“This has all happened before and it will all happen again.”

J.M. Barrie

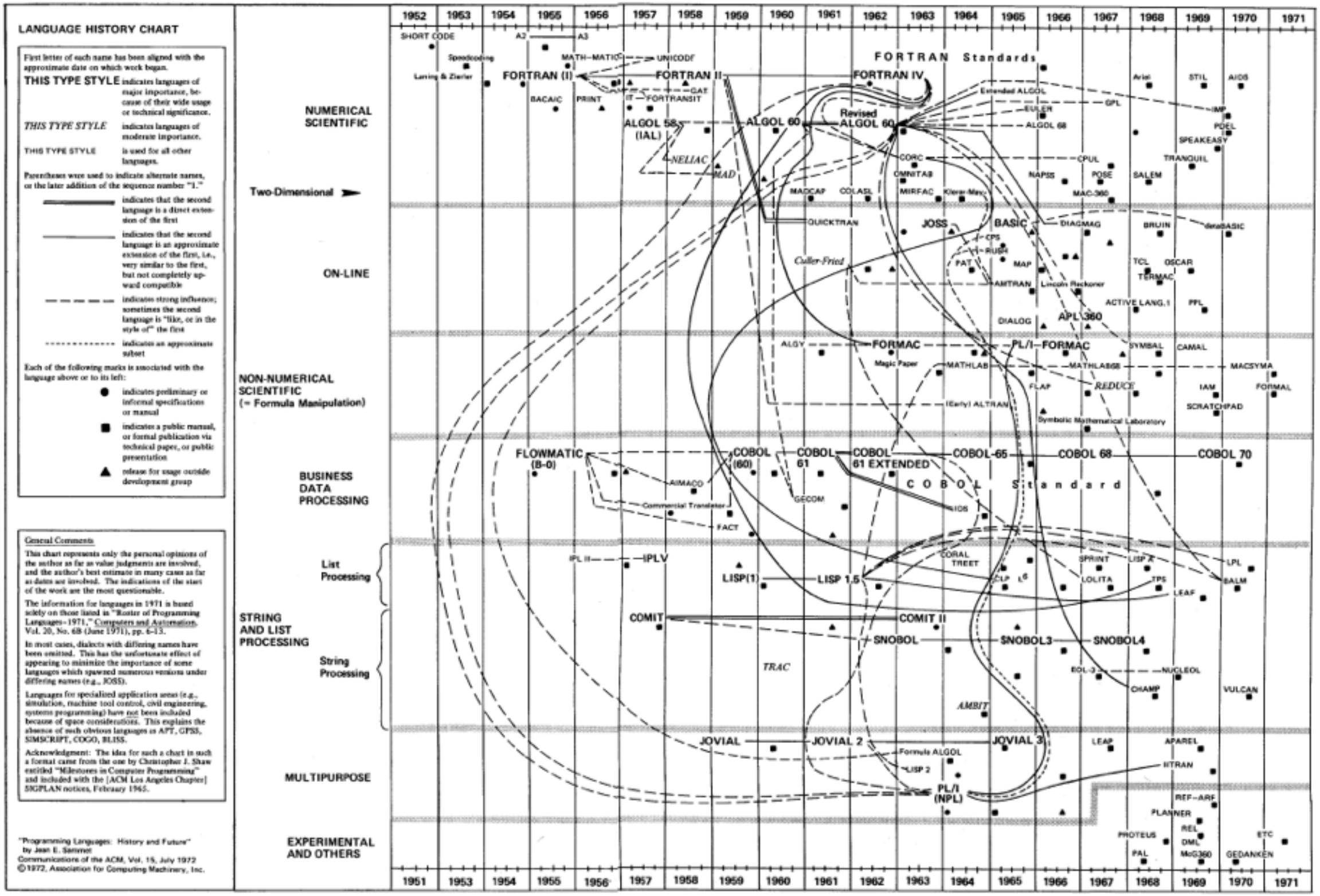
Peter Pan

Agenda

- Background
- FORTRAN (1957)
- LISP (1959)
- ALGOL (1960)
- COBOL (1960)
- Conclusions
- Bibliography

"Virtually everyone agrees that today programming is an art, not a science. Many -- although somewhat fewer -- people would contend that programming can (and should) be made into a science."

