Before 1940

The first programming languages predate the modern computer. At first, the languages were codes. During a nine-month period in 1842-1843, Ada Lovelace translated Italian mathematician Luigi Menabrea's memoir on Charles Babbage's newest proposed machine, the Analytical Engine. With the article, she appended a set of notes which specified in complete detail a method for calculating Bernoulli numbers with the Engine, recognized by some historians as the world's first computer program. But some biographers debate the extent of her original contributions versus those of her husband.\[citation needed\]

The Jacquard loom, invented in 1801, used holes in punched cards to represent sewing loom arm movements in order to generate decorative patterns automatically.

Herman Hollerith realized that he could encode information on punch cards when he observed that train conductors would encode the appearance of the ticket holders on the train tickets using the position of punched holes on the tickets. Hollerith then proceeded to encode the 1890 census data on punch cards.

The first computer codes were specialized for the applications. In the first decades of the twentieth century, numerical calculations were based on decimal numbers. Eventually it was realized that logic could be represented with numbers, as well as with words. For example, Alonzo Church was able to express the lambda calculus in a formulaic way. The Turing machine was an abstraction of the operation of a tape-marking machine, for example, in use at the telephone companies. However, unlike the lambda calculus, Turing's code does not serve well as a basis for higher-level
languages — its principal use is in rigorous analyses of algorithmic complexity.

Like many "firsts" in history, the first modern programming language is hard to identify. From the start, the restrictions of the hardware defined the language. Punch cards allowed 80 columns, but some of the columns had to be used for a sorting number on each card. Fortran included some keywords which were the same as English words, such as "IF", "GOTO" (go to) and "CONTINUE". The use of a magnetic drum for memory meant that computer programs also had to be interleaved with the rotations of the drum. Thus the programs were more hardware dependent than today.

To some people the answer depends on how much power and human-readability is required before the status of "programming language" is granted. Jacquard looms and Charles Babbage's Difference Engine both had simple, extremely limited languages for describing the actions that these machines should perform. One can even regard the punch holes on a player piano scroll as a limited domain-specific language, albeit not designed for human consumption.

**The 1940s**

In the 1940s the first recognizably modern, electrically powered computers were created. The limited speed and memory capacity forced programmers to write hand tuned assembly language programs. It was soon discovered that programming in assembly language required a great deal of intellectual effort and was error-prone.

In 1948, Konrad Zuse [1] published a paper about his programming language Plankalkül. However, it was not implemented in his time and his original contributions were isolated from other developments.

Some important languages that were developed in this period include:

- 1943 - Plankalkül (Konrad Zuse)
- 1943 - ENIAC coding system
- 1949 - C-10

**The 1950s and 1960s**

In the 1950s the first three modern programming languages whose descendants are still in widespread use today were designed:

- **FORTRAN** (1955), the "FORmula TRANslator", invented by John Backus et al.;
- **LISP**, the "LISt Processor", invented by John McCarthy et al.;
- **COBOL**, the COmmon Business Oriented Language, created by the Short Range Committee, heavily influenced by Grace Hopper.

Another milestone in the late 1950s was the publication, by a committee of American and European computer scientists, of "a new language for algorithms"; the ALGOL 60 Report (the "ALGOrithmic Language"). This report consolidated many ideas circulating at the time and featured two key language innovations:

- arbitrarily nested block structure: meaningful chunks of code could be grouped into statement blocks without having to be turned into separate, explicitly named procedures;
- lexical scoping: a block could have its own variables that code outside the chunk cannot access, let alone manipulate.

Another innovation, related to this, was in how the language was described:

- a mathematically exact notation, Backus–Naur Form (BNF), was used to describe the language's syntax. Nearly all subsequent programming languages have used a variant of
BNF to describe the context-free portion of their syntax.

Algol 60 was particularly influential in the design of later languages, some of which soon became more popular. The Burroughs large systems were designed to be programmed in an extended subset of Algol.

Algol's key ideas were continued, producing ALGOL 68:

- syntax and semantics became even more orthogonal, with anonymous routines, a recursive typing system with higher-order functions, etc.;
- not only the context-free part, but the full language syntax and semantics were defined formally, in terms of Van Wijngaarden grammar, a formalism designed specifically for this purpose.

Algol 68's many little-used language features (e.g. concurrent and parallel blocks) and its complex system of syntactic shortcuts and automatic type coercions made it unpopular with implementers and gained it a reputation of being difficult. Niklaus Wirth actually walked out of the design committee to create the simpler Pascal language.

Overview:

- 1951 - Regional Assembly Language
- 1952 - Autocode
- 1954 - FORTRAN
- 1955 - FLOW-MATIC (forerunner to COBOL)
- 1957 - COMTRAN (forerunner to COBOL)
- 1958 - LISP
- 1958 - ALGOL 58
- 1959 - FACT (forerunner to COBOL)
- 1959 - COBOL
- 1962 - APL
- 1962 - Simula
- 1964 - BASIC
- 1964 - PL/I

1967-1978: establishing fundamental paradigms

The period from the late 1960s to the late 1970s brought a major flowering of programming languages. Most of the major language paradigms now in use were invented in this period:

- **Simula**, invented in the late 1960s by Nygaard and Dahl as a superset of Algol 60, was the first language designed to support object-oriented programming.
- **C**, an early systems programming language, was developed by Dennis Ritchie and Ken Thompson at Bell Labs between 1969 and 1973.
- **Smalltalk** (mid 1970s) provided a complete ground-up design of an object-oriented language.
- **Prolog**, designed in 1972 by Colmerauer, Roussel, and Kowalski, was the first logic programming language.
- **ML** built a polymorphic type system (invented by Robin Milner in 1973) on top of Lisp, pioneering statically typed functional programming languages.

