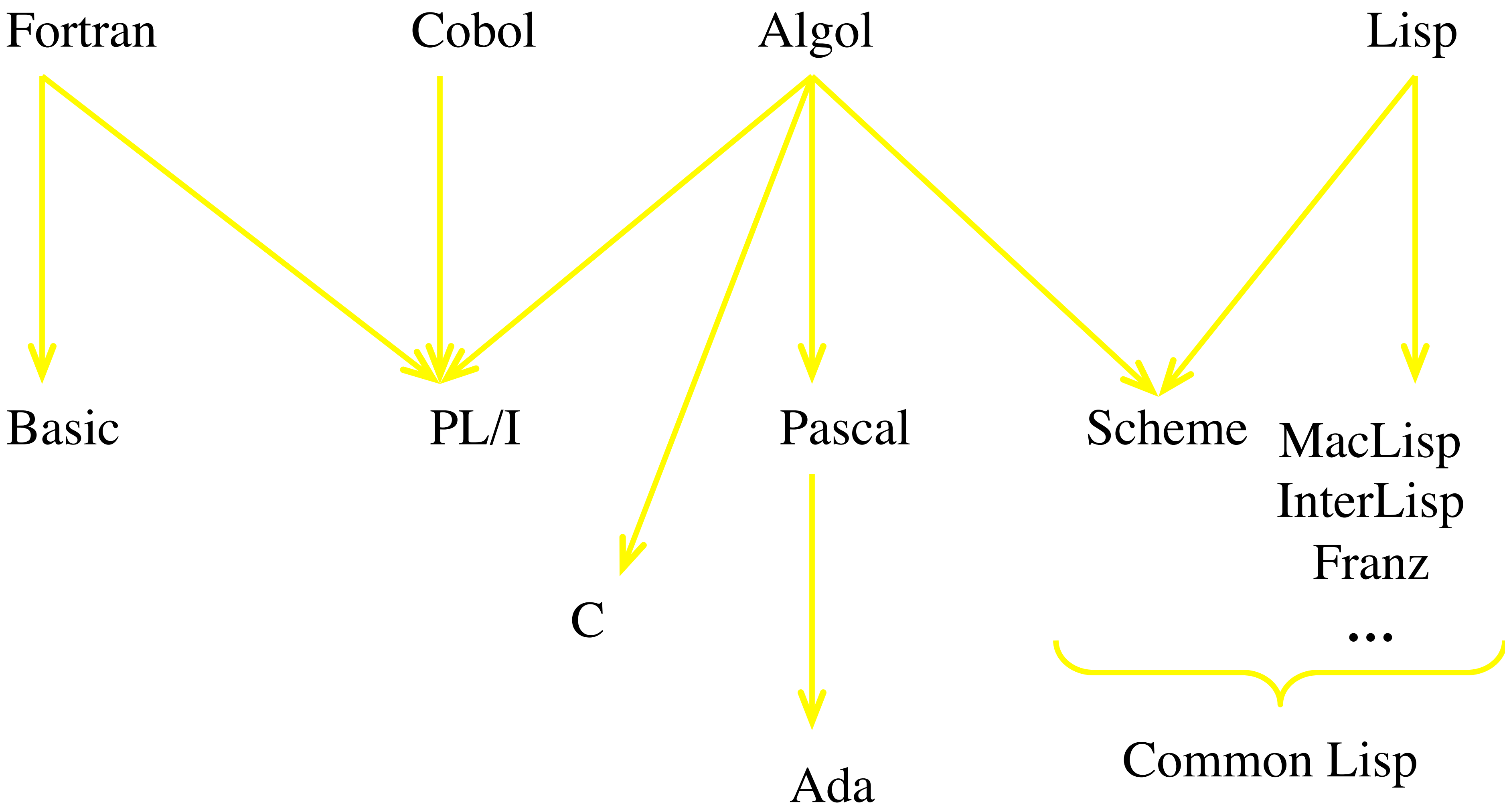
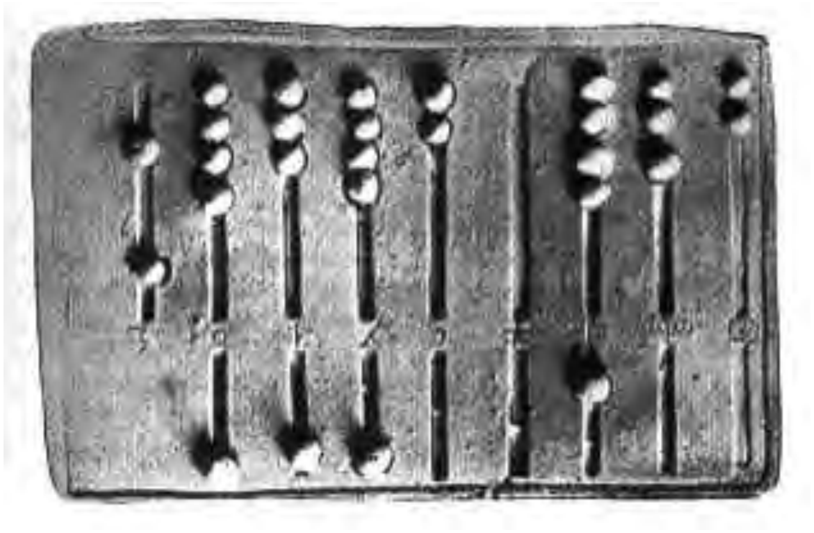
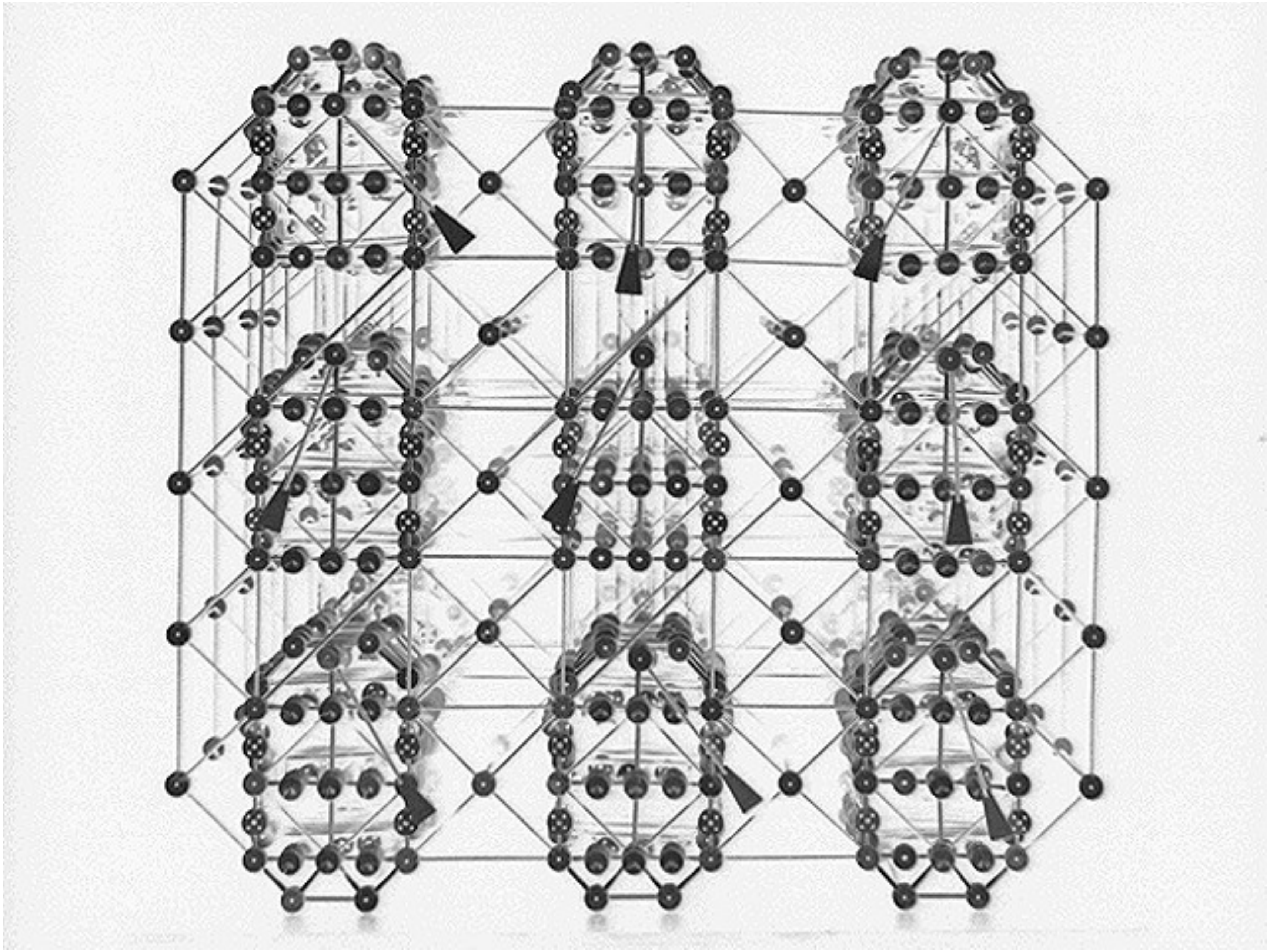
A Biased History of Programming Languages

