Programming Languages: A History

Cover of the January 1961 issue of the CACM
Although computational expression (in theory) began before the advent of (modern, electronic) computers, after the first practical general-purpose computers were build (ca 1940–1950), it became a very useful goal to express computation to these very real and very complicated machines which could handle only very simple instructions.
This machines could do no more (or no less) than the ideal machines, but they could do much faster.
Programming Languages: Early–Recent

- FORTRAN
  Diversions: standards, compiler terminology
- ALGOL—most influential
- LISP, COBOL, APL, SNOBOL, BASIC
- PL/I, SIMULA, C, Pascal
- Smalltalk, Modula-2, C++, Ada, Modula-3, Oberon, Java

Read Sebesta, Chapter 2; Webber, Chapter 24.
Remarks

ALGOL: the shape of the “family” tree suggests that ALGOL is influential. Even FORTRAN is now greatly influenced by ALGOL. APL: so different from ALGOL I have put it to off to the side. It is an imperative language with a set of polymorphic, matrix combinators. These ideas are developed in a much cleaner setting in functional languages.
SETL: the premise is that sets, the fundamental structure of mathematics, should be the fundamental data structure of programming. This premise is dubious and classical mathematics ignores the problems.
FORTRAN

- FORmula TRANslating
- Work began 1953 led by John Backus at IBM, finished in 1957
- Implemented for the IBM 704 which influenced the control constructs
- FORTRAN II including separately compiled subroutines released in 1958
- FORTRAN IV (developed 1960–1962) widely used
- FORTRAN 66 — ANSI X3.9-1966
- FORTRAN 77 — ANSI X3.9-1978
- Fortran 90 — ANSI X3.198.1992
- Fortran 95 — ISO/IEC 1539:1997
A Digression


- International Organization for Standardization. www.iso.ch

What is the significance of standards published by standards organizations?

John Warner Backus (1924–2007)

- Born 1924, joined IBM in 1946 and retired in 1991
- Helped design FORTRAN
- The metalanguage BNF for ALGOL
- Turing Award 1977, “Can programming be liberated from the von Neumann style? A functional style and its algebra of programs”
- IEEE Computer Society Pioneer Award in 1980
John Warner Backus (1924–2007)
FORTRAN, the first high-level programming language
The Preliminary Report, 1954, claims that FORTRAN will virtually eliminate coding and debugging. Designing a language was not yet recognized as important.

As far as we were aware, we simply made up the language as we went along. We did not regard language design as a difficult problem, merely a simple prelude to the real problem: designing a compiler which could produce efficient programs.

John Backus
A page from the first FORTRAN manual, the IBM manual for the 704.
DIMENSION A(11)
READ A
2   DO 3,8,11  J=1,11
3      I=11-J
       Y=SQRT(ABS(A(I+1)))+5*A(I+1)**3
       IF (400. >= Y) 8,4
4      PRINT I,999.
       GO TO 2
8      PRINT I,Y
11     STOP

From Knuth and Prado, 1991
The TPK ALGORITHM, FORTRAN STYLE

FUNF(T) = SQRTF(ABSF(T)) + 5.0*T**3

DIMENSION A(11)

1 FORMAT(6F12.4)
READ 1, A
DO 10 J=1,11
   I=11-J
   Y=FUNF(A(I+1))
   IF (400.0-Y) 4,8,8
4   PRINT 5,I
5   FORMAT(I10, 10H TOO LARGE)
   GO TO 10
8   PRINT 9, I, Y
9   FORMAT(I10, F12.7)
10  CONTINUE
STOP 52525
FORTRAN

TPK in FORTRAN 95
FORTRAN 200X will have object-oriented features.

I don’t know what the language of the year 2000 will look like but I know it will be called FORTRAN.


—“the infantile disorder”—is hopelessly inadequate for whatever computer application you have in mind today ... too clumsy, too risky and too expensive.

