Early history

During 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); she supplemented the memoir with notes that specified in 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 published computer program.[[2]](https://en.wikipedia.org/wiki/History_of_programming_languages#cite_note-2)

The first computer codes were specialized for their applications: 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 formulaic way and the [Turing machine](https://en.wikipedia.org/wiki/Turing_machine) was an abstraction of the operation of a tape-marking machine.

First programming languages

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.

An early proposal for a [high-level programming language](https://en.wikipedia.org/wiki/High-level_programming_language) was [Plankalkül](https://en.wikipedia.org/wiki/Plankalk%C3%BCl), developed by [Konrad Zuse](https://en.wikipedia.org/wiki/Konrad_Zuse) for his [Z1 computer](https://en.wikipedia.org/wiki/Z1_(computer)) between 1943 and 1945 but not implemented at the time.

The first functioning programming languages designed to communicate instructions to a computer were written in the early 1950s. [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.

In the early 1950s, [Alick Glennie](https://en.wikipedia.org/wiki/Alick_Glennie) developed [Autocode](https://en.wikipedia.org/wiki/Autocode), possibly the first compiled programming language, at the [University of Manchester](https://en.wikipedia.org/wiki/University_of_Manchester). In 1954, a second iteration of the language, known as the "Mark 1 Autocode," was developed for the Mark 1 by [Tony Brooker](https://en.wikipedia.org/wiki/Tony_Brooker). 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 [Douglas Hartree](https://en.wikipedia.org/wiki/Douglas_Hartree) 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 optimization and source-language diagnostics. 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 a team led 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.It is still a popular language for [high-performance computing](https://en.wikipedia.org/wiki/High-performance_computing)[[7]](https://en.wikipedia.org/wiki/History_of_programming_languages#cite_note-hpc-7) and is used for programs that benchmark and rank the world's [fastest supercomputers](https://en.wikipedia.org/wiki/TOP500).

Another early programming language was devised by [Grace Hopper](https://en.wikipedia.org/wiki/Grace_Hopper) in the USA, 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.The FLOW-MATIC compiler became publicly available in early 1958 and was complete in 1959. 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.

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* . 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.

Algol's key ideas were continued, producing [ALGOL 68](https://en.wikipedia.org/wiki/ALGOL_68).

Some notable languages that were developed in this period include:

|  |  |
| --- | --- |
| * 1951 – [Regional Assembly Language](https://en.wikipedia.org/wiki/Assembly_language) * 1952 – [Autocode](https://en.wikipedia.org/wiki/Autocode) * 1954 – [IPL](https://en.wikipedia.org/wiki/Information_Processing_Language) (forerunner to LISP) * 1955 – [FLOW-MATIC](https://en.wikipedia.org/wiki/FLOW-MATIC) (led to COBOL) * 1957 – [FORTRAN](https://en.wikipedia.org/wiki/Fortran) (first compiler) * 1957 – [COMTRAN](https://en.wikipedia.org/wiki/COMTRAN) (precursor to COBOL) * 1958 – [LISP](https://en.wikipedia.org/wiki/Lisp_(programming_language)) * 1958 – [ALGOL 58](https://en.wikipedia.org/wiki/ALGOL_58) * 1959 – [FACT](https://en.wikipedia.org/wiki/FACT_computer_language) (forerunner to COBOL) * 1959 – [COBOL](https://en.wikipedia.org/wiki/COBOL) | * 1959 – [RPG](https://en.wikipedia.org/wiki/IBM_RPG) * 1962 – [APL](https://en.wikipedia.org/wiki/APL_(programming_language)) * 1962 – [Simula](https://en.wikipedia.org/wiki/Simula) * 1962 – [SNOBOL](https://en.wikipedia.org/wiki/SNOBOL) * 1963 – [CPL](https://en.wikipedia.org/wiki/Combined_Programming_Language) (forerunner to C) * 1964 – [Speakeasy](https://en.wikipedia.org/wiki/Speakeasy_(computational_environment)) * 1964 – [BASIC](https://en.wikipedia.org/wiki/BASIC) * 1964 – [PL/I](https://en.wikipedia.org/wiki/PL/I) * 1966 – [JOSS](https://en.wikipedia.org/wiki/JOSS) * 1966 - [MUMPS](https://en.wikipedia.org/wiki/MUMPS) * 1967 – [BCPL](https://en.wikipedia.org/wiki/BCPL) (forerunner to C) |