Jean Sammet, [2]


LANGUAGE HISTORY CHART
First letter of each name has been aligned with the approximate date on which work began.
THIS TYPE STYLE indicates languages of major importance, because of their wide usage or technical significance.
THIS TYPE STYLE indicates languages of moderate importance.
THIS TYPE STYLE is used for all other languages.
Parentheses were used to indicate alternate names, or the later addition of the sequence number "1."
indicates that the second language is a direct extension of the first
indicates that the second language is an approximate extension of the first, i.e., very similar to the first, but not completely upward compatible
indicates strong influence; sometimes the second language is "like, or in the style of" the first
indicates an approximate subset
Each of the following marks is associated with the language above or to its left:
indicates preliminary or informal specifications or manual
indicates a public manual, or formal publication via technical paper, or public presentation
release for usage outside development group
General Comments
This chart represents only the personal opinions of the author as far as value judgments are involved, and the author's best estimate in many cases as far as dates are involved. The indications of the start of the work are the most questionable.
The information for languages in 1971 is based solely on those listed in "Roster of Programming Languages–1971," Computers and Automation, Vol. 20, No. 6B (June 1971), pp. 6–13.
In most cases, dialects with differing names have been omitted. This has the unfortunate effect of appearing to minimize the importance of some languages which spawned numerous versions under differing names (e.g., JOSS).
Languages for specialized application areas (e.g., simulation, machine tool control, civil engineering, systems programming) have not been included because of space considerations. This explains the absence of such obvious languages as APT, GPSS, SIMSCRIPT, CDGO, BLISS.
Acknowledgment: The idea for such a chart in such a format came from the one by Christopher J. Shaw entitled "Milestones in Computer Programming" and included with the [ACM Los Angeles Chapter] SIGPLAN notices, February 1965.
"Programming Languages: History and Future" by Jean E. Sammet
Communications of the ACM, Vol. 15, July 1972
©1972, Association for Computing Machinery, Inc.
NUMERICAL SCIENTIFIC
Two-Dimensional
ON-LINE
NON-NUMERICAL SCIENTIFIC (= Formula Manipulation)
BUSINESS DATA PROCESSING
STRING AND LIST PROCESSING
List Processing
String Processing
MULTIPURPOSE
EXPERIMENTAL AND OTHERS
1952 1953 1954 1955 1956 1957 1958 1959 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971
SHORT CODE
Speedcoding
Laning & Zierler
A2
A3
FORTRAN (I)
MATH-MATIC
UNICODE
FORTRAN II
BACAIC
PRINT
IT
FORTRANSIT
GAT
ALGOL 58 (IAL)
ALGOL 60
NELIAC
MAD
FORTRAN IV
FORTRAN Standards
Revised ALGOL 60
Extended ALGOL
EULER
GPL
ALGOL 68
Ariel
STIL
AIDS
IMP
PDEL
SPEAKEASY
TRANQUIL
SALEM
CORC
CPUL
OMNITAB
NAPSS
POSE
MAC-360
MADCAP
COLASL
MIRFAC
Klerer-May
QUICKTRAN
JOSS
BASIC
DIAGMAG
BRUIN
dataBASIC
CPS
RUSH
PAT
MAP
Culler-Fried
TCL
OSCAR
TERMAC
AMTRAN
Lincoln Reckoner
ACTIVE LANG.I
PPL
DIALOG
APL\360
ALGY
FORMAC
PL/I–FORMAC
SYMBAL
CAMAL
Magic Paper
MATHLAB
MATHLAB68
MACSYMA
FLAP
REDUCE
IAM
FORMAL
SCRATCHPAD
(Early) ALTRAN
Symbolic Mathematical Laboratory
FLOWMATIC (B-0)
COBOL (60)
COBOL 61
COBOL 61 EXTENDED
COBOL-65
COBOL 68
COBOL 70
COBOL Standard
AIMACO
Commercial Translator
FACT
GECOM
IOCS
IPL II
IPLV
LISP(1)
LISP 1.5
CORAL
TREET
CLP
L6
SPRINT
LISP A
LPL
LOLITA
TPS
BALM
LEAF
COMIT
COMIT II
SNOBOL
SNOBOL3
SNOBOL4
TRAC
EOL-3
NUCLEOL
CHAMP
VULCAN
AMBIT
JOVIAL
JOVIAL 2
JOVIAL 3
Formula ALGOL
LISP 2
PL/I (NPL)
LEAP
APAREL
ITRAN
REF-ARF
PLANNER
REL
PROTEUS
DML
ETC
PAL
MoG360
GEDANKEN