Edsger W. Dijkstra. EWD 498 “How do we tell truths that might hurt?” The manuscript was published as pages 129–131 of Selected Writings on Computing: A Personal Perspective, Springer-Verlag, 1982. ISBN 0 387 90652 5.
With the first high-level language, FORTRAN, begins an entirely new technology for the implementation of high-level languages.
This is the end of the long diversion concerning compilation. With Fortran begins the history of translating high-level languages to executable form, indeed the very idea of a human-oriented programming language.
Second Generation

- ALGOL (1958); ALGOL 60
- LISP (1959), McCarthy, lists, AI
- COBOL (1960), business data processing
- APL (1962), Iverson, matrices
- SNOBOL (1964), Griswold, strings
- BASIC (1965), Kemeny & Kurtz
ALGOL (for ALGOrithmic Language) was designed by committee. In the late 1950s committees of the GAMM (Gesellschaft für angewandte Mathematik und Mechanik) and the ACM (Association for Computing Machinery) were considering designing a universal programming language for communicating programs among users and to computers. In 1958 four members of each organization met at the Eidgenössische Technische Hochschule in Zürich. They hammered out the first draft of the new language. It was initially called IAL (the International Algebraic Language) by the Americans, but soon came to be known as ALGOL. In 1960 members of both organizations met again, this time in Paris. Proposals to include blocks, call-by-value and call-by-name, and recursion made it into the language.
TPK:

begin integer \( i \); real array \( a[0 : 10] \);

real procedure \( f(t) \); real \( t \); value \( t \);

\[
f := \sqrt{\text{abs}(t)} + 5 \times t \uparrow 3;
\]

for \( i := 0 \) step 1 until 10 do read(\( a[i] \))

for \( i := 10 \) step \(-1\) until 0 do

begin \( y := f(a[i]) \);

if \( y > 400 \) then write(\( i, \text{"TOO LARGE"} \))

else write(\( i, y \));

end

end.
ALGOL-like

- variables are changed
- block and procedures are the basic units
- procedures may call themselves recursively
- data are organized into different types
- identifiers have lexical scope
Cambridge-Polish notation for lists, functions:

\[(element_1 \ element_2 \ \cdots \ element_n)\]

\[(operator \ argument_1 \ \cdots \ argument_n)\]

Some examples:

\[(PLUS \ 3 \ (MINUS \ 6 \ 2)) \ ;; \ 3 + (6-2)\]

\[(QUOTE \ (PLUS \ X \ Y)) \ ;; \ "X+Y"\]

\[(SETQ \ A \ 3) \ ;; \ A := 3\]

\[(LAMBDA \ (X) \ (EXPT \ ((ADD1 \ X) \ 3)))\]
LISP

- Simple syntax
- Data and program have same form
- Recursive data structures (list), recursive programs, garbage collection
- Used in symbolic computing, artificial intelligent applications
- Originally interpreted and slow
- Later dialects Scheme, T, Common Lisp
- More about functional languages later
The United States Department of Defense sponsored a meeting in May 1959 to foster a common programming language for data-processing applications. A result of the meeting was the formation of the CODASYL (Committee on Data Systems Languages) Executive Committee to oversee the work and two task groups to examine the problem in the short-term and the intermediate-term future. In six months the short-term committee had fashioned a new language called COBOL (COmmon Business Oriented Language) seen as a “stopgap” solution to curtail the proliferation of languages by individual computer manufacturers (IBM, Honeywell, RCA, and Sylvania, among others). Although not on the committee, Grace Murray Hopper influenced the language through her previous work in designing the first data-processing compiler: Flow-matic. Wirth (in 2001) states that records came from COBOL.
Then there was COBOL’s creed that if your programming language was similar enough to English, even officers would be able to program. A major force was IBM, which wanted to sell hardware and hence was eager to present its machines as solutions rather than as the source of new problems.

EWD 1298, 2000
Kenneth E. Iverson began developing APL before 1960. At first the language was not intended for implementation. It was designed to describe computer architecture and was influenced by the field of linear algebra. Currently in use in financial institutions.
APL Example

An example to determine if a number \( N \) is prime

<table>
<thead>
<tr>
<th>APL expression</th>
<th>value for ( N = 11 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N - 2 )</td>
<td>( 9 )</td>
</tr>
<tr>
<td>( 1 + i )</td>
<td>( 12 )</td>
</tr>
<tr>
<td>( 2 + i )</td>
<td>( 14 )</td>
</tr>
<tr>
<td>( (1 + i)</td>
<td>( 11 )</td>
</tr>
<tr>
<td>( 1 )</td>
<td>( 1 )</td>
</tr>
</tbody>
</table>

So, \( N = 11 \) is prime (11 divided by 7 leaves a remainder of 4, etc.)
APL Example

An example to determine if a number $N$ is prime

<table>
<thead>
<tr>
<th>APL expression</th>
<th>value for $N = 11$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N - 2$</td>
<td></td>
</tr>
</tbody>
</table>

