The following is the Wikipedia entry on C history. The discussion in the videos are imprecise- based to some extent on personal experience when certain features entered the language. K&R edition 1(1978) and 2(19880 are critical for actual practice.

History[[edit](https://en.wikipedia.org/w/index.php?title=C_(programming_language)&action=edit&section=4)]

**Early developments**[[edit](https://en.wikipedia.org/w/index.php?title=C_(programming_language)&action=edit&section=5)]

[](https://en.wikipedia.org/wiki/File:Ken_n_dennis.jpg)

[Ken Thompson](https://en.wikipedia.org/wiki/Ken_Thompson) (left) with [Dennis Ritchie](https://en.wikipedia.org/wiki/Dennis_Ritchie) (right, the inventor of the C programming language)

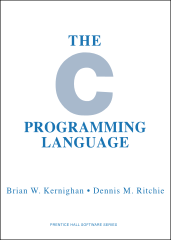
The origin of C is closely tied to the development of the [Unix](https://en.wikipedia.org/wiki/Unix) operating system, originally implemented in [assembly language](https://en.wikipedia.org/wiki/Assembly_language) on a [PDP-7](https://en.wikipedia.org/wiki/PDP-7) by Ritchie and Thompson, incorporating several ideas from colleagues. Eventually, they decided to port the operating system to a [PDP-11](https://en.wikipedia.org/wiki/PDP-11). The original PDP-11 version of Unix was developed in assembly language. The developers were considering rewriting the system using the [B language](https://en.wikipedia.org/wiki/B_(programming_language)), Thompson's simplified version of [BCPL](https://en.wikipedia.org/wiki/BCPL).[[9]](https://en.wikipedia.org/wiki/C_(programming_language)#cite_note-sigplan-9) However B's inability to take advantage of some of the PDP-11's features, notably [byte](https://en.wikipedia.org/wiki/Byte) addressability, led to C.

The development of C started in 1972 on the PDP-11 Unix system[[10]](https://en.wikipedia.org/wiki/C_(programming_language)#cite_note-unixport-10) and first appeared in [Version 2 Unix](https://en.wikipedia.org/wiki/Version_2_Unix).[[11]](https://en.wikipedia.org/wiki/C_(programming_language)#cite_note-11) The language was not initially designed with portability in mind, but soon ran on different platforms as well: a compiler for the [Honeywell 6000](https://en.wikipedia.org/wiki/Honeywell_6000) was written within the first year of C's history, while an [IBM System/370](https://en.wikipedia.org/wiki/IBM_System/370) port followed soon.[[1]](https://en.wikipedia.org/wiki/C_(programming_language)#cite_note-k.26r1e-1)[[10]](https://en.wikipedia.org/wiki/C_(programming_language)#cite_note-unixport-10) The name of C simply continued the alphabetic order started by B.[[12]](https://en.wikipedia.org/wiki/C_(programming_language)#cite_note-12)

Also in 1972, a large part of Unix was rewritten in C.[[13]](https://en.wikipedia.org/wiki/C_(programming_language)#cite_note-Stallings-13) By 1973, with the addition of struct types, the C language had become powerful enough that most of the [Unix](https://en.wikipedia.org/wiki/Unix)'s [kernel](https://en.wikipedia.org/wiki/Kernel_(operating_system)) was now in C.

Unix was one of the first operating system kernels implemented in a language other than [assembly](https://en.wikipedia.org/wiki/Assembly_language). (Earlier instances include the [Multics](https://en.wikipedia.org/wiki/Multics) system (written in [PL/I](https://en.wikipedia.org/wiki/PL/I)), and MCP ([Master Control Program](https://en.wikipedia.org/wiki/MCP_(Burroughs_Large_Systems))) for the [Burroughs B5000](https://en.wikipedia.org/wiki/Burroughs_large_systems)written in [ALGOL](https://en.wikipedia.org/wiki/ALGOL) in 1961.) Circa 1977, Ritchie and [Stephen C. Johnson](https://en.wikipedia.org/wiki/Stephen_C._Johnson) made further changes to the language to facilitate portability of the Unix operating system. Johnson's [Portable C Compiler](https://en.wikipedia.org/wiki/Portable_C_Compiler) served as the basis for several implementations of C on new platforms.[[10]](https://en.wikipedia.org/wiki/C_(programming_language)#cite_note-unixport-10)

**K&R C**[[edit](https://en.wikipedia.org/w/index.php?title=C_(programming_language)&action=edit&section=6)]

[](https://en.wikipedia.org/wiki/File:The_C_Programming_Language,_First_Edition_Cover_(2).svg)

The cover of the book, *The C Programming Language*

In 1978, [Brian Kernighan](https://en.wikipedia.org/wiki/Brian_Kernighan) and [Dennis Ritchie](https://en.wikipedia.org/wiki/Dennis_Ritchie) published the first edition of [*The C Programming Language*](https://en.wikipedia.org/wiki/The_C_Programming_Language).[[1]](https://en.wikipedia.org/wiki/C_(programming_language)#cite_note-k.26r1e-1) This book, known to C programmers as "K&R", served for many years as an informal [specification](https://en.wikipedia.org/wiki/Specification) of the language. The version of C that it describes is commonly referred to as *K&R C*. The second edition of the book[[14]](https://en.wikipedia.org/wiki/C_(programming_language)#cite_note-k.26r2e-14) covers the later [ANSI C](https://en.wikipedia.org/wiki/ANSI_C) standard, described below.

