hardware: Powerful inexpensive computers, Large inexpensive storage systems, Distributed computers systems,
  - Methods: Concurrent and real-time programming using high level languages, Interactive programming environments, Data abstraction, Software components, Formal semantic definitions, Reliability and ease of maintenance as language design goals
  - Languages: Ada, Fortran 77, ML
- 1981–85:

    - Hardware:Personal computers, first workstations, large mass storage systems, distributed computing
    - Methods: Object-oriented programming; interactive environments; syntax-directed editors;
    - Languages: Turbo Pascal, Smalltalk-80, Ada 83, Postscript.
- 1986–90:
    - Hardware: Microcomputers; Rise of engineering workstations; RISC architectures; global networking; Internet
    - Methods: client-server computing
    - Languages: FORTRAN 90, C++, SML
- 1991–Today:
    - Hardware: Very fast inexpensive workstations, Massively parallel architectures, voice, video, multi-media
    - Methods: Open Systems, Environmental frameworks; Information super-highway
    - Languages: Ada 95, TCL, Java, Python, C#, etc.