Programming Languages: A Short History





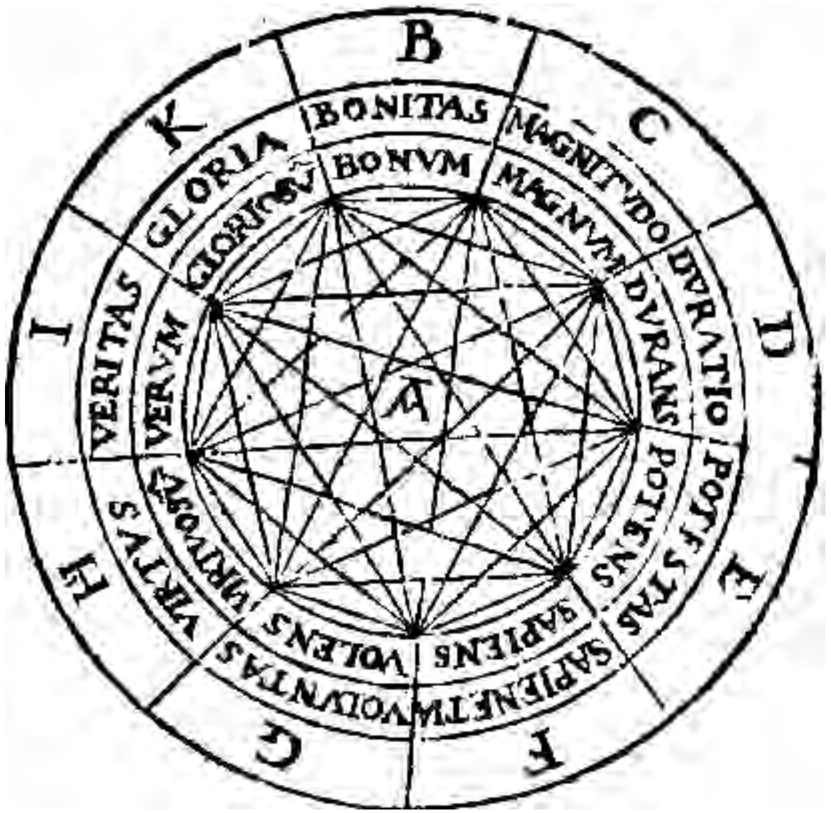
Roman Hand-Abacus. Image is from Museo (Nazionale Ramano at Piazzi delle Terme, Rome)



History

- Pre-History : The first programmers
- Pre-History : The first programming languages
- The 1940s: Von Neumann and Zuse
- The 1950s: The First Programming Language
- The 1960s: An Explosion in Programming languages
- The 1970s: Simplicity, Abstraction, Study
- The 1980s: Consolidation and New Directions
- The 1990s: Internet and the Web
- The 2000s: Constraint-Based Programming