J.E. Sammet, [2]

State of the Art <1954

- UNIVAC Short Code (William Schmitt)
- 1st compiler "A-0" developed for UNIVAC in 1952 at Remington-Rand (Grace M. Hopper)

“The programmer has been supplied with a "code" into which he translates his instructions to the computer...until the novelty of inventing programs wears off and degenerates into the dull labor of writing and checking programs.

This duty now looms as an imposition on the human brain.”

Grace Murray Hopper [1]

The goal of early programming languages was to overcome the limitations of early computers and their (octal) op codes, such as:

- Lack of floating point computation
- Lack of sufficient (hardware) registers
- Primitive input/output operations
- Restrictive (hardware) instruction sets (Logical AND support but not OR)

Backus, [4]

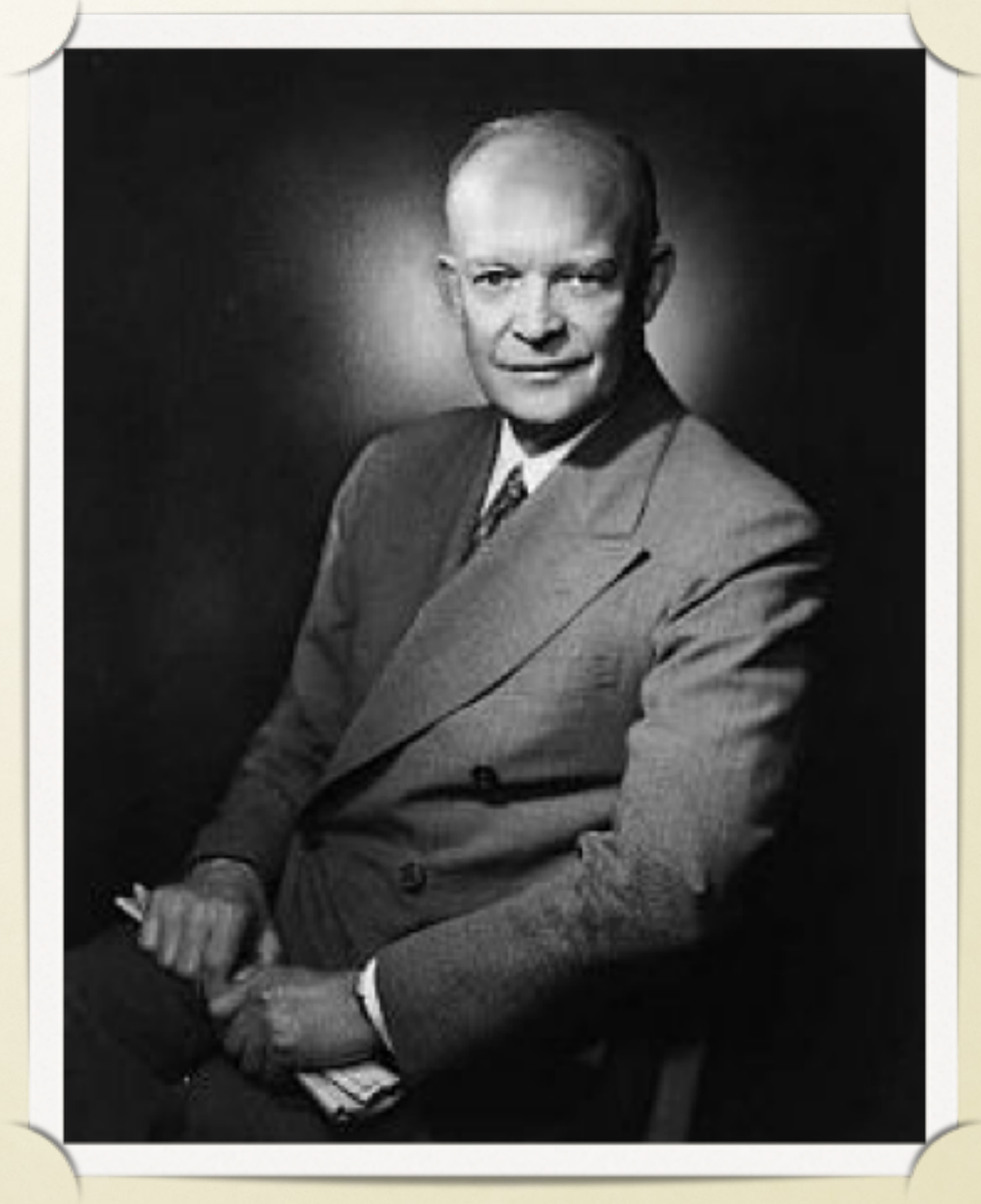
What did "automatic programming" mean to contemporaries in 1954:

- Providing mnemonic operational codes (instead of octal)
- Symbolic addresses (instead of physical memory locations)
- Library subroutines
- Floating point operations

Backus [4]

1954

- Dwight Eisenhower is the President of the United States.
- First children given polio vaccine.
- *USS Nautilus* (1st atomic submarine) commissioned.



Library of Congress

Fortran

Dr. Cuthbert Hurd
(shown with T.J. Watson, seated)
IBM Archives, [3]

IBM

IBM archives, [3]

John Backus

- Hated "programming"
- Invented "Speedcoding"
- Lead FORTRAN team 1954-57
- Invented "Backus Normal Form" (BNF) to formally describe ALGOL grammar, 1959
- National Medal of Science, 1975
- ACM Turing Award, 1977

“Early systems were slow and expensive. Through experience, programmers began to doubt writing efficient programs could be automated.

Also some marketing departments claimed their systems could have human-level lexical understanding of programmer intentions.”

John Backus, [4]

"One was accustomed to finding lots of peculiar but significant restrictions in a system when it finally arrived that had not been mentioned in its original description."

John Backus [4]

“[I]t is difficult to convey … the strength of skepticism about 'automatic programming' in general and about its ability to produce efficient programs in particular as it existed in 1954.”

John Backus [4]

Fortran Design Goals

- "Virtually eliminate coding and debugging..."
- Make programming "faster, cheaper [and] more reliable."
- Serve as an implementation for others to copy.

"It will suffice to say that [the compiler] produced code of such efficiency that its output would startle programmers who studied it… The degree of optimization performed… was not equaled again until optimizing compilers began to appear in the middle and late 1960s."

John Backus [4]

“It was an exciting period; we were often astonished at the amazing transformations which made the program efficient but which we would not have thought to make as programmers ourselves.”

