**HIGH LEVEL LANGUAGES**

**HISTORY OF HIGH LEVEL LANGUAGE**

Even though a huge number of computer languages exist, computer languages are a fairly new field, since the first high-level languages were written in the 1950s, around the time computers were invented. The earliest computers were programmed in binary so the set of instructions was just a series of 0 and 1. Before computer programming languages were made, paper tapes and punch cards which held complicated weaving patterns for the loom Tabulating Machine Company Looms by Jacquard in 1710. A century later, Charles Babbage started building a computing and the analytical Machine. In the 20th century, Herman Hollerith founded the Tabulating Machine. His machine tabulators were used to speed up the counting and sorting of punch cards. In the early 1940s J. Presper Eckert and John W. Mauchly started building the ENIAC (Electronic Numerical Integrator and Calculator), which was completed by 1946.

The very first high-level programming language was FORTRAN, which stands for FORmula TRANslation. It was developed in 1956 (first manual appeared in 1956, but first developed in 1954) by John Backus, a worker at IBM. FORTRAN's goal was to ease the pain of writing in assembly language. When FORTRAN was first introduced, it was looked on suspiciously since almost all the programmers then only worked with the machine code and assembly languages. The programmers at that time had an initial belief that programs compiled from high-level language would be less efficient than those written at low-level. In order to persuade potential users of the benefits of working with a high-level language, Backus designed an excellent compiler for FORTRAN so the programs were just as efficient as those written in low-level languages. This was the best compiler for many years.

FORTRAN soon became popular because it provided a realistic and desirable alternative to low-level language programming for mathematical and scientific applications. This programming language was also the first to be widely used. However, one of the limitations of FORTRAN was that it was specifically oriented toward the [IBM 704](https://en.wikipedia.org/wiki/IBM_704) machine. This is a major set-back since the language syntax contains many idiosyncrasies from the IBM 704 machine. From a pure programming language design standpoint this is a "cardinal sin" but historically, it is understandable. FORTRAN is known for its efficiency. Over the years, FORTRAN had been upgraded and developed into FORTRAN -II, FORTRAN -IV, FORTRAN -66, and FORTRAN -77. Early versions of FORTRAN largely restricted users. On the other hand, the later versions allowed more flexibility, but since it has adapted, it is not as easy to use as it should be. The development of the fast string processing makes FORTRAN more of a general-purpose language than it used to be, but it is still described as a mathematical or scientific language.

Early history[[edit](https://en.wikipedia.org/w/index.php?title=History_of_programming_languages&action=edit&section=2)]

During a nine-month period in 1842–1843, [Ada Lovelace](https://en.wikipedia.org/wiki/Ada_Lovelace) translated the memoir of Italian mathematician [Luigi Menabrea](https://en.wikipedia.org/wiki/Luigi_Menabrea) about [Charles Babbage](https://en.wikipedia.org/wiki/Charles_Babbage)'s newest proposed machine, the [analytical engine](https://en.wikipedia.org/wiki/Analytical_engine). With the article she appended a set of notes which specified in complete detail a method for calculating [Bernoulli numbers](https://en.wikipedia.org/wiki/Bernoulli_number) with the engine, recognized by some historians as the world's first computer program.[[1]](https://en.wikipedia.org/wiki/History_of_programming_languages#cite_note-1)

[Herman Hollerith](https://en.wikipedia.org/wiki/Herman_Hollerith) realized that he could encode information on [punch cards](https://en.wikipedia.org/wiki/Punched_card) when he observed that [train](https://en.wikipedia.org/wiki/Train) [conductors](https://en.wikipedia.org/wiki/Conductor_(transportation)) encode the appearance of the ticket holders on the train tickets using the position of punched holes on the tickets[[2]](https://en.wikipedia.org/wiki/History_of_programming_languages#cite_note-2). Hollerith then encoded the 1890 American census data on punch cards.