Ramon Lull (1274)



Raymondus Lullus
Ars Magna et Ultima

Gottfried Wilhelm Freiherr von Leibniz (1666)

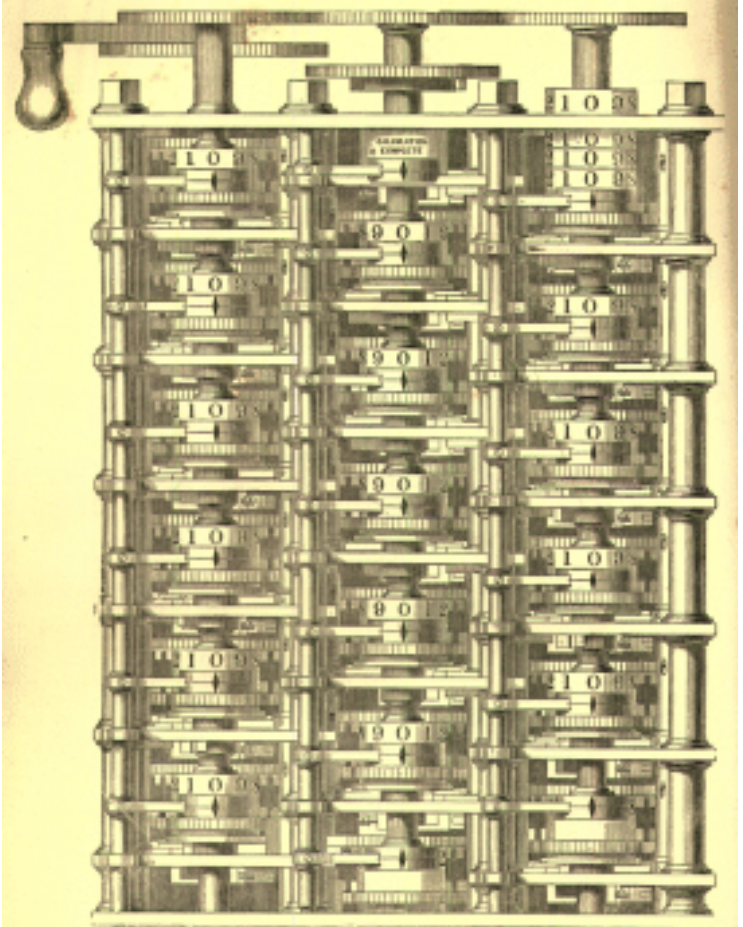
The only way to rectify our reasonings is to make them as tangible as those of the Mathematician, so that we can find our error at a glance, and when there are disputes among persons, we can simply say: Let us calculate, without further ado, in order to see who is right.

Charles Babbage

- English mathematician
- Inventor of mechanical computers:
  - Difference Engine, construction started but not completed (until a 1991 reconstruction)
  - Analytical Engine, never built

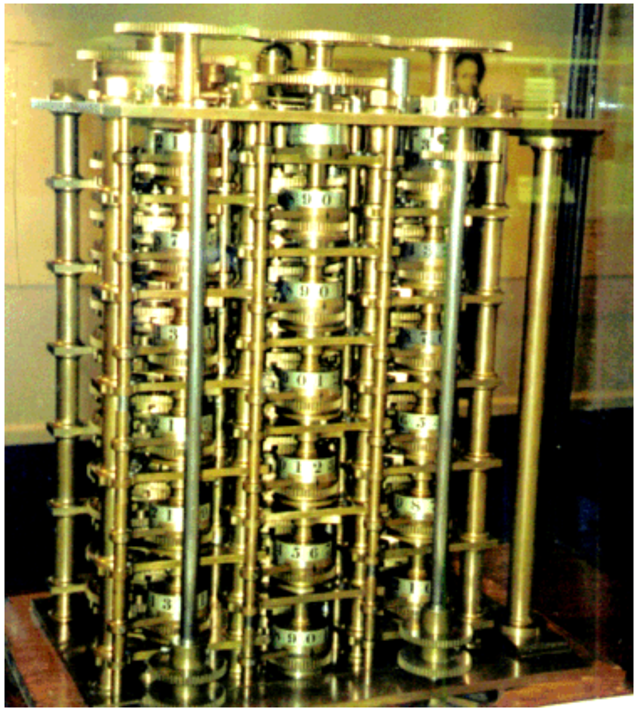
I wish to God these calculations had been executed by steam!

