C (programming language)

In computing, **C** (/ˈsiː/, like the letter C) is a general-purpose programming language initially developed by Dennis Ritchie between 1969 and 1973 at Bell Labs.[4] Its design provides constructs that map efficiently to typical machine instructions, and therefore it found lasting use in applications that had formerly been coded in assembly language, most notably system software like the Unix computeroperating system.[5]

C is one of the most widely used programming languages of all time,[6][7] and there are very few computer architectures for which a Ccompiler does not exist.

Many later languages have borrowed directly or indirectly from C, including: C#, D, Go, Java, JavaScript, Limbo, LPC, Perl, PHP,Python, and Unix's C Shell. The most pervasive influence on these languages has been syntactical, and they tend to combine the recognizable expression and statement syntax of C with underlying type systems and data models that can be radically different. C++started as a preprocessor for C and is currently nearly a superset of C.[8]

Before there was an official standard for C, many users and implementors relied on an informal specification contained in a book byRitchie and Brian Kernighan; that version is generally referred to as "K&R" C. In 1989 the American National Standards Institutepublished a standard for C (generally called "ANSI C" or "C89"). The next year, the same specification was approved by theInternational Organization for Standardization as an international standard (generally called "C90"). ISO later released an extension to the internationalization support of the standard in 1995, and a revised standard (known as "C99") in 1999. The current version of the standard (now known as "C11") was approved in December of 2011.

Design

C is an imperative (procedural) language. It was designed to be compiled using a relatively straightforward compiler, to provide low-level access to memory, to provide language constructs that map efficiently to machine instructions, and to require minimal run-time support. C was therefore useful for many applications that had formerly been coded inassembly language, such as in system programming.

Despite its low-level capabilities, the language was designed to encourage cross-platform programming. A standards-compliant and portably written C program can be compiled for a very wide variety of computer platforms and operating systems with few changes to its source code. The language has become available on a very wide range of platforms, from embedded microcontrollers to supercomputers.

Uses

C is often used for "system programming", including implementing operating systems and embedded system applications, due to a combination of desirable characteristics such as code portability and efficiency, ability to access specific hardware addresses, ability to pun types to match externally imposed data access requirements, and low run-time demand on system resources. C can also be used for website programming using CGI as a "gateway" for information between the Web application, the server, and the browser.[22] Some reasons for choosing C over interpreted languages are its speed, stability, and near-universal availability.[23]

One consequence of C's wide availability and efficiency is that compilers, libraries, and interpreters of *other* programming languages are often implemented in C. The primary implementations of Python (CPython), Perl 5, and PHP are all written in C.

Due to its thin layer of abstraction and low overhead, C allows efficient implementations of algorithms and data structures, which is useful for programs that perform a lot of computations. For example, the GNU Multi-Precision Library, the GNU Scientific Library, Mathematica and MATLAB are completely or partially written in C.

C is sometimes used as an intermediate language by implementations of other languages. This approach may be used for portability or convenience; by using C as an intermediate language, it is not necessary to develop machine-specific code generators. C has some features, such as line-number preprocessor directives and optional superfluous commas at the end of initializer lists, which support compilation of generated code. However, some of C's shortcomings have prompted the development of other C-based languages specifically designed for use as intermediate languages, such as C--.

C has also been widely used to implement end-user applications, but much of that development has shifted to newer languages.