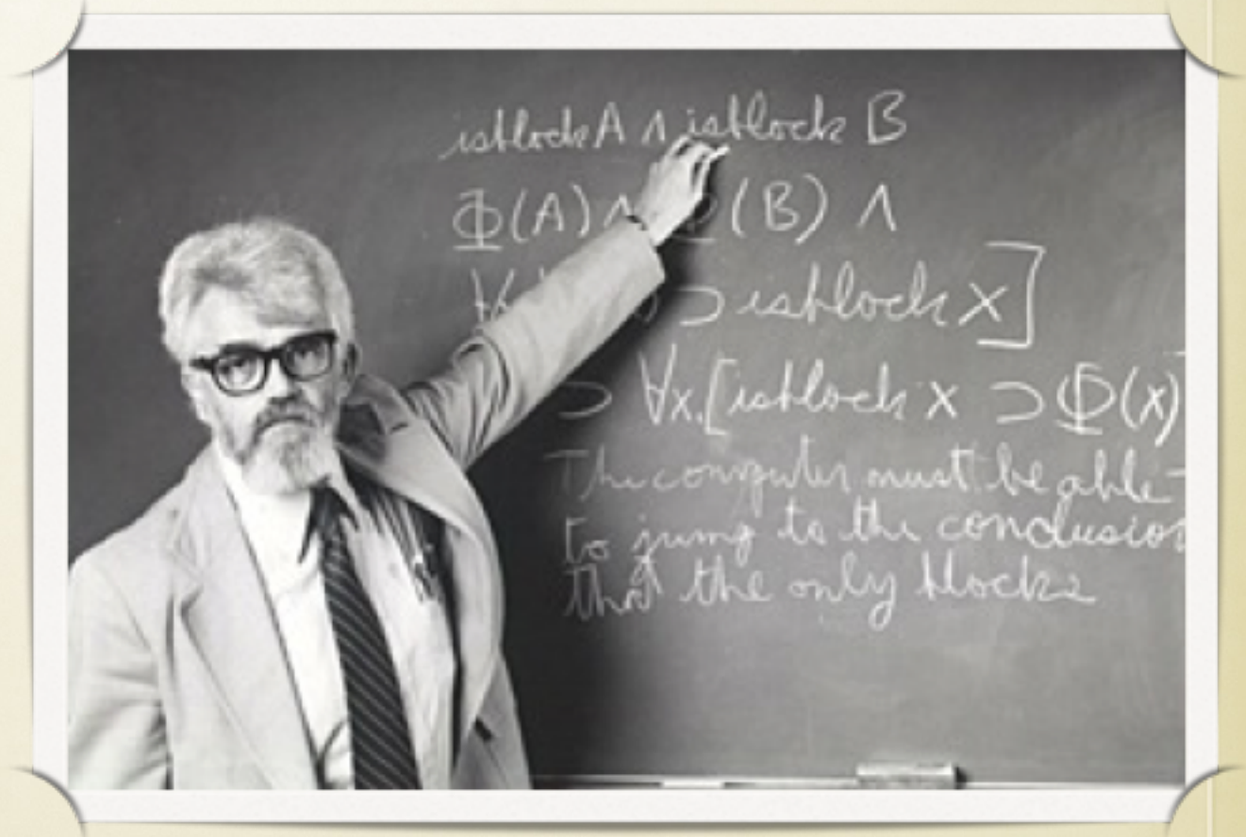
John Backus [4]

- Weak debugging facilities
- Poor design choices
- Project timeline estimation problems
- Other flaws included whitespace issues (ignoring blanks, even in the middle of identifiers), disallowing mixed integer and floating point expressions.
- Cost savings and efficiency gains are eaten up by increases in program size and complexity.

LISP

John McCarthy

- Invented LISP, 1959
- Worked on ALGOL, 1958-1963 (if/then/else)
- Invented the term and technique of "garbage collection", 1959
- Coined the term "artificial intelligence."
- Predicted "computing utilities" (aka cloud computing), 1961.
- ACM Turing Award, 1971
- National Medal of Science, 1990



“LISP seems to be the second oldest surviving programming language after Fortran, so maybe we should plan on holding one of these newspaper interviews in which grandpa is asked to what he attributes having lived to 100.”