Charles Babbage, 1821

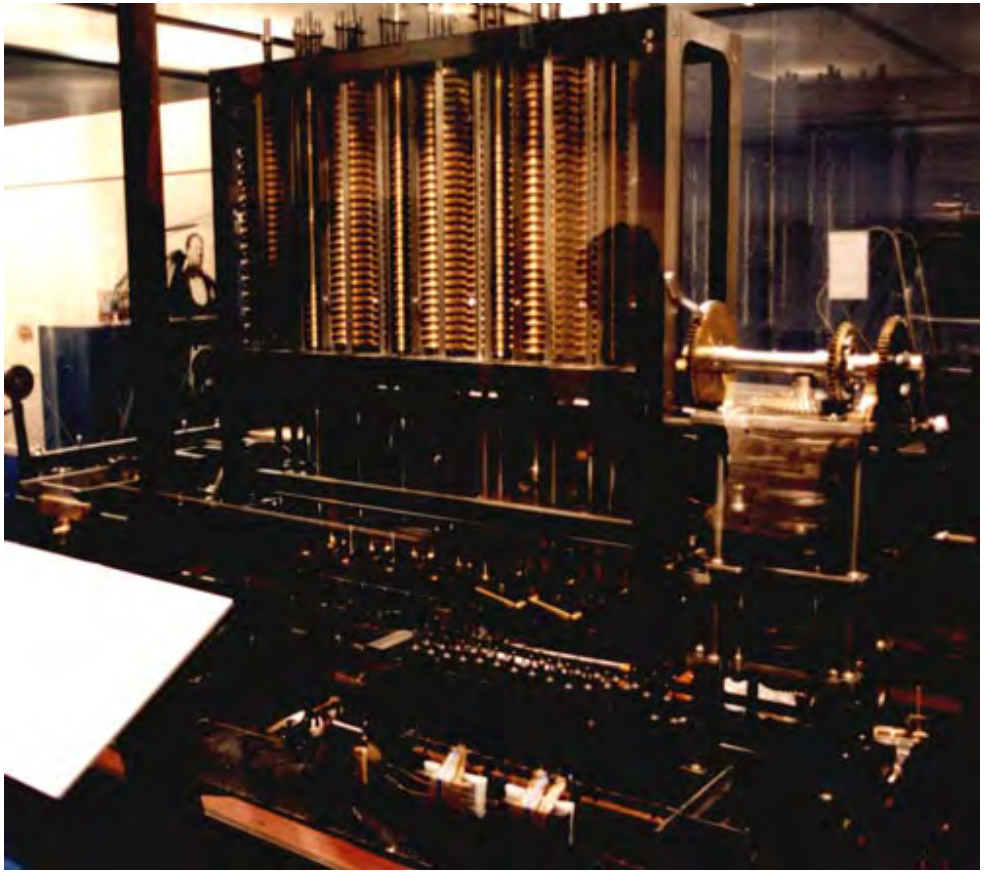

B. H. Babbage, del.

Impression from a woodcut of a small portion of Mr. Babbage's Difference Engine No. 1, the property of Government, at present deposited in the Museum at South Kensington.

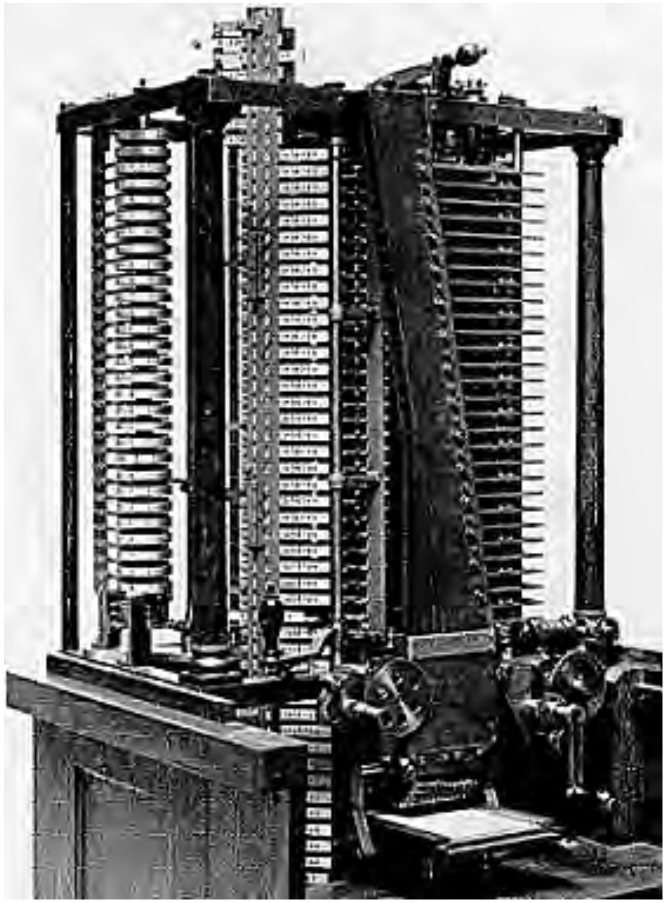
It was commenced 1823.
This portion put together 1833.
The construction abandoned 1842.
This plate was printed June, 1853.
This portion was in the Exhibition 1862.



Difference Engine No.1
Woodcut of a small portion of Mr. Babbage's Difference Engine No.1, built 1823-33. Construction was abandoned 1842.



Difference Engine.
Built to specifications 1991. It has 4,000 parts and weighs over 3 tons. Fixed two bugs.



Portion of Analytical Engine (Arithmetic and Printing Units). Under construction in 1871 when Babbage died; completed by his son in 1906. The first program was to calculate and print the first 25 multiples of π to 29 decimal places. (Science Museum, London.)

Augusta Ada

- Daughter of George Gordon, Lord Byron
- Early 1800's in England (as elsewhere) women were generally denied education, especially math and science
- Ada studied math with a private tutor (as an antidote to feared Byronic tendencies)
- Married at 19 (Lady Lovelace), 3 children

Analytical Engine

- Processing unit (the Mill)
- Memory (the Store)
- Programmable (punched cards; cf. Jacquard loom)
- Iteration, conditional branching, pipelining, many I/O devices

Ada Lovelace (1843)

"The engine can arrange and combine its numerical quantities exactly as if they were letters or any other general symbols; and in fact might bring out its results in algebraic notation, were provision made."

[The Analytical Engine is for] developing and tabulating any function whatever.... The engine [is] the material expression of any indefinite function of any degree of generality and complexity.

The Analytical Engine has no pretensions to *originate* anything. It can do *whatever we know how to order it* to perform.

A A L

- In 1840 Babbage was invited to give a seminar at the University of Turin about his analytical engine.
- Luis Menabrea, a young Italian engineer wrote up Babbage's lecture in French, and this transcript was subsequently published in the Bibliothèque Universelle de Genève in 1842.
- Babbage asked Ada to translate Menabrea's paper into English.
- Babbage then asked Ada to augment the notes she had added to the translation, and she spent most of a year doing this.
- These notes, which are more extensive than Menabrea's paper, were published in *The Ladies Diary* and *Taylor's Scientific Memoirs* (under the intialism A.A.L.).
- In Note G, Ada describes an algorithm for the analytical engine to compute Bernoulli numbers.
- It is generally considered the first algorithm ever specifically tailored for implementation on a computer, and for this reason she is considered by many to be the first computer programmer.

Not Just For Numbers

The bounds of *arithmetic* were however outstepped the moment the idea of applying the cards had occurred; and the Analytical Engine does not occupy common ground with mere "calculating machines." … In enabling mechanism to combine together *general* symbols in successions of unlimited variety and extent, a uniting link is established between the operations of matter and the abstract mental processes of the *most abstract* branch of mathematical science.

