Chapter 2

1. Plankalkul - 1945

- Never implemented
- Advanced data structures
    - floating point, arrays, records
- Invariants
- Notation:

```
A(7) := 5 * B(6)

     |  5  *  B  =>  A
V   |        6        7          (subscripts)
S   |       1.n    1.n          (data types)
```

2. Pseudocodes - 1949

What was wrong with machine code?

a. Readability
b. Modifiability
c. Expression coding
d. Machine deficiencies--indexing and fl. pt.

- *Short code; 1949; BINAC; Mauchly*
    - Expressions were coded, left to right
    - Some operations:

      1n => (n+2)nd power
      2n => (n+2)nd root
      07 => addition

Chapter 2

2. Pseudocodes (continued)

- *Speedcoding; 1954; IBM 701, Backus*
  - Pseudo ops for arithmetic and math functions
  - Conditional and unconditional branching
  - Autoincrement registers for array access
  - Slow!
  - Only 700 words left for user program

3. Laning and Zierler System - 1953

- MIT Whirlwind
- First "algebraic" compiler system
- Subscripted variables, function calls, expression translation
- Never ported to any other machine

4. FORTRAN 0 - 1954

- Designed for the new IBM 704, which had index registers and floating point hardware
- Not implemented
- Environment of development:
  1. Computers were small and unreliable
  2. Applications were scientific
  3. No programming methodology or tools
  4. Machine efficiency was most important

Chapter 2

4. FORTRAN 0 (continued)

- *Impact of environment on design*
  1. No need for dynamic storage
  2. Need good array handling and counting loops
  3. No string handling, decimal arithmetic, or powerful input/output (commercial stuff)

- *Features:*
  - Two-character variable names
  - Two-way selector
  - Posttest counting loop
  - No data typing statement
  - No formatted i/o
  - No user-written subprograms

5. FORTRAN I - 1956

- First implemented version of FORTRAN
- Extend names to six characters
- Formatted i/o
- User-defined subprograms
- Still no separate compilation
- Compiler released in April 1957
- Programs larger than 400 lines seldom compiled correctly
- Code was very fast

Chapter 2

6. FORTRAN II - 1958

- Independent compilation
- Fix the bugs

7. FORTRAN IV - 1960-62

- Type declarations
- Subprogram names could be parameters
- ANSI standard in 1966

8. FORTRAN 77 - 1978

- Character string handling
- Logical loop control statement
- IF-THEN-ELSE statement

9. FORTRAN 90 - 1990

- Modules
- Dynamic arrays
- Pointers
- Recursion
- CASE statement
- Parameter type checking

Chapter 2

10. LISP - 1959

- Designed at MIT by McCarthy
- *AI research needed a language that:*
  1. Process data in lists (rather than arrays)
  2. Symbolic computation (rather than numeric)
- Only two data types: atoms and lists
- *Pioneered functional programming*
  - No need for variables or assignment
  - Recursion and conditional expressions
- Still the dominant language for AI

11. ALGOL 58 - 1958

- *Environment of development:*
  1. FORTRAN had (barely) arrived for IBM 70x
  2. Many other languages were being developed, all for specific machines
  3. No portable language; all were machine-dependent
  4. No universal language for communicating algorithms

- ACM and GAMM met for four days for design
- *Goals:*
  1. Close to mathematical notation
  2. Good for describing algorithms
  3. Must be translatable to machine code

Chapter 2

11. ALGOL 58 (continued)

- *Three representations*:
    1. Reference language
    2. Publication language
    3. Hardware language

- *Features*:
    - Names could have any length
    - Arrays could have any number of subscripts
    - Parameters were separated by mode
    - Subscripts in brackets
    - Compound statements (begin ... end)
    - Semicolon as a statement separator
    - Assignment operator was :=
    - If had an else-if clause

- *Comments:*
    - Not meant to be implemented, but variations of it were implemented (MAD, JOVIAL)
    - Although IBM was initially enthusiastic, all support was dropped by mid-1959

Chapter 2

12. ALGOL 60 - 1960

- Modified ALGOL 58 at 6-day meeting in Paris
- *Features*:
  - Block structure (local scope)
  - Two parameter passing methods
  - Recursion
  - Semidynamic arrays
  - Still no i/o or string handling

- *Successes*:
  - It was the standard way to publish algorithms for over 20 years
  - All subsequent imperative languages are based on it
  - First machine-independent language
  - First language whose syntax was formally defined (BNF)

- *Failure*:
  - Never widely used, especially in U.S.