The first computer codes were specialized for their applications. In the first decades of the 20th century, numerical calculations were based on decimal numbers. Eventually it was realized that logic could be represented with numbers, not only with words. For example, [Alonzo Church](https://en.wikipedia.org/wiki/Alonzo_Church) was able to express the [lambda calculus](https://en.wikipedia.org/wiki/Lambda_calculus) in a formulatic way. The [Turing machine](https://en.wikipedia.org/wiki/Turing_machine) was an abstraction of the operation of a tape-marking machine, for example, in use at the telephone companies. Turing machines set the basis for storage of programs as data in the [von Neumann architecture](https://en.wikipedia.org/wiki/Von_Neumann_architecture#Development_of_the_stored-program_concept) of computers by representing a machine through a finite number. 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](https://en.wikipedia.org/wiki/Computational_complexity_theory).

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](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/Drum_memory) for memory meant that computer programs also had to be interleaved with the rotations of the drum. Thus the programs were more hardware-dependent.

To some people, what was the first modern programming language 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](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/Player_piano) scroll as a limited [domain-specific language](https://en.wikipedia.org/wiki/Domain-specific_language), albeit not designed for human consumption.

First programming languages[[edit](https://en.wikipedia.org/w/index.php?title=History_of_programming_languages&action=edit&section=3)]

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](https://en.wikipedia.org/wiki/Assembly_language) programs. It was eventually realized that programming in assembly language required a great deal of intellectual effort.

The first programming languages designed to communicate instructions to a computer were written in the 1950s. An early [high-level programming language](https://en.wikipedia.org/wiki/High-level_programming_language) to be designed for a computer was [Plankalkül](https://en.wikipedia.org/wiki/Plankalk%C3%BCl" \o "Plankalkül), developed by the Germans for [Z1](https://en.wikipedia.org/wiki/Z1_(computer)) by [Konrad Zuse](https://en.wikipedia.org/wiki/Konrad_Zuse) between 1943 and 1945. However, it was not implemented until 1998 and 2000.[[3]](https://en.wikipedia.org/wiki/History_of_programming_languages#cite_note-3)

[John Mauchly](https://en.wikipedia.org/wiki/John_Mauchly)'s [Short Code](https://en.wikipedia.org/wiki/Short_Code_(computer_language)), proposed in 1949, was one of the first high-level languages ever developed for an [electronic computer](https://en.wikipedia.org/wiki/Electronic_computer).[[4]](https://en.wikipedia.org/wiki/History_of_programming_languages#cite_note-Sebesta-4) Unlike [machine code](https://en.wikipedia.org/wiki/Machine_code), Short Code statements represented mathematical expressions in understandable form. However, the program had to be translated into [machine code](https://en.wikipedia.org/wiki/Machine_code) every time it ran, making the process much slower than running the equivalent machine code.

At the [University of Manchester](https://en.wikipedia.org/wiki/University_of_Manchester), [Alick Glennie](https://en.wikipedia.org/wiki/Alick_Glennie" \o "Alick Glennie) developed [Autocode](https://en.wikipedia.org/wiki/Autocode" \o "Autocode) in the early 1950s. A [programming language](https://en.wikipedia.org/wiki/High-level_programming_language), it used a [compiler](https://en.wikipedia.org/wiki/Compiler) to automatically convert the language into machine code. The first code and compiler was developed in 1952 for the [Mark 1](https://en.wikipedia.org/wiki/Manchester_Mark_1) computer at the University of Manchester and is considered to be the first [compiled](https://en.wikipedia.org/wiki/Compiler) high-level programming language.[[5]](https://en.wikipedia.org/wiki/History_of_programming_languages#cite_note-5)[[6]](https://en.wikipedia.org/wiki/History_of_programming_languages#cite_note-6)