A.A.L.

Axel Thue (1910)

$ab \rightarrow bbaa$

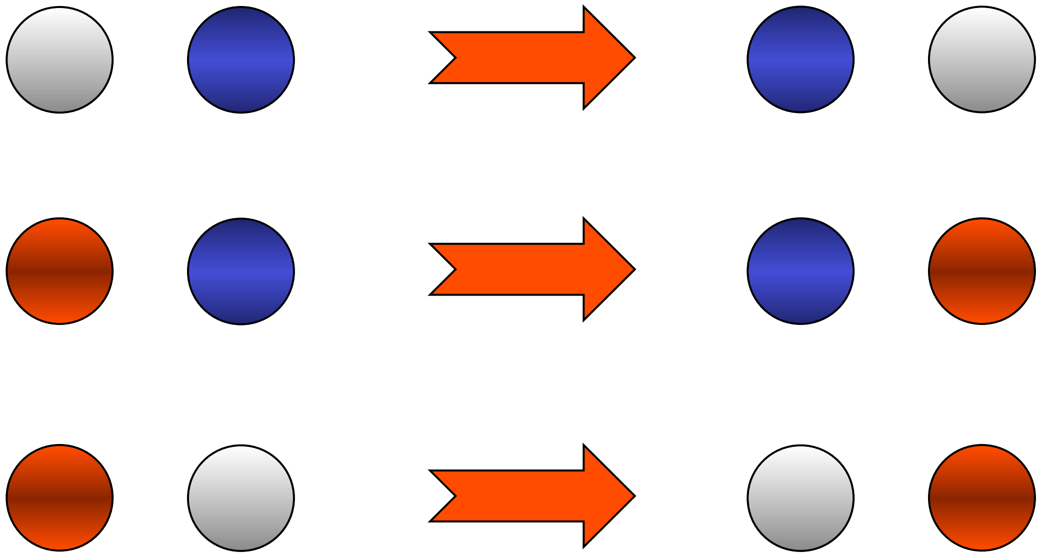
A. Markov

01 → 1100

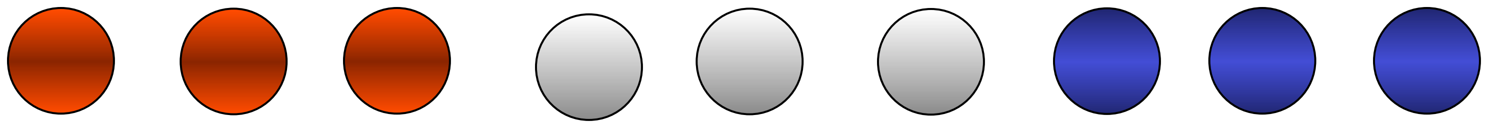
А. А. Марков (1886).