    Reasons:

    1. No i/o
    2. Character set made programs nonportable
    3. Too flexible--hard to implement
    4. Intrenchment of FORTRAN
    5. Formal syntax description

Chapter 2

13. COBOL - 1960

- *Environment of development:*
  - UNIVAC was beginning to use FLOW-MATIC
  - USAF was beginning to use AIMACO
  - IBM was developing COMTRAN

- *Based on FLOW-MATIC*
  - FLOW-MATIC features
    - names up to 12 characters, with embedded hyphens
    - English names for arithmetic operators
    - Data and code are completely separate
    - Verbs were first thing in every statement

- *First Design Meeting - May 1959*
  - Design goals:
    1. Must look like simple English
    2. Must be easy to use, even if that means it will be less powerful
    3. Must broaden the base of computer users
    4. Must not be biased by current compiler problems
  - Design committee were all from computer manufacturers and DoD branches
  - Design Problems: arithmetic expressions? subscripts? Fights among manufacturers

Chapter 2

13. COBOL (continued)

- *Contributions:*
  - First macro facility in a high-level language
  - Hierarchical data structures (records)
  - Nested selection
  - Long names (up to 30 characters)
  - Data division

- *Comments:*
  - First language required by DoD
  - Would have failed without DoD
  - Still the most widely used business applications language

14. BASIC - 1964

- Designed by Kemeny & Kurtz at Dartmouth
- Design Goals:
  - Easy to learn and use for non-science students
  - "Pleasant and friendly"
  - Fast turnaround for homework
  - Free and private access
  - User time is more important than computer time

Chapter 2

15. PL/I - 1965

- Designed by IBM and Share
- *Design environment (IBM's point of view)*
    1. Scientific computing
        - IBM 1620 and 7090 computers
        - FORTRAN
        - Share user group

    2. Business computing
        - IBM 1401, 7080 computers
        - COBOL
        - GUIDE user group

    3. Scientific users began to need more elaborate i/o, like COBOL had; Business users began to need fl. pt. and arrays (MIS)

    4. It looked like many shops would begin to need two kinds of computers, languages, and support staff

- *The obvious solution:*
    1. Build a new computer to do both kinds of applications
    2. Design a new language to do both kinds of applications

Chapter 2

15. PL/I (continued)

- *PL/I contributions:*
  1. First concurrency
  2. First exception handling
  3. Switch-selectable recursion
  4. First pointers
  5. First array cross sections

- *Comments:*
  - Many of the new features were poorly designed
  - Too large and too much redundancy
  - Was (and still is) actually used for both scientific and business applications

16. SIMULA 67 - 1967

- Based on ALGOL 60 and SIMULA I
- Designed for system simulation
- *Contributions:*
  - Coroutines - a kind of subprogram
    - Implemented in a structure called a class
      - Classes are the basis for data abstraction; a structure that includes local data and functionality

Chapter 2

17. ALGOL 68 - 1968

- From the continued development of ALGOL 60, but it is not a superset of that language
- Design is based on the concept of orthogonality
- *Contributions:*
  1. User-defined data structures
  2. Reference types
  3. Dynamic arrays (called flex arrays)

- *Comments:*
  - Had even less usage than ALGOL 60
  - Had strong influence on subsequent languages, especially Pascal, C, and Ada

18. Pascal - 1971

- Designed by Wirth, who quit the ALGOL 68 committee (didn't like the direction of that work)
- Designed for teaching structured programming
- Small, simple, nothing really new
- Still the most widely used language for teaching programming in colleges

Chapter 2

19. C - 1972

- Evolved primarily from B, but also ALGOL 68
- Designed for systems programming
- Powerful set of operators, but poor type checking
- Initially spread through UNIX

20. Prolog - 1972

- Developed at the University of Edinburgh and the University of Aix-Marseille, by Comerauer, Roussel, and Kowalski
- Based on formal logic
- Non-procedural
- Can be summarized as being an intelligent database system that uses an inferencing process to infer the truth of given queries

21. Smalltalk - 1980

- Developed at Xerox PARC, initially by Alan Kay
- First full implementation of an object-oriented language (data abstraction, inheritance, dynamic type binding)
- Pioneered the graphical user interface everyone now uses

Chapter 2

22. Ada - 1983

- Huge design effort, involving hundreds of people, much money, and about eight years

- *Contributions:*
  1. Packages - support for data abstraction
  2. Exception handling - elaborate
  3. Generic program units
  4. Concurrency - through the tasking model

- *Comments:*
  - Competitive design
  - Included all that was then known about software engineering and language design
  - First compilers were very difficult; the first really usable compiler came nearly five years after the language design was completed

23. C++ - 1985

- Developed at Bell Labs by Stroustrup
- Evolved from C and SIMULA 67
- Facilities for object-oriented programming, taken partially from SIMULA 67, were added to C
- A large and complex language
- Rapidly growing in popularity, along with OOP