The second autocode was developed for the Mark 1 by [R. A. Brooker](https://en.wikipedia.org/wiki/Tony_Brooker) in 1954 and was called the "Mark 1 Autocode". Brooker also developed an autocode for the [Ferranti Mercury](https://en.wikipedia.org/wiki/Ferranti_Mercury) in the 1950s in conjunction with the University of Manchester. The version for the [EDSAC](https://en.wikipedia.org/wiki/EDSAC) 2 was devised by [D. F. Hartley](https://en.wikipedia.org/w/index.php?title=D._F._Hartley&action=edit&redlink=1) of [University of Cambridge Mathematical Laboratory](https://en.wikipedia.org/wiki/University_of_Cambridge_Mathematical_Laboratory) in 1961. Known as EDSAC 2 Autocode, it was a straight development from Mercury Autocode adapted for local circumstances, and was noted for its object code optimisation and source-language diagnostics which were advanced for the time. A contemporary but separate thread of development, [Atlas Autocode](https://en.wikipedia.org/wiki/Atlas_Autocode) was developed for the University of Manchester [Atlas 1](https://en.wikipedia.org/wiki/Atlas_Computer_(Manchester))machine.

In 1954, language [FORTRAN](https://en.wikipedia.org/wiki/FORTRAN) was invented at IBM by [John Backus](https://en.wikipedia.org/wiki/John_Backus); it was the first widely used [high level general purpose programming language](https://en.wikipedia.org/wiki/High-level_language) to have a functional implementation, as opposed to just a design on paper.[[7]](https://en.wikipedia.org/wiki/History_of_programming_languages#cite_note-7)[[8]](https://en.wikipedia.org/wiki/History_of_programming_languages#cite_note-8) It is still a popular language for [high-performance computing](https://en.wikipedia.org/wiki/High-performance_computing)[[9]](https://en.wikipedia.org/wiki/History_of_programming_languages#cite_note-hpc-9) and is used for programs that benchmark and rank the world's [fastest supercomputers](https://en.wikipedia.org/wiki/TOP500).[[10]](https://en.wikipedia.org/wiki/History_of_programming_languages#cite_note-10)

Another early programming language was devised by [Grace Hopper](https://en.wikipedia.org/wiki/Grace_Hopper) in the US, called [FLOW-MATIC](https://en.wikipedia.org/wiki/FLOW-MATIC). It was developed for the [UNIVAC I](https://en.wikipedia.org/wiki/UNIVAC_I) at [Remington Rand](https://en.wikipedia.org/wiki/Remington_Rand) during the period from 1955 until 1959. Hopper found that business data processing customers were uncomfortable with mathematical notation, and in early 1955, she and her team wrote a specification for an [English](https://en.wikipedia.org/wiki/English_language) programming language and implemented a prototype.[[11]](https://en.wikipedia.org/wiki/History_of_programming_languages#cite_note-11) The FLOW-MATIC compiler became publicly available in early 1958 and was substantially complete in 1959.[[12]](https://en.wikipedia.org/wiki/History_of_programming_languages#cite_note-12) Flow-Matic was a major influence in the design of [COBOL](https://en.wikipedia.org/wiki/COBOL), since only it and its direct descendent [AIMACO](https://en.wikipedia.org/wiki/AIMACO) were in actual use at the time.[[13]](https://en.wikipedia.org/wiki/History_of_programming_languages#cite_note-13)

Other languages still in use today include [LISP](https://en.wikipedia.org/wiki/Lisp_(programming_language)) (1958), invented by [John McCarthy](https://en.wikipedia.org/wiki/John_McCarthy_(computer_scientist)) and [COBOL](https://en.wikipedia.org/wiki/COBOL) (1959), created by the Short Range Committee. 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*](https://en.wikipedia.org/wiki/ALGOL)*60 Report* (the "**ALGO**rithmic **L**anguage"). This report consolidated many ideas circulating at the time and featured three key language innovations:

* nested block structure: code sequences and associated declarations could be grouped into [blocks](https://en.wikipedia.org/wiki/Block_(programming)) without having to be turned into separate, explicitly named procedures;
* [lexical scoping](https://en.wikipedia.org/wiki/Scope_(programming)): a block could have its own private variables, procedures and functions, invisible to code outside that block, that is, [information hiding](https://en.wikipedia.org/wiki/Information_hiding).

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

* a mathematically exact notation, [Backus–Naur form](https://en.wikipedia.org/wiki/Backus%E2%80%93Naur_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](https://en.wikipedia.org/wiki/Context-free_grammar) 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](https://en.wikipedia.org/wiki/Burroughs_large_systems) were designed to be programmed in an extended subset of Algol.