Marbles

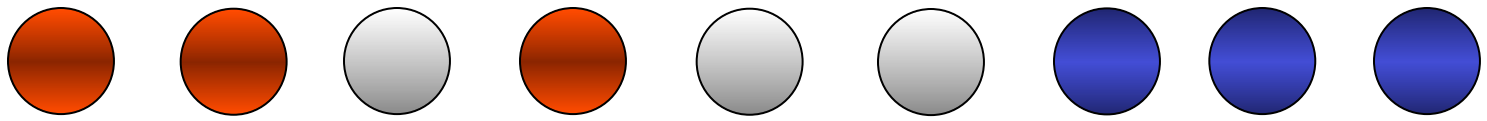
Flag Problem



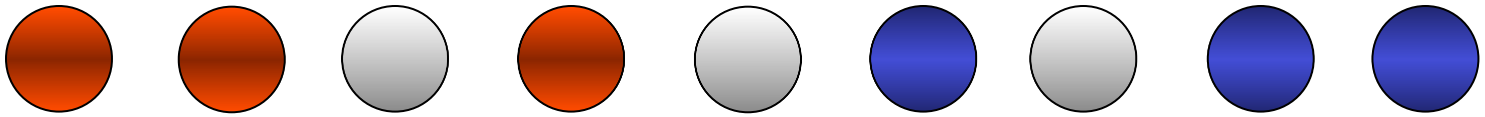
Dutch National Flag



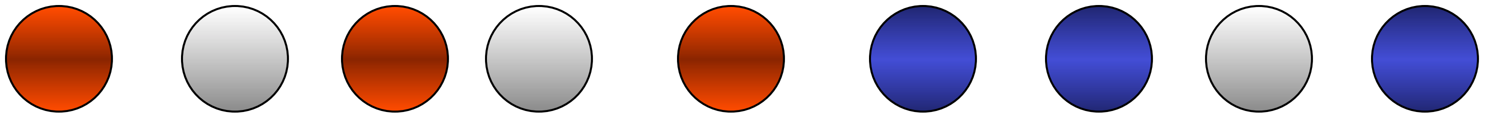
Dutch National Flag



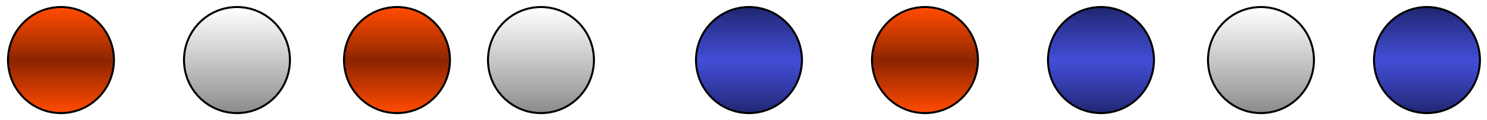
Dutch National Flag



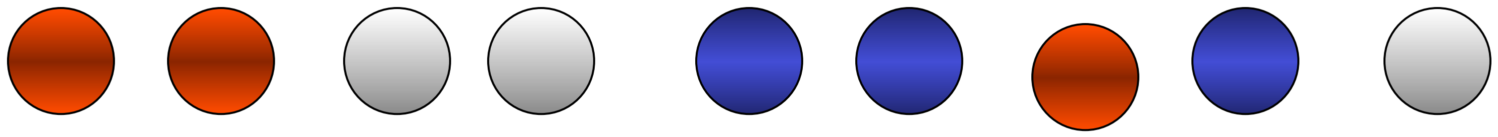
Dutch National Flag



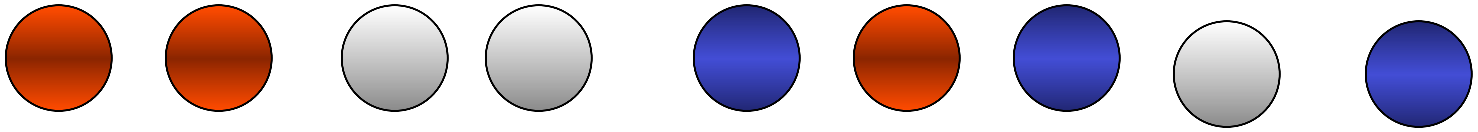
Dutch National Flag



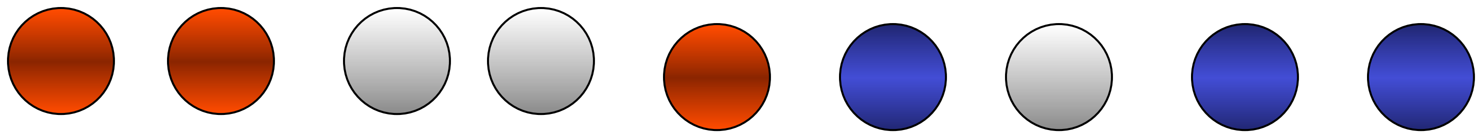
Dutch National Flag



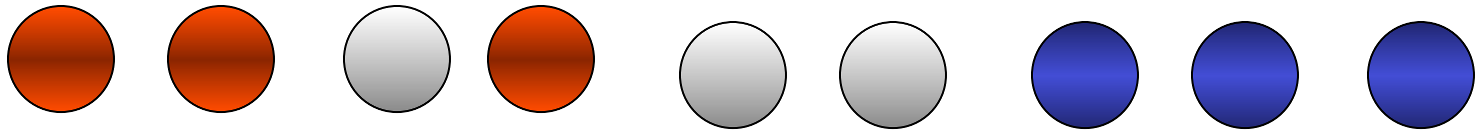
Dutch National Flag



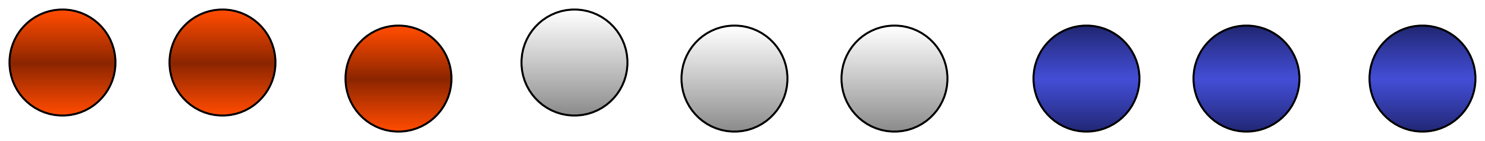
Dutch National Flag



Dutch National Flag



Dutch National Flag



λx. (x x)

Alonzo Church
(1941)

$$f(n) = \begin{cases} 1 & n=0 \\ n \times f(n-1) & \text{otherwise} \end{cases}$$

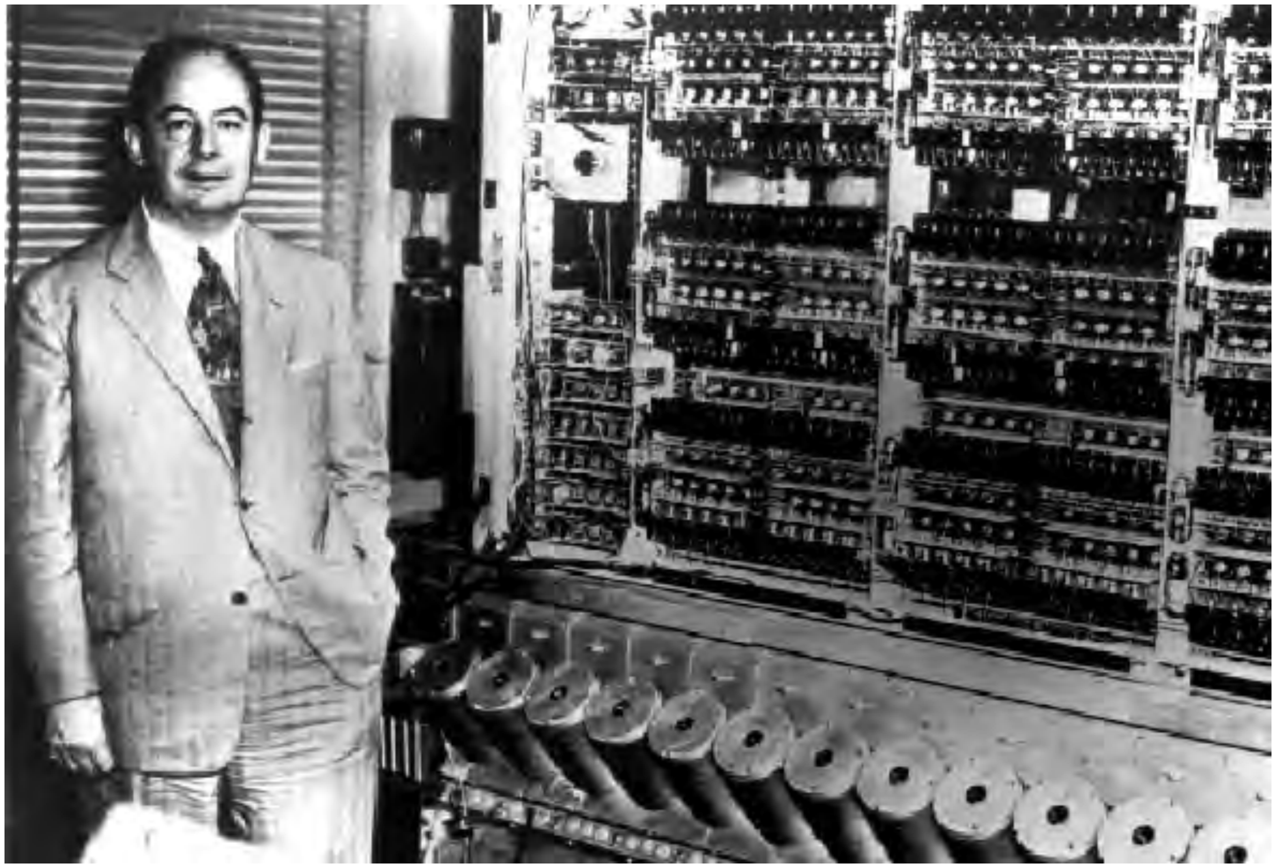
Kurt Gödel
(1930s)