K&R introduced several language features:

* standard I/O library
* long int data type
* unsigned int data type
* compound assignment operators of the form =*op* (such as =-) were changed to the form *op*= (that is, -=) to remove the semantic ambiguity created by such constructs as i=-10, which had been interpreted as i =- 10(decrement i by 10) instead of the possibly intended i = -10 (let i be -10)

Even after the publication of the 1989 ANSI standard, for many years K&R C was still considered the "[lowest common denominator](https://en.wikipedia.org/wiki/Lowest_common_denominator_(computers))" to which C programmers restricted themselves when maximum portability was desired, since many older compilers were still in use, and because carefully written K&R C code can be legal Standard C as well.

In early versions of C, only functions that returned a non-int value needed to be declared if used before the function definition; a function used without any previous declaration was assumed to return type int, if its value was used.

For example:

long some\_function();

*/\* int \*/* other\_function();

*/\* int \*/* calling\_function()

{

long test1;

**register** */\* int \*/* test2;

test1 = some\_function();

**if** (test1 > 0)

test2 = 0;

**else**

test2 = other\_function();

**return** test2;

}

The int type specifiers which are commented out could be omitted in K&R C, but are required in later standards.

Since K&R function declarations did not include any information about function arguments, function parameter [type checks](https://en.wikipedia.org/wiki/Type_checking) were not performed, although some compilers would issue a warning message if a local function was called with the wrong number of arguments, or if multiple calls to an external function used different numbers or types of arguments. Separate tools such as Unix's [lint](https://en.wikipedia.org/wiki/Lint_programming_tool) utility were developed that (among other things) could check for consistency of function use across multiple source files.

In the years following the publication of K&R C, several features were added to the language, supported by compilers from AT&T (in particular [PCC](https://en.wikipedia.org/wiki/Portable_C_Compiler)[[15]](https://en.wikipedia.org/wiki/C_(programming_language)#cite_note-15)) and some other vendors. These included:

* [void](https://en.wikipedia.org/wiki/Void_type) functions (i.e., functions with no return value)
* functions returning [struct](https://en.wikipedia.org/wiki/Struct_(C_programming_language)" \o "Struct (C programming language)) or [union](https://en.wikipedia.org/wiki/Union_(computer_science)) types (rather than pointers)
* [assignment](https://en.wikipedia.org/wiki/Assignment_(computer_science)) for struct data types
* [enumerated types](https://en.wikipedia.org/wiki/Enumerated_type)

The large number of extensions and lack of agreement on a [standard library](https://en.wikipedia.org/wiki/C_standard_library), together with the language popularity and the fact that not even the Unix compilers precisely implemented the K&R specification, led to the necessity of standardization.

**ANSI C and ISO C**[[edit](https://en.wikipedia.org/w/index.php?title=C_(programming_language)&action=edit&section=7)]

*Main article:*[*ANSI C*](https://en.wikipedia.org/wiki/ANSI_C)

During the late 1970s and 1980s, versions of C were implemented for a wide variety of [mainframe computers](https://en.wikipedia.org/wiki/Mainframe_computer), [minicomputers](https://en.wikipedia.org/wiki/Minicomputer), and [microcomputers](https://en.wikipedia.org/wiki/Microcomputer), including the [IBM PC](https://en.wikipedia.org/wiki/IBM_PC), as its popularity began to increase significantly.

In 1983, the [American National Standards Institute](https://en.wikipedia.org/wiki/American_National_Standards_Institute) (ANSI) formed a committee, X3J11, to establish a standard specification of C. X3J11 based the C standard on the Unix implementation; however, the non-portable portion of the Unix C library was handed off to the[IEEE](https://en.wikipedia.org/wiki/Institute_of_Electrical_and_Electronics_Engineers) [working group](https://en.wikipedia.org/wiki/Working_group) 1003 to become the basis for the 1988 [POSIX](https://en.wikipedia.org/wiki/POSIX) standard. In 1989, the C standard was ratified as ANSI X3.159-1989 "Programming Language C". This version of the language is often referred to as [ANSI C](https://en.wikipedia.org/wiki/ANSI_C), Standard C, or sometimes C89.

In 1990, the ANSI C standard (with formatting changes) was adopted by the [International Organization for Standardization](https://en.wikipedia.org/wiki/International_Organization_for_Standardization) (ISO) as ISO/IEC 9899:1990, which is sometimes called C90. Therefore, the terms "C89" and "C90" refer to the same programming language.