So, $N = 11$ is prime (11 divided by 7 leaves a remainder of 4, etc.)
### APL Example

An example to determine if a number $N$ is prime

<table>
<thead>
<tr>
<th>APL expression</th>
<th>value for $N = 11$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N - 2$</td>
<td>9</td>
</tr>
</tbody>
</table>

So, $N = 11$ is prime (11 divided by 7 leaves a remainder of 4, etc.)
### APL Example

An example to determine if a number $N$ is prime

<table>
<thead>
<tr>
<th>APL expression</th>
<th>value for $N = 11$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N - 2$</td>
<td>9</td>
</tr>
<tr>
<td>$i N - 2$</td>
<td></td>
</tr>
</tbody>
</table>
APL Example

An example to determine if a number $N$ is prime

<table>
<thead>
<tr>
<th>APL expression</th>
<th>value for $N = 11$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N - 2$</td>
<td>9</td>
</tr>
<tr>
<td>$iN - 2$</td>
<td>1 2 3 4 5 6 7 8 9</td>
</tr>
</tbody>
</table>
APL Example

An example to determine if a number $N$ is prime

<table>
<thead>
<tr>
<th>APL expression</th>
<th>value for $N = 11$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N - 2$</td>
<td>9</td>
</tr>
<tr>
<td>$iN - 2$</td>
<td>1 2 3 4 5 6 7 8 9</td>
</tr>
<tr>
<td>$1 + iN - 2$</td>
<td></td>
</tr>
</tbody>
</table>

So, $N = 11$ is prime (11 divided by 7 leaves a remainder of 4, etc.)
### APL Example

An example to determine if a number $N$ is prime

<table>
<thead>
<tr>
<th>APL expression</th>
<th>value for $N = 11$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N - 2$</td>
<td>9</td>
</tr>
<tr>
<td>$iN - 2$</td>
<td>1 2 3 4 5 6 7 8 9</td>
</tr>
<tr>
<td>$1 + iN - 2$</td>
<td>2 3 4 5 6 7 8 9 10</td>
</tr>
</tbody>
</table>

So, $N = 11$ is prime (11 divided by 7 leaves a remainder of 4, etc.)
An example to determine if a number $N$ is prime

<table>
<thead>
<tr>
<th>APL expression</th>
<th>value for $N = 11$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N - 2$</td>
<td>9</td>
</tr>
<tr>
<td>$iN - 2$</td>
<td>1 2 3 4 5 6 7 8 9</td>
</tr>
<tr>
<td>$1 + iN - 2$</td>
<td>2 3 4 5 6 7 8 9 10</td>
</tr>
<tr>
<td>$(1 + iN - 2) \mid N$</td>
<td></td>
</tr>
</tbody>
</table>
**APL Example**

An example to determine if a number $N$ is prime

<table>
<thead>
<tr>
<th>APL expression</th>
<th>value for $N = 11$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N - 2$</td>
<td>9</td>
</tr>
<tr>
<td>$iN - 2$</td>
<td>1 2 3 4 5 6 7 8 9</td>
</tr>
<tr>
<td>$1 + iN - 2$</td>
<td>2 3 4 5 6 7 8 9 10</td>
</tr>
<tr>
<td>$(1 + iN - 2) \mid N$</td>
<td>1 2 3 1 5 4 3 2 1</td>
</tr>
</tbody>
</table>
### APL Example

An example to determine if a number $N$ is prime

<table>
<thead>
<tr>
<th>APL expression</th>
<th>value for $N = 11$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N - 2$</td>
<td>9</td>
</tr>
<tr>
<td>$iN - 2$</td>
<td>1 2 3 4 5 6 7 8 9</td>
</tr>
<tr>
<td>$1 + iN - 2$</td>
<td>2 3 4 5 6 7 8 9 10</td>
</tr>
<tr>
<td>$(1 + iN - 2) \mid N$</td>
<td>1 2 3 1 5 4 3 2 1</td>
</tr>
<tr>
<td>$0 \neq (1 + iN - 2) \mid N$</td>
<td>1 2 3 1 5 4 3 2 1</td>
</tr>
</tbody>
</table>
An example to determine if a number $N$ is prime