Algol's key ideas were continued, producing [ALGOL 68](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/Van_Wijngaarden_grammar), a formalism designed specifically for this purpose.

Algol 68's many little-used language features (for example, 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](https://en.wikipedia.org/wiki/Niklaus_Wirth) actually walked out of the design committee to create the simpler [Pascal](https://en.wikipedia.org/wiki/Pascal_(programming_language)) language

## The History and Influence of Programming Languages

***1957* - Fortran (short for “The IBM Mathematical Formula Translating System”)** General-purpose, high-level. For numeric and scientific computing (as an alternative to assembly language). Oldest programming language still used today.

***1958* - Lisp (short for “List Processor”)** High-level. For mathematical notation. Several new computer science topics: tree data structures, automatic storage management, dynamic typing, and self-hosting compilers

***1959* - Cobol (short for "Common Business-Oriented Language)** High-level. Primarily for business computing. First programming language to be mandated by the US Department of Defense.

***1964* - BASIC (acronym for “Beginner’s All-purpose Symbolic Instruction Code”)** General-purpose, high-level. Designed for simplicity. Popularity exploded in the mid-‘70s with home computers; early computer games were often written in Basic, including Mike Mayfield’s Star Trek.

***1970* - Pascal (after French mathematician/physicist Blaise Pascal)** High-level. For teaching structured programming and data structuring. Commercial versions widely used throughout the ‘80s.

***1972* - C (based on an earlier language called "B")** General-purpose, low-level. Created for Unix systems. Currently the world’s most popular programming language.2 Many leading languages are derivatives, including C#, Java, JavaScript, Perl, PHP, and Python.

***1980* - Ada (After Ada Lovelace, inventor of the first programming language)** High-level. Derived from Pascal. Contracted by the US Department of Defense in 1977 for developing large software systems.

***1983* - C++ (formerly “C with Classes”; ++ is the increment operator in “C”)** Intermediate-level, object-oriented. An extension of C, with enhancements such as classes, virtual functions, and templates.

***1983* - Objective-C (object-oriented extension of “C”)** General-purpose, high-level. Expanded on C, adding message-passing functionality based on Smalltalk language.

***1987* - Perl (a language named "PEARL" already existed, so "Pearl" wasn't an option...)** General-purpose, high-level. Created for report processing on Unix systems. Today it’s known for high power and versatility.

***1991* - Python (for British comedy troupe Monty Python – tutorials, sample code, and instructions often reference them)** General-purpose, high-level. Created to support a variety of programming styles and be fun to use.

***1993* - Ruby (the birthstone of one of the creator's collaborator)** General-purpose, high-level. A teaching language influence by Perl, Ada, Lisp, Smalltalk, etc. Designed for productive and enjoyable programming.

***1995* - Java (for the amount of coffee consumed while developing the language)** General-purpose, high-level. Made for an interactive TV project. Cross-platform functionality. Second most popular language (behind C).2

***1995* - PHP ("Personal Home Page")** Open-source, general-purpose. For building dynamic web pages. Most widely used open-source software by enterprises.

***1995* - JavaScript (final choice after "Mocha" and "LiveScript"** High-level. Created to extend web page functionality. Dynamic web pages use for form submission/validation, interactivity, animations, user activity tracking, etc

In [computer science](https://en.wikipedia.org/wiki/Computer_science), a **high-level programming language** is a [programming language](https://en.wikipedia.org/wiki/Programming_language) with strong [abstraction](https://en.wikipedia.org/wiki/Abstraction_(computer_science)) from the details of the [computer](https://en.wikipedia.org/wiki/Computer). In comparison to [low-level programming languages](https://en.wikipedia.org/wiki/Low-level_programming_language), it may use [natural language](https://en.wikipedia.org/wiki/Natural_language) elements, be easier to use, or may automate (or even hide entirely) significant areas of computing systems (e.g. [memory management](https://en.wikipedia.org/wiki/Memory_management)), making the process of developing a program simpler and more understandable relative to a lower-level language. The amount of abstraction provided defines how "high-level" a programming language is.[[1]](https://en.wikipedia.org/wiki/High-level_programming_language#cite_note-1)

