

- Developed for the IBM 701
- Interpreter converted the 701 to a virtual threeaddress f-p calculator
- pseudoinstructions for + * / sqrt, since, atan, exp, log
- Conditional and unconditional branching
- Input/output conversions
- Autoincrement registers for array access(allowed matrix multiplication in 12 instructions)
- But

Chapter 2

- Slow!
- Only 700 words left for user program

Prog

Programming Languages

5

2.3 The UNIVAC "Compiling" System (1954)

- Compiling system expanded a pseudocode into machine code in the same way as macros are expanded into assembly language.
 - Primitive but much shorter source codes
 2.4 Related Work
- Cambridge Un. using blocks of relocatable addresses to solve the problem of absolute addresses
- This idea was extended to design an assembly language (no impact of Als on HLLs) that could combine chosen subroutines and allocate storage.
 Chapter 2 Programming Languages 6

FORTRAN: The IBM FORmula TRANslating System (1957)

FORTRAN 0 & IBM 704 Machine

- · first computer with index registers and f-p hw
 - So far all f-p ops had to be simulated in software which made interpretation an acceptable alternative to machine code execution (interp. cost used to be hidden by the high cost of f-p calcs)
- FORTRAN is credited with being the first compiled HLL. [no consensus though!]
 - The Laning and Zierler system (1954) of MIT was the first algebraic translation system: it translated arith. expressions, used function calls for math. functions, and included subscripted var. references.

7

9

11

Programming Languages

FORTRAN I (1956)

- Input/output formatting
- var. names of up to six characters (from 2 in v.0)
- user-defined subroutines(no separate compilation)
- · IF selection statement, post-test counting DO loops
- Arithmetic IF (3-way branch) instead of logical IF!!
- No data typing (I to N: integer, otherwise: f-p)
- Compiler released in April 1957, after 18 worker/ years of effort
- Programs larger than 400 lines rarely compiled correctly, mainly due to poor reliability of the 704

8

10

- Code was very fast
- Quickly became widely used

Chapter 2 Programming Languages

Environment in which FORTRAN was developed:

- Computers were small and unreliable
- Applications were scientific
- No programming methodology or tools
- · Machine efficiency was most important

FORTRAN II (1958)

- · Independent compilation of subroutines + linkers
- Fix the bugs

Chapter 2

FORTRAN IV (1960-62)

- Explicit type declarations
- · Logical selection statement
- · Subprogram names could be parameters
- ANSI standard in 1966
- Chapter 2 Programming Languages

FORTRAN 77 (1978)

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

FORTRAN 90 (1990)

- Modules- PRIVATE or PUBLIC data and subprograms
- Built-in array operations: SUM, MATMUL, etc.
- Dynamic arrays: ALLOCATABLE
- Pointers
- Recursion recursive procedures
- · CASE multiple selection statement
- Parameter type checking
- Chapter 2 Programming Languages

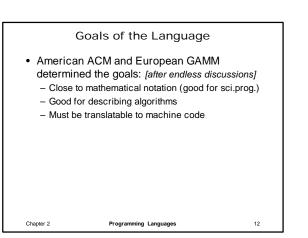
ALGOL - ALGOrithmic Language

ALGOL 58 (1958)

- · Environment of development:
 - FORTRAN had (barely) arrived for IBM 70x
 - Many other languages were being developed, all for specific machines
 - No portable language; all were machine-dependent
 - No universal language for communicating algorithms
 FORTRAN could not become a universal language because at the time it was solely owned by IBM.

Chapter 2

Programming Languages

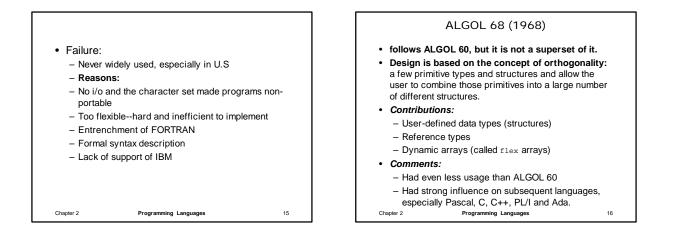


ALGOL 58 Features

- · Concept of type was formalized - vars. not f-p numbers required explicit declaration
- · Names could have any length
- · Arrays could have any number of subscripts
- · Parameters were separated by mode (in & out)
- · Subscripts were placed in brackets
- Compound statements (begin ... end)
- Semicolon as a statement separator
- Assignment operator was :=
- if had an else-if clause a for statement for counting loops
- a do statement for subprogram calls.
- Chapter 2 Programming Languages

ALGOL 60 (1960)

- · New Features:
 - Block structure (local scope)
 - Two parameter passing methods: by value/by name
 - Subprogram recursion
 - Stack-dynamic arrays
 - Still no i/o and no string handling -subscript range is specified by vars during execution.
- · Successes:
 - It was the standard way to publish algorithms for over 20 years
 - All subsequent imperative langs are based on it
 - First machine-independent language
- First lang. whose syntax was formally defined (BNF) Chapter 2 Programming Language



13

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 (but use is shrinking)

Weaknesses:

- · A subprogram can not take an array of variable length as argument
- · Lack of any separate compilation capability

Chapter 2

Programming Languages

17

C (1972)

- Designed for systems programming (at Bell Labs by Dennis Richie)
- Evolved primarily from B, but also ALGOL 68
 - for and switch statements
 - assignment operators
 - treatment of pointers
- · Powerful set of operators: high expresiveness
- But poor type checking: esp. function parameters
- Too flexible & very insecure
- Initially spread through UNIX: inexpensive or free compiler support

Chapter 2

Programming Languages

18



- Developed at Bell Labs by Stroustrup
- C with Classes(1983): Evolved from C & SIMULA 67:
 - function parameter type checking and conversion
 - classes

Chapter 2

- derived classes
- public/private access control of inherited components

19

- constructor and destructor functions
- friend classes
- And later in 1981:
- inline functions
- default parameters
- overloading of assignment operator
 - Programming Languages

And later in 1984: (It has become C++)

- virtual functions
- function name and operator overloading
- reference types
- And later multiple inheritance, abstract classes, typesafe linkage.
- Downward compatible with C
- · Also has exception handling
- A large and complex language, in part because it supports both procedural and OO programming

Programming Languages

- Rapidly grew in popularity, along with OOP
- ANSI standard approved in November, 1997

Chapter 2

20