John McCarthy [7]

LISP Design Goals

- Enable symbolic reasoning (instead of explicit numerical computation)
- Representation of data and expressions as lists in memory
- Functional composition and recursion as programming idioms
- Automatic memory management

- Used Church's λ notation as a means to give a function a name which could be passed as an argument to other functions. [6]
- Developed garbage collection to deal with the "erasure" problem. (This is the answer to the question of how to reclaim memory occupied by the remnants of a list after applying a *car* or *cdr* function.)
- Steve Russell hand coded the *eval* function (defined in [5]) in machine code for the IBM 704 which served as an interpreter for the language.

“Writing eval required inventing a notation representing LISP functions as LISP data... with no thought that it would be used to express LISP programs in practice.”

John McCarthy [6]

- Interpreted programs ran about 60 times slower than compiled programs.
- No lexical scope for variable bindings (that came in LISP 1.5)
- McCarthy planned that the awkwardness of the S-expression syntax would be solved by the use of "M-expressions" in an ALGOL like syntax (but that goal "receded into the indefinite future." [6])

"[W]e had the intention of producing a compiler. But the FORTRAN people were expressing their shock at having spent 30 man-years on [a compiler]... and it didn't seem like these graduate students would hold still long enough that we could get 30 man-years of work out of them."

John McCarthy [6]

“LISP has jokingly been described as 'the most intelligent way to misuse a computer'. I think that description a great compliment because it transmits the full flavour of liberation: it has assisted a number of our most gifted fellow humans in thinking previously impossible thoughts.”

E.W. Dijkstra [8]

ALGOL

ALGOL Design Goals

- Create a language "as close a possible to standard mathematical notation and be readable with little further explanation." [11]
- "Use [this language] for the description of computing processes in publications." [11]
- Advance a "universal" agreed-upon language without worrying about specific implementation details.

Alan Perlis

- Original member of the U.S. ALGOL delegation.
- Considered to be a pioneer in promoting computer science as a discipline distinct from mathematics.
- President of ACM, 1962
- ACM Turing Award, 1966

“There are two ways to write error-free programs; only the third one works.”

Alan Perlis, [10]

“Some programming languages manage to absorb change, but withstand progress.”

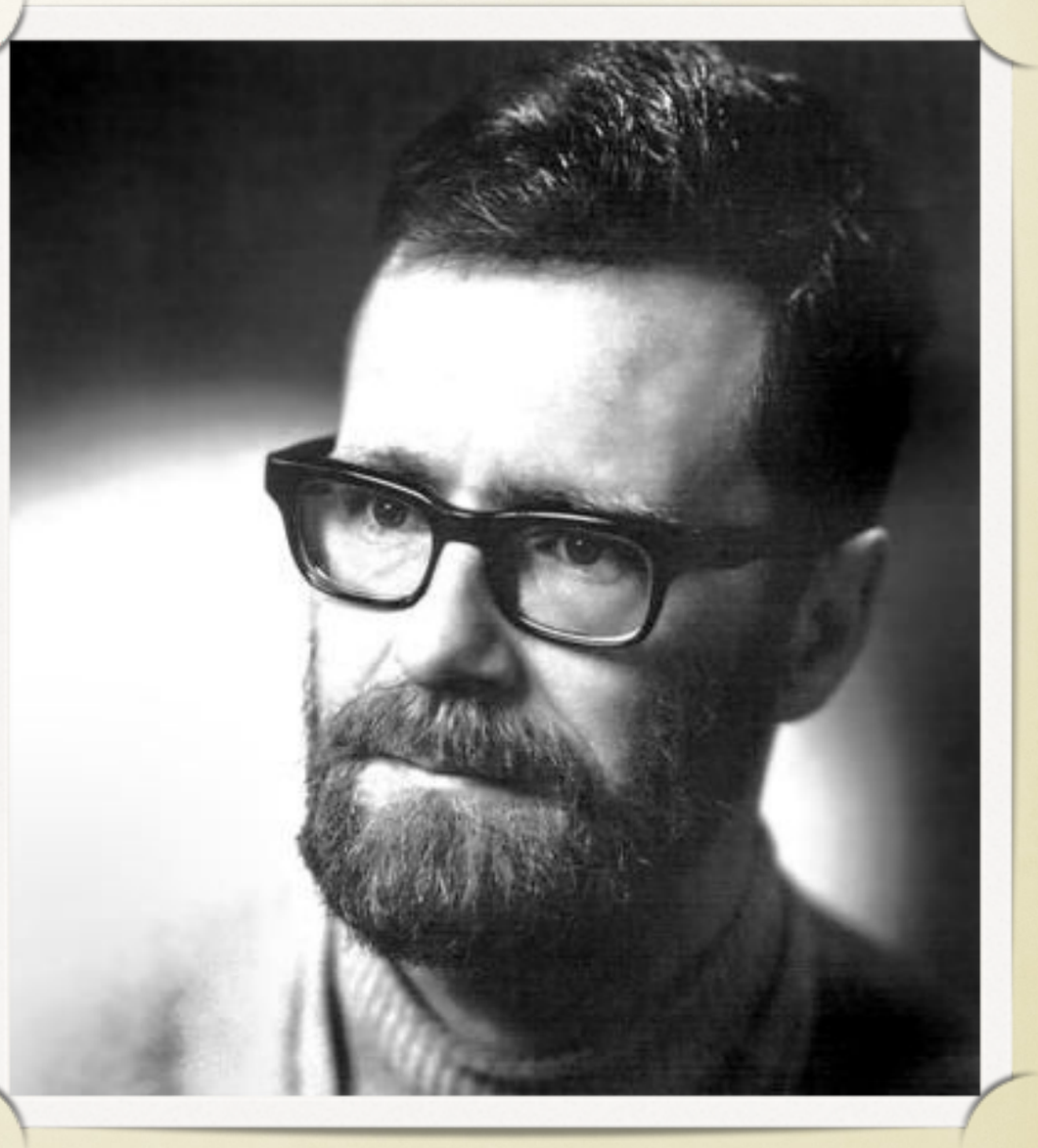
Alan Perlis, [10]