Alan Turing (1936)

Von Neumann

John Von Neumann led a team that built computers with stored programs and a central processor

ENIAC, however, was also programmed with patch cords.



Von Neuman with ENIAC

Konrad Zuse

Konrad Zuse began work on Plankalkul (plan calculus), the first algorithmic programming
language, with an aim of creating the theoretical preconditions for the formulation of problems of a general nature.

Konrad Zuse

- Built a mechanical computer in his parents' living room in Berlin in 1936: the Z1
- Metal strips and pins—very different from Babbage's wheelwork
- Programmable using punched tapes
- Binary floating point numbers with an explicit exponent

Konrad Zuse

- In Germany - in isolation because of the war
- Defined Plankalkul (program calculus) circa 1945 but never implemented it.
- His work finally published in 1972.
- Wrote algorithms in the language, including sorting, graphs, numeric algorithms, syntax analysis, ***chess***
- Many advanced ideas:
  - Assignment, expressions, subscripts
  - Constructed types: from primitive (bit) other types are constructed: integers, reals, ***arrays***, etc.
  - Conditional execution, loops, subroutines
  - Assertions; type assertions
  - Floating point, used 2s complement and hidden bits
  - Records (that could be nested)

Early Development

- More computers:
  - Z2 experimented with relays for the ALU
  - Z3: all-relay technology, 1941 (the first fully functional program-controlled electromechanical digital computer)
  - Z4: envisioned as a commercial system
- Most designs and prototypes destroyed in the war
  - Only the Z4 survived WWII.
  - Zuse flees Berlin in 1945 with wife and Z4

Machine Code (1940s)

- Initial computers were programmed in raw machine code.
- These were entirely numeric.
- What was wrong with using machine code? Everything!
    - Poor readability
    - Poor modifiability
    - Expression coding was tedious
    - Inherit deficiencies of hardware, e.g., no indexing or floating point numbers

The 1950s:
The First Programming Languages

- Fortran: first higher level programming language
- COBOL: first business oriented language
- Algol: one of the most influential programming languages ever designed
- Lisp: first functional language
- APL: abstruse functional language with powerful matrix operations

John Backus

- Many contributions to programming languages: Fortran, Algol 58 and 60, BNF, FP (a purely functional language)

My point is this: while it was perhaps natural and inevitable that languages like FORTRAN and its successors should have developed out of the concept of the von Neumann computer as they did, the fact that such languages have dominated our thinking for twenty years is unfortunate. It is unfortunate because their long-standing familiarity will make it hard for us to understand and adopt new programming styles which one day will offer far greater intellectual and computation power.

John Backus, 1978

Fortran

- The first popular high-level programming language
- A team led by John Backus at IBM
- "The IBM Mathematical FORmula TRANSlating System: FORTRAN", 1954:
  - supposed to take six months -- took two years
  - supposed to eliminate coding errors and debugging
  - supposed to generate efficient code, comparable with hand-written code -- very successful at this
  - closely tied to the IBM 704 architecture

Fortran (1954-57)

- FORmula TRANslator
- Developed at IBM under the guidance of John Backus primarily for scientific programming
- Dramatically changed forever the way computers used
- Has continued to evolve, adding new features & concepts.
  - FORTRAN II, FORTRAN IV, FORTRAN 66, FORTRAN 77, FORTRAN 90
- Always among the most efficient compilers, producing fast code
- Still popular, e.g. for supercomputers

Fortran I Features

- Loop (DO)
- Formatted I/O
- User-defined subprograms
- Three-way selection statement (arithmetic IF)

  IF (ICOUNT-1) 100, 200, 300
- Implicit typing

  variables beginning with i, j, k, l, m or n were integers, all else floating point

Separate Compilation

- First Fortran: no separate compilation
- Compiling “large” programs – a few hundred lines – was impractical, since compilation time approached 704 MTTF
- Fortran II added separate compilation
- Later Fortrans evolved with platform independence: no more **`PAUSE`** statement!

I don't know what the language of the year 2000 will look like, but I know it will be *called* FORTRAN.
C.A.R. Hoare

Fortran's Influence

- Many languages followed, but all designers learned from Fortran
- Fortran team pioneered many techniques of scanning, parsing, register allocation, code generation, and optimization

COBOL

COBOL

COmmon Business Oriented Language
Principal mentor: (Rear Admiral Dr.)
Grace Murray Hopper (1906-1992)

Design goals:

- Must look like simple English
- Must be easy to use, even if that means it will be less powerful
- Must broaden the base of computer users
- Must not be biased by current compiler problems

Design committee were all from computer manufacturers and DoD branches. Fights among manufacturers

COBOL

Contributions:

- First macro facility in a high-level language
- Hierarchical data structures (records)
- Nested selection statements
- Long names (up to 30 characters), with hyphens