<table>
<thead>
<tr>
<th>APL expression</th>
<th>value for $N = 11$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N - 2$</td>
<td>9</td>
</tr>
<tr>
<td>$iN - 2$</td>
<td>1 2 3 4 5 6 7 8 9</td>
</tr>
<tr>
<td>$1 + iN - 2$</td>
<td>2 3 4 5 6 7 8 9 10</td>
</tr>
<tr>
<td>$(1 + iN - 2) \mid N$</td>
<td>1 2 3 1 5 4 3 2 1</td>
</tr>
<tr>
<td>$0 \neq (1 + iN - 2) \mid N$</td>
<td>1 1 1 1 1 1 1 1 1</td>
</tr>
</tbody>
</table>
APL Example

An example to determine if a number $N$ is prime

<table>
<thead>
<tr>
<th>APL expression</th>
<th>value for $N = 11$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N - 2$</td>
<td>9</td>
</tr>
<tr>
<td>$iN - 2$</td>
<td>$1 2 3 4 5 6 7 8 9$</td>
</tr>
<tr>
<td>$1 + iN - 2$</td>
<td>$2 3 4 5 6 7 8 9 10$</td>
</tr>
<tr>
<td>$(1 + iN - 2) \mid N$</td>
<td>$1 2 3 1 5 4 3 2 1$</td>
</tr>
<tr>
<td>$0 \neq (1 + iN - 2) \mid N$</td>
<td>$1 1 1 1 1 1 1 1 1$</td>
</tr>
<tr>
<td>$\land/0 \neq (1 + iN - 2) \mid N$</td>
<td>$1$</td>
</tr>
</tbody>
</table>
**APL Example**

An example to determine if a number \( N \) is prime

<table>
<thead>
<tr>
<th>APL expression</th>
<th>value for ( N = 11 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N - 2 )</td>
<td>9</td>
</tr>
<tr>
<td>( iN - 2 )</td>
<td>1 2 3 4 5 6 7 8 9</td>
</tr>
<tr>
<td>( 1 + iN - 2 )</td>
<td>2 3 4 5 6 7 8 9 10</td>
</tr>
<tr>
<td>( (1 + iN - 2) \mid N )</td>
<td>1 2 3 1 5 4 3 2 1</td>
</tr>
<tr>
<td>( 0 \neq (1 + iN - 2) \mid N )</td>
<td>1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>( \land/0 \neq (1 + iN - 2) \mid N )</td>
<td>1</td>
</tr>
</tbody>
</table>

So, \( N = 11 \) is prime (11 divided by 7 leaves a remainder of 4, etc.)
APL Example

An example to determine if a number $N$ is prime

<table>
<thead>
<tr>
<th>APL expression</th>
<th>value for $N = 11$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N - 2$</td>
<td>9</td>
</tr>
<tr>
<td>$iN - 2$</td>
<td>1 2 3 4 5 6 7 8 9</td>
</tr>
<tr>
<td>$1 + iN - 2$</td>
<td>2 3 4 5 6 7 8 9 10</td>
</tr>
<tr>
<td>$(1 + iN - 2) \mid N$</td>
<td>1 2 3 1 5 4 3 2 1</td>
</tr>
<tr>
<td>$0 \neq (1 + iN - 2) \mid N$</td>
<td>1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>$\land/0 \neq (1 + iN - 2) \mid N$</td>
<td>1</td>
</tr>
</tbody>
</table>

So, $N = 11$ is prime
(11 divided by 7 leaves a remainder of 4, etc.)
J is a high level general-purpose language, with a emphasis on functional programming and array processing. J was designed and developed by Ken Iverson and Roger Hui, and implemented by Iverson Software Inc (ISI). Iverson Software Inc. has changed its name to Jsoftware Inc. http://www.jsoftware.com/. Since March 2011, J is free and open source software under the GPLv3 license. Currently (2016) version J805 is available.
$ date=. 1,7,7,6
4

$ word=. 's','a','w'
$ |. word
was

$ qsort =:
]`((:$:@((().<:(.)#).)),(.,($:@((().?(.#)).))))@.(##)
The programming language SNOBOL (pronounced “snowball”) was developed by David J. Farber, Ralph Griswold, and Ivan P. Polonsky while at Bell Telephone Laboratories in the mid 1960s. The language was designed primarily to process string data. Griswold claims that it took them longer to find a name for the language than to implement it. At first SEXI (String EXpressions Interpreter) was proposed, and rejected. The commonly accepted interpretation of the acronym SNOBOL is StriNg Oriented symBOlic Language, but this was actually not proposed seriously. There are a number of spin-offs from the original SNOBOL language. Some of them have catchy names of their own: SPITBOL (speedy implementation), FASBOL, SLOBOL, and SNOBAT. The most familiar version is SNOBOL4.
There is an ALGOL-like successor to SNOBOL called Icon developed by Griswold at the University of Arizona.