Peter Naur

- Editor of ALGOL Bulletin
- Refined Backus Normal Form grammars (as part of the ALGOL 1960 report)
- One of the 1st ALGOL compiler implementations
- ACM Turing Winner, 2005

Edsger W. Dijkstra

- Best known for his graph traversal algorithm
- Foundational paper in concurrent computing (invented the mutex)
- "GO TO considered harmful"
- 1st implementation of ALGOL 1960
- ACM Turing Winner, 1972



ALGOL Firsts

- Recursion
- Code blocks with lexically scoped variables
- Call by value and call by name (an early form of "lazy" evaluation)
- An intrinsic boolean type
- Keywords "modern" readers find familiar
- Formally specified grammar

Sammet, [12]

“This language proved to be an object of stunning beauty.”

Alan Perlis, [9]

“Neither [Backus'] nor any other adequate formal semantic definition technique was available to the ALGOL 60 designers. We now know that semantics is considerably more difficult to treat than syntax.”

Alan Perlis, [9]

- ALGOL was never very popular in the United States.
- Failed to replace FORTRAN (as indeed, it is still around today!)
- Semantic meaning of the formal grammar was open to interpretation.
- Opened new avenues of inquiry in computer science. (Such as [13])

COBOL

COBOL Design Goals

- Desired a single programming language that worked across multiple manufacturers' computers.
- Maximum use of English (instead of mathematical notation.)
- "Easy to use" even if that meant the language would be less expressive.
- Broaden the base of who can state problems to computers.

Grace Murray Hopper

- PhD, Mathematics, Yale, 1934.
- Retired US Navy as Rear Admiral, 1986.
- Famously visualized a nanosecond [14]
- Popularized the term "debugging"
- Lead "automatic programming" team at Remington-Rand
- "FLOWMATIC" was a major source for the design of COBOL [15]

Jean E. Sammet

- M.A. Mathematics, Illinois, 1949.
- Worked at Sperry Rand 1955-58 on UNIVAC I
- Worked on COBOL for Sylvania 1958-61
- Joined IBM 1961 and developed FORMAC

"It was quite clear that the disputes between people from the same organization were often as great (and sometimes greater) than between people from different organizations, and this applied to virtually all the organizations involved, whether manufacturers or users."