ANSI, like other national standards bodies, no longer develops the C standard independently, but defers to the international C standard, maintained by the working group [ISO/IEC JTC1/SC22](https://en.wikipedia.org/wiki/ISO/IEC_JTC1/SC22)/WG14. National adoption of an update to the international standard typically occurs within a year of ISO publication.

One of the aims of the C standardization process was to produce a [superset](https://en.wikipedia.org/wiki/Superset) of K&R C, incorporating many of the subsequently introduced unofficial features. The standards committee also included several additional features such as [function prototypes](https://en.wikipedia.org/wiki/Function_prototype) (borrowed from C++), void pointers, support for international [character sets](https://en.wikipedia.org/wiki/Character_encoding) and [locales](https://en.wikipedia.org/wiki/Locale_(computer_software)), and preprocessor enhancements. Although the [syntax](https://en.wikipedia.org/wiki/C_syntax) for parameter declarations was augmented to include the style used in C++, the K&R interface continued to be permitted, for compatibility with existing source code.

C89 is supported by current C compilers, and most C code being written today is based on it. Any program written only in Standard C and without any hardware-dependent assumptions will run correctly on any [platform](https://en.wikipedia.org/wiki/Computing_platform) with a conforming C implementation, within its resource limits. Without such precautions, programs may compile only on a certain platform or with a particular compiler, due, for example, to the use of non-standard libraries, such as [GUI](https://en.wikipedia.org/wiki/Graphical_user_interface) libraries, or to a reliance on compiler- or platform-specific attributes such as the exact size of data types and byte [endianness](https://en.wikipedia.org/wiki/Endianness).

In cases where code must be compilable by either standard-conforming or K&R C-based compilers, the \_\_STDC\_\_ macro can be used to split the code into Standard and K&R sections to prevent the use on a K&R C-based compiler of features available only in Standard C.

After the ANSI/ISO standardization process, the C language specification remained relatively static for several years. In 1995 Normative Amendment 1 to the 1990 C standard (ISO/IEC 9899/AMD1:1995, known informally as C95) was published, to correct some details and to add more extensive support for international character sets.

**C99**

*Main article:*[*C99*](https://en.wikipedia.org/wiki/C99)

The C standard was further revised in the late 1990s, leading to the publication of ISO/IEC 9899:1999 in 1999, which is commonly referred to as "[C99](https://en.wikipedia.org/wiki/C99)". It has since been amended three times by Technical Corrigenda.[[16]](https://en.wikipedia.org/wiki/C_(programming_language)#cite_note-AutoTX-5-16)

C99 introduced several new features, including [inline functions](https://en.wikipedia.org/wiki/Inline_function), several new [data types](https://en.wikipedia.org/wiki/Data_type) (including long long int and a complex type to represent [complex numbers](https://en.wikipedia.org/wiki/Complex_number)), [variable-length arrays](https://en.wikipedia.org/wiki/Variable-length_array) and [flexible array members](https://en.wikipedia.org/wiki/Flexible_array_member), improved support for [IEEE 754](https://en.wikipedia.org/wiki/IEEE_754) floating point, support for [variadic macros](https://en.wikipedia.org/wiki/Variadic_macro" \o "Variadic macro) (macros of variable [arity](https://en.wikipedia.org/wiki/Arity)), and support for one-line comments beginning with //, as in [BCPL](https://en.wikipedia.org/wiki/BCPL) or C++. Many of these had already been implemented as extensions in several C compilers.

C99 is for the most part backward compatible with C90, but is stricter in some ways; in particular, a declaration that lacks a type specifier no longer has int implicitly assumed. A standard macro \_\_STDC\_VERSION\_\_ is defined with value 199901L to indicate that C99 support is available. [GCC](https://en.wikipedia.org/wiki/GNU_Compiler_Collection), [Solaris Studio](https://en.wikipedia.org/wiki/Sun_Studio_(software)), and other C compilers now support many or all of the new features of C99. The C compiler in [Microsoft Visual C++](https://en.wikipedia.org/wiki/Microsoft_Visual_C%2B%2B), however, implements the C89 standard and those parts of C99 that are required for compatibility with[C++11](https://en.wikipedia.org/wiki/C%2B%2B11).[[17]](https://en.wikipedia.org/wiki/C_(programming_language)#cite_note-17)

**C11**

*Main article:*[*C11 (C standard revision)*](https://en.wikipedia.org/wiki/C11_(C_standard_revision))

In 2007, work began on another revision of the C standard, informally called "C1X" until its official publication on 2011-12-08. The C standards committee adopted guidelines to limit the adoption of new features that had not been tested by existing implementations.

The C11 standard adds numerous new features to C and the library, including type generic macros, anonymous structures, improved Unicode support, atomic operations, multi-threading, and bounds-checked functions. It also makes some portions of the existing C99 library optional, and improves compatibility with C++. The standard macro \_\_STDC\_VERSION\_\_ is defined as 201112L to indicate that C11 support is available.