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:

* [**Speakeasy**](https://en.wikipedia.org/wiki/Speakeasy_(computational_environment)), developed in 1964 at [Argonne National Laboratory](https://en.wikipedia.org/wiki/Argonne_National_Laboratory) (ANL) by [Stanley Cohen](https://en.wikipedia.org/wiki/Stanley_Cohen_(physicist)), is an [object-oriented programming](https://en.wikipedia.org/wiki/Object-oriented_programming) system, much like the later [MATLAB](https://en.wikipedia.org/wiki/MATLAB), [IDL](https://en.wikipedia.org/wiki/IDL_(programming_language)) and [Mathematica](https://en.wikipedia.org/wiki/Mathematica) numerical package. Speakeasy has a clear [Fortran](https://en.wikipedia.org/wiki/Fortran) foundation syntax. Speakeasy and Modeleasy are still in use currently.
* [**Simula**](https://en.wikipedia.org/wiki/Simula), invented in the late 1960s by [Nygaard](https://en.wikipedia.org/wiki/Kristen_Nygaard) and [Dahl](https://en.wikipedia.org/wiki/Ole-Johan_Dahl) as a superset of Algol 60, was the first language designed to support [object-oriented programming](https://en.wikipedia.org/wiki/Object-oriented_programming).
* [**C**](https://en.wikipedia.org/wiki/C_(programming_language)), an early [systems programming](https://en.wikipedia.org/wiki/System_programming) language, was developed by [Dennis Ritchie](https://en.wikipedia.org/wiki/Dennis_Ritchie) and [Ken Thompson](https://en.wikipedia.org/wiki/Ken_Thompson) at [Bell Labs](https://en.wikipedia.org/wiki/Bell_Labs) between 1969 and 1973.
* [**Smalltalk**](https://en.wikipedia.org/wiki/Smalltalk) (mid-1970s) provided a complete ground-up design of an object-oriented language.
* [**Prolog**](https://en.wikipedia.org/wiki/Prolog), designed in 1972 by [Colmerauer](https://en.wikipedia.org/wiki/Alain_Colmerauer), [Roussel](https://en.wikipedia.org/w/index.php?title=Phillipe_Roussel&action=edit&redlink=1), and [Kowalski](https://en.wikipedia.org/wiki/Robert_Kowalski), was the first [logic programming](https://en.wikipedia.org/wiki/Logic_programming) language.
* [**ML**](https://en.wikipedia.org/wiki/ML_(programming_language)) built a polymorphic type system (invented by [Robin Milner](https://en.wikipedia.org/wiki/Robin_Milner) in 1973) on top of Lisp, pioneering [statically typed](https://en.wikipedia.org/wiki/Type_system) [functional programming](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/Structured_programming)", which essentially meant programming without the use of "[goto](https://en.wikipedia.org/wiki/Goto)". A significant fraction of programmers believed that, even in languages that provide "goto", it is bad [programming style](https://en.wikipedia.org/wiki/Programming_style) to use it except in rare circumstances. This debate was closely related to language design: some languages did not include a "goto" at all, which forced structured programming to the programmers.

To provide even faster compile times, some languages were structured for "[one-pass compilers](https://en.wikipedia.org/wiki/One-pass_compiler)" which expect subordinate routines to be defined first, as with [Pascal](https://en.wikipedia.org/wiki/Pascal_(programming_language)), where the main routine, or driver function, is the final section of the program listing.