procedure f (x)
    return (sqrt(abs(x)) + 5.0*x*x*x)
end

procedure main (args)
    A := table (0);
    every A[0 to 10] := read()
    every i := 10-(0 to 10) do
        write (i,",",400.0>f(A[i]) | "TOO LARGE", "\n")
    end
end
Now there is Unicon a unified extended dialect of Icon.

procedure f (x)
    return (sqrt(abs(x)) + 5.0*x*x*x)
end

procedure main (args)
    A := []
    every A[0 to 10] := read()
    every i := !reverse(A) do
        write (i,",",400.0>f(A[i]) | "TOO LARGE", \\n    end
BASIC (Beginner’s All-purpose Symbolic Instruction Code) was designed for students at Dartmouth College by John G. Kemeny and Thomas E. Kurtz. The goals were to make an interactive language that was easy to learn, quick to compile, and easy to debug. Time-shared BASIC provided remote access to GE 225 computer.

```
10 REM BASIC PROGRAM FOR TPK ALGORITHM
20 DIM A(11)
30 FOR I = 1 TO 11
40 INPUT A(I)
50 NEXT I
60 FOR J = 1 TO 11
70 LET I = 11 - J
80 LET Y = A(I+1) + 5 * A(I+1)
90 IF Y > 400.0 THEN 120
100 PRINT I+1, Y
110 GO TO 130
120 PRINT I, "TOO LARGE"
130 NEXT J
140 STOP
150 END
```
Third generation

After exploration of a large range of domain-specific languages, there came a period of consolidation.

- PL/I (1965) IBM
- SIMULA (1966), Dahl & Nygard
- BCPL (1967); C (1972) Dennis Richie
- Pascal (1971), Niklaus Wirth
PL/I

- IBM tried to replace FORTRAN and COBOL in commercial, systems, and scientific applications
- exception handling and multitasking
- no success; PL/I is complex
- ANSI standard 1976; “general-purpose subset” approved 1981
The development of SIMULA was carried out at the Norwegian Computing Center, a semi-governmental research institute supervised by the Royal Norwegian Council for Scientific and Industrial Research. SIMULA was closely related to ALGOL and, like ALGOL, has had significant influence on programming language development but is not widely used.

The impetus for the language was the need to simulate situations like queues at a supermarket, response times of emergency services, or chain reactions of nuclear reactors—thus the name SIMULA. Kristen Nygaard and Ole-Johan Dahl designed and implemented the language in the early 1960s.
Originally designed and implemented by Dennis M. Ritchie at Bell Laboratories in New Jersey in 1972 for rewriting the implementation of Unix operating system.

Influenced by the programming language B created by Ken Thompson and implemented on the PDP-7.

B, in turn, was influenced by BCPL a systems-programming language developed for compiler writing.

C an untyped, structured assembly language for the PDP-11.


TPK in C
In the decade of the 1970s there was consolidation and improvement. This decade saw the emergence of a number of influential languages: Pascal, Modula, and CLU.

The programming language ALGOL had demonstrated the enormous range of possibilities. But all the intricacies had to be melded into a whole that had many competing requirements. Discipline was required and this was provided by Niklaus Wirth, who first tried to influence the direction of ALGOL 68 with his proposal (ALGOL W) and then went his own way with the development of Pascal. The first draft of the language excluded dynamic arrays and recursive procedures, and was implemented by a single graduate student in FORTRAN for the CDC 6000 in 1969.

ISO 7185 standard for Pascal in 1990.
Recent

- Smalltalk (1976), Kay & Xerox
- Modula-2 (1977), Wirth
- C++ (1979), Stroustrup
- Ada (1979), US Department of Defense
- Oberon (1988), Wirth
- Modula-3 (1990), Olivetti & DEC
- Java (1995), Sun

(and functional languages—discussed later)

Era of modularity and object-orientation.
SmallTalk (1976)

SmallTalk emerged out of the plan to find the basic elements of computation and build an entire computer out of them. If the right elements were identified, the hope was, the whole result be elegant and simple enough the children could use it. Compare this quixotic research effort with the OLPC (One Laptop Per Child) initiative of today—equally quixotic but more practical. The OLPC initiative apparently uses Squeak (an open source implementation of the Smalltalk programming language and environment.