Jean Sammet, [15]

"[T]here was a strong anti-IBM bias in this committee from me and some (but certainly not all) of the others. Since I was not working for IBM at the time, I can freely (although not *with pride) admit that in some cases suggestions or decisions were made on the basis of doing something differently than IBM did it."*

Jean Sammet, [15]

“Experience with real users and real compilers is desperately needed before 'freezing' a set of language specifications. Had we known from the beginning that COBOL would have such a long life, I think many of us would have refused to operate under the time scale that was handed to us…”

Jean Sammet, [15]

- Introduced the notion of a typed record or data definition distinct from executable code.
- Was at one point the single most widely used programming language. COBOL still in use at many companies today.
- Met its design goal of implementation independence (largely due to influence of US government contract requirements.)

Conclusions

- Many early practitioners expected problems related to programmer error and efficiency to be solved by high level computer languages.
- Even when it was an explicit goal to reduce what today we might term "cognitive load" in a particular programming language, it proved difficult to achieve satisfactory results in practice.
- I would strongly recommend browsing through the bibliography and reading one or two of these papers.

THANK YOU!

Mark Allen
mrallen1@yahoo.com
@bytemeorg

Bibliography

- [1] Hopper, Grace Murray, The education of a computer, Proceedings of the ACM National Conference, 1952. http://dl.acm.org/citation.cfm?id=609818
- [2] Sammet, Jean, Programming languages: history and future, Commications of the ACM, July, 1972. http://dl.acm.org/citation.cfm?id=361454.361485

Bibliography

- [3] FORTRAN, The Pioneering Computer Language, http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/fortran/
- [4] Backus, John, The History of FORTRAN I, II, III, History of Programming Languages I, ACM. http://dl.acm.org/citation.cfm?id=800025.1198345

Bibliography

- [5] McCarthy, John, Recursive functions of symbolic expressions and their computation by machine, Part I, Communications of the ACM, 1960. http://dl.acm.org/citation.cfm?id=367177.367199

- [6] McCarthy, John, History of LISP, History of Programming Languages I, ACM, 1978. http://dl.acm.org/citation.cfm?id=800025.1198360

Bibliography

- [7] McCarthy, John, LISP - notes on its past and future, Proceedings of the 1980 ACM conference on LISP and functional programming, ACM, 1980. http://dl.acm.org/citation.cfm?id=800087.802782

- [8] Dijkstra, Edsger, W., The humble programmer, Communications of the ACM, 1972. http://www.cs.utexas.edu/users/EWD/ewd03xx/EWD340.PDF

Bibliography

- [9] Perlis, Alan, The American side of the development of ALGOL, History of Programming Language I, ACM, 1978. http://dl.acm.org/citation.cfm?id=800025.1198352

- [10] Perlis, Alan, Epigrams on programming, ACM SIGPLAN Notices, September 1982, http://dl.acm.org/citation.cfm?id=947955.1083808

Bibliography

- [11] Naur, Peter, The European side of the last phase of the development of ALGOL 60, History of Programming Languages I, ACM, 1978. http://dl.acm.org/citation.cfm?id=800025.1198353

- [12] Sammet, Jean, Programming Languages: History and Fundamentals, Addison-Wesley, 1967.

Bibliography

- [13] Irons, Edgar, A syntax directed compiler for ALGOL 60, Communications of the ACM, 1961. http://dl.acm.org/citation.cfm?id=366062.366083
- [14] Hopper, Grace Murray, video clip explaining the length of a nanosecond and a microsecond. https://www.youtube.com/watch?v=JEpsKnWZrJ8

Bibliography

- [15] Sammet, Jean, The early history of COBOL, History of Programming Languages I, ACM, 1978. http://dl.acm.org/citation.cfm?id=800025.1198367