Some notable languages that were developed in this period include:

|  |  |
| --- | --- |
| * 1967 – [BCPL](https://en.wikipedia.org/wiki/BCPL) (forerunner to B) * 1968 – [Logo](https://en.wikipedia.org/wiki/Logo_(programming_language)) * 1969 – [B](https://en.wikipedia.org/wiki/B_(programming_language)) (forerunner to C) * 1970 – [Pascal](https://en.wikipedia.org/wiki/Pascal_(programming_language)) * 1970 – [Forth](https://en.wikipedia.org/wiki/Forth_(programming_language)) * 1972 – [C](https://en.wikipedia.org/wiki/C_(programming_language)) | * 1972 – [Smalltalk](https://en.wikipedia.org/wiki/Smalltalk) * 1972 – [Prolog](https://en.wikipedia.org/wiki/Prolog) * 1973 – [ML](https://en.wikipedia.org/wiki/ML_(programming_language)) * 1975 – [Scheme](https://en.wikipedia.org/wiki/Scheme_(programming_language)) * 1978 – [SQL](https://en.wikipedia.org/wiki/SQL) (a query language,  later extended) |

1980s: consolidation, modules, performance

The 1980s were years of relative consolidation in [imperative languages](https://en.wikipedia.org/wiki/Imperative_language). Rather than inventing new paradigms, all of these movements elaborated upon the ideas invented in the previous decade. [C++](https://en.wikipedia.org/wiki/C%2B%2B)combined object-oriented and systems programming. The United States government standardized [Ada](https://en.wikipedia.org/wiki/Ada_(programming_language)), 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](https://en.wikipedia.org/wiki/Fifth-generation_programming_language) that incorporated logic programming constructs. The functional languages community moved to standardize ML and Lisp. Research in [Miranda](https://en.wikipedia.org/wiki/Miranda_(programming_language)), a functional language with [lazy evaluation](https://en.wikipedia.org/wiki/Lazy_evaluation), began to take hold in this decade.

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](https://en.wikipedia.org/wiki/Modula), Ada, and ML all developed notable module systems in the 1980s.

Although major new paradigms for imperative 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](https://en.wikipedia.org/wiki/Argus_(computer_system)) and [Emerald](https://en.wikipedia.org/w/index.php?title=Emerald_(computer_system)&action=edit&redlink=1) systems adapted object-oriented programming to [distributed systems](https://en.wikipedia.org/wiki/Distributed_computing).

Some notable languages that were developed in this period include:

|  |  |
| --- | --- |
| * 1980 – [C++](https://en.wikipedia.org/wiki/C%2B%2B) (as [C with classes](https://en.wikipedia.org/wiki/C_with_classes), renamed in 1983) * 1983 – [Ada](https://en.wikipedia.org/wiki/Ada_(programming_language)) * 1984 – [Common Lisp](https://en.wikipedia.org/wiki/Common_Lisp) * 1984 – [MATLAB](https://en.wikipedia.org/wiki/MATLAB) * 1984 – dBase III, dBase III Plus (Clipper and [FoxPro](https://en.wikipedia.org/wiki/FoxPro) as [FoxBASE](https://en.wikipedia.org/wiki/FoxBASE), later developing into [Visual FoxPro](https://en.wikipedia.org/wiki/Visual_FoxPro)) * 1985 – [Eiffel](https://en.wikipedia.org/wiki/Eiffel_(programming_language)) * 1986 – [Objective-C](https://en.wikipedia.org/wiki/Objective-C) | * 1986 – [LabVIEW](https://en.wikipedia.org/wiki/LabVIEW) (Visual Programming Language) * 1986 – [Erlang](https://en.wikipedia.org/wiki/Erlang_(programming_language)) * 1987 – [Perl](https://en.wikipedia.org/wiki/Perl) * 1988 – [Tcl](https://en.wikipedia.org/wiki/Tcl) * 1988 – [Wolfram Language](https://en.wikipedia.org/wiki/Wolfram_Language) (as part of [Mathematica](https://en.wikipedia.org/wiki/Mathematica), only got a separate name in June 2013) * 1989 – [FL](https://en.wikipedia.org/wiki/FL_(programming_language)) (Backus) |