In the 1960s, low-level programming languages using a [compiler](https://en.wikipedia.org/wiki/Compiler) were commonly called **[autocodes](https://en.wikipedia.org/wiki/Autocode" \o "Autocode)**.[[2]](https://en.wikipedia.org/wiki/High-level_programming_language#cite_note-kleith-2) Examples of autocodes are [COBOL](https://en.wikipedia.org/wiki/COBOL) and [Fortran](https://en.wikipedia.org/wiki/Fortran).[[3]](https://en.wikipedia.org/wiki/High-level_programming_language#cite_note-kleith2-3)

The first high-level programming language designed for computers was [Plankalkül](https://en.wikipedia.org/wiki/Plankalk%C3%BCl" \o "Plankalkül), created by [Konrad Zuse](https://en.wikipedia.org/wiki/Konrad_Zuse).[[4]](https://en.wikipedia.org/wiki/High-level_programming_language#cite_note-4) However, it was not implemented in his time, and his original contributions were (due to [World War II](https://en.wikipedia.org/wiki/World_War_II)) largely isolated from other developments, although it influenced [Heinz Rutishauser](https://en.wikipedia.org/wiki/Heinz_Rutishauser)'s language "Superplan" (and to some degree also Algol). The first really widespread high-level language was [Fortran](https://en.wikipedia.org/wiki/Fortran), a machine independent development of IBM's earlier [Autocode](https://en.wikipedia.org/wiki/Autocode" \o "Autocode) systems. [Algol](https://en.wikipedia.org/wiki/ALGOL), defined in 1958 and 1960, by committees of European and American computer scientists, introduced [recursion](https://en.wikipedia.org/wiki/Recursion) as well as [nested functions](https://en.wikipedia.org/wiki/Nested_functions) under [lexical scope](https://en.wikipedia.org/wiki/Lexical_scope). It was also the first language with a clear distinction between [value](https://en.wikipedia.org/wiki/Call_by_value) and [name-parameters](https://en.wikipedia.org/wiki/Call_by_name) and their corresponding [semantics](https://en.wikipedia.org/wiki/Semantics).[[5]](https://en.wikipedia.org/wiki/High-level_programming_language#cite_note-5) Algol also introduced several [structured programming](https://en.wikipedia.org/wiki/Structured_programming) concepts, such as the **while-do** and **if-then-else** constructs and its [syntax](https://en.wikipedia.org/wiki/Syntax) was the first to be described by a formal method, [Backus–Naur form](https://en.wikipedia.org/wiki/Backus%E2%80%93Naur_form) (BNF). During roughly the same period [Cobol](https://en.wikipedia.org/wiki/Cobol) introduced [records](https://en.wikipedia.org/wiki/Record_(computer_science)) (also called structs) and [Lisp](https://en.wikipedia.org/wiki/Lisp_(programming_language)" \o "Lisp (programming language))introduced a fully general [lambda abstraction](https://en.wikipedia.org/wiki/Lambda_abstraction) in a programming language for the first time.

A ***h***igh-***l***evel ***l***anguage (**HLL**) is a [programming language](https://www.webopedia.com/TERM/P/programming_language.html) such as [C](https://www.webopedia.com/TERM/C/C.html), [FORTRAN](https://www.webopedia.com/TERM/F/FORTRAN.html), or [Pascal](https://www.webopedia.com/TERM/P/Pascal.html) that enables a [programmer](https://www.webopedia.com/TERM/P/programmer.html) to write [programs](https://www.webopedia.com/TERM/P/program.html)that are more or less independent of a particular type of [computer](https://www.webopedia.com/TERM/C/computer.html). Such [languages](https://www.webopedia.com/TERM/L/language.html) are considered high-level because they are closer to human languages and further from [machine languages](https://www.webopedia.com/TERM/M/machine_language.html).

In contrast, [assembly languages](https://www.webopedia.com/TERM/A/assembly_language.html) are considered low-level because they are very close to machine languages.