Each of these languages spawned an entire family of descendants, and most modern languages count at least one of them in their ancestry.

The 1960s and 1970s also saw considerable debate over the merits of "structured programming", which essentially meant programming without the use of Goto. This debate was closely related to language design: some languages did not include GOTO, which forced structured programming on
the programmer. Although the debate raged hotly at the time, nearly all programmers now agree that, even in languages that provide GOTO, it is bad programming style to use it except in rare circumstances. As a result, later generations of language designers have found the structured programming debate tedious and even bewildering.

Some important languages that were developed in this period include:

- 1970 - Pascal
- 1970 - Forth
- 1972 - C
- 1972 - Smalltalk
- 1972 - Prolog
- 1973 - ML
- 1978 - SQL (initially only a query language, later extended with programming constructs)

The 1980s: consolidation, modules, performance

The 1980s were years of relative consolidation. C++ combined object-oriented and systems programming. The United States government standardized Ada, a systems programming language intended for use by defense contractors. In Japan and elsewhere, vast sums were spent investigating so-called fifth-generation programming languages that incorporated logic programming constructs. The functional languages community moved to standardize ML and Lisp. Rather than inventing new paradigms, all of these movements elaborated upon the ideas invented in the previous decade.

However, one important new trend in language design was an increased focus on programming for large-scale systems through the use of modules, or large-scale organizational units of code. Modula, Ada, and ML all developed notable module systems in the 1980s. Module systems were often wedded to generic programming constructs—generics being, in essence, parameterized modules (see also polymorphism in object-oriented programming).

Although major new paradigms for programming languages did not appear, many researchers expanded on the ideas of prior languages and adapted them to new contexts. For example, the languages of the Argus and Emerald systems adapted object-oriented programming to distributed systems.

The 1980s also brought advances in programming language implementation. The RISC movement in computer architecture postulated that hardware should be designed for compilers rather than for human assembly programmers. Aided by processor speed improvements that enabled increasingly aggressive compilation techniques, the RISC movement sparked greater interest in compilation technology for high-level languages.

Language technology continued along these lines well into the 1990s.

Some important languages that were developed in this period include:

- 1983 - Ada
- 1983 - C++
- 1985 - Eiffel
- 1987 - Perl
- 1989 - FL (Backus)

The 1990s: the Internet age

The 1990s saw no fundamental novelty, but much recombination as well as maturation of old ideas. A big driving philosophy was programmer productivity. Many "rapid application
development" (RAD) languages emerged, which usually came with an IDE, garbage collection, and were descendants of older languages. All such languages were object-oriented. These included Object Pascal, Visual Basic, and C#. Java was a more conservative language that also featured garbage collection and received much attention. More radical and innovative than the RAD languages were the new scripting languages. These did not directly descend from other languages and featured new syntaxes and more liberal incorporation of features. Many consider these scripting languages to be more productive than even the RAD languages, but often because of choices that make small programs simpler but large programs more difficult to write and maintain. [citation needed] Nevertheless, scripting languages came to be the most prominent ones used in connection with the Web.

Some important languages that were developed in this period include:

- 1990 - Haskell
- 1991 - Python
- 1991 - Java
- 1993 - Ruby
- 1993 - Lua
- 1994 - ANSI Common Lisp
- 1995 - JavaScript
- 1995 - PHP
- 2000 - C#
- 2008 - JavaFX Script

**Current trends**

Programming language evolution continues, in both industry and research. Some of the current trends include:

- Mechanisms for adding security and reliability verification to the language: extended static checking, information flow control, static thread safety.
- Alternative mechanisms for modularity: mixins, delegates, aspects.
- Component-oriented software development.
- Metaprogramming, reflection or access to the abstract syntax tree
- Increased emphasis on distribution and mobility.
- Integration with databases, including XML and relational databases.
- Support for Unicode so that source code (program text) is not restricted to those characters contained in the ASCII character set; allowing, for example, use of non-Latin-based scripts or extended punctuation.
- XML for graphical interface (XUL, XAML).

**Prominent people in the history of program languages**

- John Backus, inventor of Fortran.
- John McCarthy, inventor of LISP.
- Alan Cooper, developer of Visual Basic.
- Edsger W. Dijkstra, developed the framework for proper programming.
- James Gosling, developer of Oak, the precursor of Java.
- Anders Heijlsberg, developer of Turbo Pascal and C#.
- Grace Hopper, developer of Flow-Matic, influencing COBOL.
- Kenneth E. Iverson, developer of APL.
- Bill Joy, inventor of vi, early author of BSD Unix, and originator of SunOS, which became Solaris.
• Alan Kay, pioneering work on object-oriented programming, and originator of Smalltalk.
• Brian Kernighan, co-author of the first book on the C programming language with Dennis Ritchie, coauthor of the AWK and AMPL programming languages.
• John von Neumann, originator of the operating system concept.
• Dennis Ritchie, inventor of C.
• Bjarne Stroustrup, developer of C++.
• Ken Thompson, inventor of Unix.
• Niklaus Wirth, inventor of Pascal and Modula.

See also

• ACM History of Programming Languages Conference
• History of computing hardware
• Programming language
• Timeline of computing
• Timeline of programming languages
• List of programming languages

References

McCullough, Jacob A. - Wrote this article.

Further reading

• Sammet, Jean E., "Programming Languages: History and Fundamentals" Prentice-Hall, 1969

External links

• The History of Programming Languages by Diarmuid Pigott
• History and evolution of programming languages.
• The Evolution of Programming Languages by Peter Grogono. PDF.
• Graph of programming language history