Early AI Language Efforts

- An IBM group developed FLPL: Fortran List Processing Language
- McCarthy had a wish list, developed while writing AI programs (chess and differential calculus)
  - Conditional expressions
  - Recursion
  - Higher-order functions (like ML's **`map`**)
  - Garbage collection
- FLPL wasn't the answer for McCarthy's group at MIT in 1958…

Lisp

- AI conference at Dartmouth, 1956: McCarthy, Minsky, Newell, Simon
- Newell, Shaw and Simon demonstrate Logic Theorist, a reasoning program written in IPL (Information Processing Language)
- IPL had support for linked lists, and caught McCarthy's attention
- He wanted a language for AI projects, but not IPL: too low-level and machine-specific

Lisp's Unusual Syntax

- A Lisp program is a list representing an AST:
  ```
  (+ a (* b c))
  ```
- The plan was to use some Fortran-like notation
- But McCarthy wrote a paper showing a simple Lisp interpreter in Lisp: a function called **`eval`**
- To avoid syntax issues, he used the list-AST form, both for **`eval`**'s input and for **`eval`** itself
- This **`eval`**, hand-translated into assembly language, became the first implementation of Lisp

Lisp's Unusual Syntax

- The group never gave up the idea of compiling from some Fortran-like syntax
- But they never got around to it either
- In later years, people often tried to compile Lisp from a Fortran- or Algol-like syntax
- None of them caught on
- There are advantages to having programs and data use the same syntax, as we saw with Prolog

Lisp Influence

- The second-oldest general-purpose programming language still in use
- Some ideas, like the conditional expression and recursion, were adopted by Algol and later by many other imperative languages
- The function-oriented approach influenced modern functional languages like ML
- Garbage collection is increasingly common in many different language families

Lisp (1959)

- LISt Processing language (Designed at MIT by McCarthy)
- *AI research needed a language that:*
  - Process data in lists (rather than arrays)
  - Handles symbolic computation (rather than numeric)
- One universal, recursive data type: the s-expression
  - An s-expression is either an atom or a list of zero or more s-expressions
- Syntax is based on the lambda calculus
- *Pioneered functional programming*
  - No need for variables or assignment
  - Control via recursion and conditional expressions
- Status
  - Still the dominant language for AI
  - COMMON LISP and Scheme are contemporary dialects
  - ML, Miranda, and Haskell are related languages

Algol

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 of the language:*
    1. Close to mathematical notation
    2. Good for describing algorithms
    3. Must be translatable to machine code

The Good News

- Virtually all languages after 1958 used ideas pioneered by the Algol designs:
    - Compound statements: **`begin`** *statements* **`end`**
    - Free-format lexical structure
    - BNF definition of syntax
    - Local variables with block scope
    - Static typing with explicit type declarations
    - Nested if-then-else
    - Call by value (and call by name)
    - Recursive subroutines and conditional expressions (ex Lisp)
    - Dynamic arrays
    - First-class procedures
    - User-defined operators

Structured Programming

- Using phrase-level control instead of labels was called *structured programming*
- There was a long debate: many programmers found it difficult at first to do without labels
- Now, the revolution is over:
  - Some languages (like Java) eliminated **`go to`**
  - Others (like C++) still have it
  - But programmers rarely use it, even when permitted
- The revolution was triggered (or at least fueled) by the Algol designs

APL

- A Programming Language
- Designed by Ken Iverson at Harvard in late 1950’s
- A language for programming mathematical computations
  - especially those using matrices
- Functional style and many whole array operations
- Drawback is requirement of special keyboard

The 1960s: An Explosion in Programming Languages

- The development of hundreds of programming languages
- PL/I designed in 1963-4
- Algol 68
- SNOBOL
- Simula
- BASIC

The 1980s: Consolidation and New Paradigms

- Ada
  - US Department of Defence
  - European team lead by Jean Ichbiah. (Sam Lomonaco was also on the ADA team :-)
- Functional programming
  - Scheme, ML, Haskell
- Logic programming
  - Prolog
- Object-oriented programming
  - Smalltalk, C++, Eiffel

- SETL
    - Set operations
    - Compiler chooses data structure
- SASL
    - Normal-order functional language

Logic Programming: Prolog

- Developed at the University of Aix Marseille, by Comerauer and Roussel, with some help from Kowalski at the University of Edinburgh
- 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

Functional Programming

- **Common Lisp:** consolidation of LISP dialects spurred practical use, as did the development of Lisp Machines.
- **Scheme:** a simple and pure LISP like language used for teaching programming.
- **Logo:** Used for teaching young children how to program.
- **ML:** (MetaLanguage) a strongly-typed functional language first developed by Robin Milner in the 70's
- **Haskell:** polymorphicly typed, lazy, purely functional language.

ML

- Robin Milner, Edinburgh, 1974
- LCF: a tool for developing machine-assisted construction of formal logical proofs
- ML was designed as the implementation language for LCF
- Strong typing, parametric polymorphism, and type inference were in the first designs
- Remained closely tied to LCF development for several years

Issue: Formal Semantics

- The definition of Standard ML includes a formal semantics (a natural semantics)
- This was part of the initial design, not (as is more common) added after implementation
- Fits with the intended application: to trust the proofs produced by LCF, you must trust the language in which LCF is implemented

Simula (1962-67)

- Designed and built by Ole-Johan Dahl and Kristen Nygaard at the Norwegian Computing Centre (NCC) in Oslo between 1962 and 1967
- Originally designed and implemented as a language for discrete event simulation
- Based on ALGOL 60

Primary Contributions:

- Coroutines - a kind of subprogram
- Classes (data plus methods) and objects
- Inheritance
- Dynamic binding

=> Introduced the basic ideas that developed into object-oriented programming.

The Good Old Days

In the early years of programming languages, the most frequent phrase we heard was that the only way to program a computer was in octal. Of course a few years later a few people admitted that maybe you could use assembly language…. I have here a copy of the manual for Mark I. I think most of you would be totally flabbergasted if you were faced with programming a computer, using a Mark I manual. All it gives you are the codes. From there on you're on your own to write a program. We were not programmers in those days. The word had not yet come over from England. We were "coders."

Rear Admiral Dr. Grace Murray Hopper