The forces to play down the difficulty of programming were just too strong. . . . Then there was the religion at Xerox PARC that it should all be so natural that toddlers would love to do it.

EWD 1298, 2000
Smalltalk appears to be responsible for the unconventional vocabulary used in object-oriented programming languages today (method versus function) and the unfortunate camelCase style of writing identifiers. The “object” has failed to be the simple, basic element of computation. Dijkstra’s point is that there will be no “magic bullet,” no magic notation or abstraction that will make it easy to programming correctly. Just as no English language spelling or grammar reform will make it easy to write like Ernest Hemmingway, John Steinbeck, or Joseph Conrad.
In the summer of ’71 I refined the KiddiKomp idea into a tighter design called miniCOM. It used a bit-slice approach, had a bit-map display, a pointing device, a choice of “secondary” storage, and a language I now called “Smalltalk”—as in “programming should be a matter of . . .” and “children should program in . . .”. The name was also a reaction against the “IndoEuropean god theory” where systems were named Zeus, Odin, and Thor, and hardly did anything. I figured that “Smalltalk” was so innocuous a label that if it ever did anything nice people would be pleasantly surprised.

Alan Kay, 1993
Sometime in the late 1970s Niklaus Wirth begins work on a successor to Pascal for the Lilth workstation developed at ETH Zurich. The language, called Modula, was described “Modula–A Language of Modular Multiprogramming” by Niklaus Wirth in *Software Practice and Experience*, 1977. Its features include support from concurrent programming and modules. A revised language, called Modula-2, was implemented in 1980. Influenced by Mesa, the most novel feature was the module.
C++

C with classes 1979–1983 preprocessor
Evolution of C++ 1982–1985 first commercial release
Release 2.0 1985-1988 early commercial use
Growth in use 1987– large-scale use
Release 3.0 1989
ISO/IEC Standard 1998

The TPK program in C++
Development of Ada

- 1973-1974 cost study determined that the DOD was spending $3 billion annually on software, over half on embedded computer systems.

- The HOLWG evaluated 23 existing languages against the Tinman requirements:
  
  FORTRAN, COBOL, PL/I, HAL/S, TACPOL, CMS-2, CS-4, SPL/I, JOVIAL J3, JOVIAL J73, ALGOL 60, ALGOL 68, CORAL 66, Pascal, SUMULA 67, LIS, LTR, TRL/2, EUCLID, PDL2, PEARL, MORAL, EL/I

concluding in January 1977 that none were suitable, though Pascal, ALGOL 68 or PL/I would be a good starting point.
Development of Ada

- Request for proposals issued April 1977; 17 proposals received. Four contractors picked to produce prototype languages:
  - Cii Honeywell Bull led by J. Ichbiah (green)
  - Intermetrics led by B. Bros gol (red)
  - SofTech led by John Goodenough (blue)
  - SRI led by Jay Spitzen (yellow)

- Green chosen the winner in May 1979. The design team was led by Jean D. Ichbiah (1940-2007) and developed by Cii Honeywell Bull and later Alsys, and by Honeywell Systems and Research Center, under contract to the DoD.
Development of Ada

▶ The language, known only as DoD-1 up to that point, was given the name Ada in May 1979.


▶ Intermetrics, Inc. was the prime contractor for the “mapping/revision team” for the new Ada 9X standard. S. Tucker Taft served as Technical Director.


Ada Features

- Concurrency
- Modules
- Universal polymorphism
- Strongly typed – interesting type system
- Typed objects (Ada 95)
- OO-like syntax (Ada 2005)
Modula-3 (1990)

Not really related to Modula-2.
Java

- June 1991—C++ rejected, Gosling begins work on “Oak”
- June 1993—Time-Warner chooses SGI over Sun for its interactive cable TV trial in Orlando, FL
- 1994—Van Hoff rewrote the compiler in Oak itself
- September 1994—Payne and Naughton demo “WebRunner” (later called HotJava)
- May 1995—Sun announces Java and HotJava at SunWorld’95
- September 2004 — J2SE v1.5
Comparison of Languages

Knuth’s TPK program in different programming languages:

- tpk.f95 – FORTRAN 95
- tpk.c – C
- tpk.cc – C++
- TPK.java – Java
- tpk.cs – C#
- tpk.adb – Ada
- TPK.m3 – Modula-3
- tpk.py – Python
- tpk.hs – Haskell
What is a programming language?

Inventor or Killer Quiz

http://www.malevole.com/mv/misc/killerquiz/