1990s: the Internet age

The rapid growth of the Internet in the mid-1990s was the next major historic event in programming languages. By opening up a radically new platform for computer systems, the Internet created an opportunity for new languages to be adopted. In particular, the JavaScript programming language rose to popularity because of its early integration with the Netscape Navigator web browser. Various other scripting languages achieved widespread use in developing customized applications for web servers such as PHP. The 1990s saw no fundamental novelty in [imperative languages](https://en.wikipedia.org/wiki/Imperative_language), but much recombination and maturation of old ideas. This era began the spread of [functional languages](https://en.wikipedia.org/wiki/Functional_language). A big driving philosophy was programmer productivity. Many "rapid application development" (RAD) languages emerged, which usually came with an [IDE](https://en.wikipedia.org/wiki/Integrated_development_environment), [garbage collection](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)), and were descendants of older languages. All such languages were [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming). These included [Object Pascal](https://en.wikipedia.org/wiki/Object_Pascal), [Visual Basic](https://en.wikipedia.org/wiki/Visual_Basic), and [Java](https://en.wikipedia.org/wiki/Java_(programming_language)).

More radical and innovative than the RAD languages were the new [scripting languages](https://en.wikipedia.org/wiki/Scripting_language). These did not directly descend from other languages and featured new syntaxes and more liberal incorporation of features. Scripting languages came to be the most prominent ones used in connection with the Web.

Some notable languages that were developed in this period include:

|  |  |
| --- | --- |
| * 1990 – [Haskell](https://en.wikipedia.org/wiki/Haskell_(programming_language)) * 1991 – [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) * 1991 – [Visual Basic](https://en.wikipedia.org/wiki/Visual_Basic) * 1993 – [Lua](https://en.wikipedia.org/wiki/Lua_(programming_language)) * 1993 – [R](https://en.wikipedia.org/wiki/R_(programming_language)) * 1994 – [CLOS](https://en.wikipedia.org/wiki/CLOS) (part of ANSI [Common Lisp](https://en.wikipedia.org/wiki/Common_Lisp)) | * 1995 – [Ruby](https://en.wikipedia.org/wiki/Ruby_(programming_language)) * 1995 – [Ada 95](https://en.wikipedia.org/wiki/Ada_95) * 1995 – [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) * 1995 – [Delphi (Object Pascal)](https://en.wikipedia.org/wiki/Embarcadero_Delphi) * 1995 – [JavaScript](https://en.wikipedia.org/wiki/JavaScript) * 1995 – [PHP](https://en.wikipedia.org/wiki/PHP) * 1997 – [Rebol](https://en.wikipedia.org/wiki/REBOL) |

Current trends

Programming language evolution continues, in both industry and research. Some of the recent trends have included:

* Increasing support for [functional programming](https://en.wikipedia.org/wiki/Functional_programming) in mainstream languages used commercially, including [pure functional programming](https://en.wikipedia.org/wiki/Purely_functional_programming) for making code easier to reason about and easier to parallelise (at both micro- and macro- levels)
* Constructs to support [concurrent](https://en.wikipedia.org/wiki/Concurrent_computing) and [distributed](https://en.wikipedia.org/wiki/Distributed_computing) programming.
* Mechanisms for adding security and reliability verification to the language: extended static checking, [dependent typing](https://en.wikipedia.org/wiki/Dependent_typing), information flow control, static [thread safety](https://en.wikipedia.org/wiki/Thread_safety).
* Alternative mechanisms for composability and modularity: [mixins](https://en.wikipedia.org/wiki/Mixin), [traits](https://en.wikipedia.org/wiki/Trait_(computer_programming)), [type classes](https://en.wikipedia.org/wiki/Typeclass), [delegates](https://en.wikipedia.org/wiki/Delegation_(programming)), [aspects](https://en.wikipedia.org/wiki/Aspect-oriented_programming).
* Component-oriented software development.
* [Metaprogramming](https://en.wikipedia.org/wiki/Metaprogramming), [reflection](https://en.wikipedia.org/wiki/Reflection_(computer_science)) or access to the [abstract syntax tree](https://en.wikipedia.org/wiki/Abstract_syntax_tree):
  + [Aspect Oriented Programming](https://en.wikipedia.org/wiki/Aspect_Oriented_Programming) allowing developers to insert code in another module or class at "join points"
  + [Domain specific languages](https://en.wikipedia.org/wiki/Domain_specific_language) and [code generation](https://en.wikipedia.org/wiki/Automatic_programming)
    - XML for graphical interface ([XUL](https://en.wikipedia.org/wiki/XUL), [XAML](https://en.wikipedia.org/wiki/Extensible_Application_Markup_Language))
* Increased interest in distribution and mobility.
* Integration with databases, including [XML](https://en.wikipedia.org/wiki/XML) and [relational databases](https://en.wikipedia.org/wiki/Relational_database).
* [Open source](https://en.wikipedia.org/wiki/Open-source_software) as a developmental philosophy for languages, including the GNU Compiler Collection and languages such as [Python](https://en.wikipedia.org/wiki/Python_(programming_language)), [Ruby](https://en.wikipedia.org/wiki/Ruby_(programming_language)), and [Scala](https://en.wikipedia.org/wiki/Scala_(programming_language)).
* Massively parallel languages for coding 2000 processor GPU graphics processing units and supercomputer arrays including [OpenCL](https://en.wikipedia.org/wiki/OpenCL)
* Early research into (as-yet-unimplementable) [quantum computing](https://en.wikipedia.org/wiki/Quantum_computing) programming languages
* More interest in [visual programming languages](https://en.wikipedia.org/wiki/Visual_programming_language) like [Scratch](https://en.wikipedia.org/wiki/Scratch_(programming_language)).

Some notable languages developed during this period include:

|  |  |
| --- | --- |
| * 2000 – [ActionScript](https://en.wikipedia.org/wiki/ActionScript) * 2001 – [C#](https://en.wikipedia.org/wiki/C_Sharp_(programming_language)) * 2001 – [D](https://en.wikipedia.org/wiki/D_(programming_language)) * 2002 – [Scratch](https://en.wikipedia.org/wiki/Scratch_(programming_language)) * 2003 – [Groovy](https://en.wikipedia.org/wiki/Groovy_(programming_language)) * 2003 – [Scala](https://en.wikipedia.org/wiki/Scala_(programming_language)) * 2005 – [F#](https://en.wikipedia.org/wiki/F_Sharp_(programming_language)) * 2006 – [PowerShell](https://en.wikipedia.org/wiki/Windows_PowerShell) * 2007 – [Clojure](https://en.wikipedia.org/wiki/Clojure) | * 2009 – [Go](https://en.wikipedia.org/wiki/Go_(programming_language)) * 2010 – [Rust](https://en.wikipedia.org/wiki/Rust_(programming_language)) * 2011 – [Dart](https://en.wikipedia.org/wiki/Dart_(programming_language)) * 2011 – [Kotlin](https://en.wikipedia.org/wiki/Kotlin_(programming_language)) * 2011 – [Elixir](https://en.wikipedia.org/wiki/Elixir_(programming_language)) * 2012 – [Julia](https://en.wikipedia.org/wiki/Julia_(programming_language)) * 2012 - [TypeScript](https://en.wikipedia.org/wiki/TypeScript) * 2014 – [Swift](https://en.wikipedia.org/wiki/Swift_(programming_language)) |