FOR021-1 24H - Engelsk
Obligatory Individual Project #1

A Short History of the C Programming Language

Casper Eide Özdemir-Børretzen

This expository essay was written with the main purpose of practicing and improving my skills in the English language, and the choice of topic was motivated by a personal interest in learning more about the C programming language.

Western Norway
University of
Applied Sciences

"C is quirky, flawed, and an enormous success. Although accidents of history surely helped, it evidently satisfied a need for a system implementation language efficient enough to displace assembly language, yet sufficiently abstract and fluent to describe algorithms and interactions in a wide variety of environments."

Dennis M. Ritchie

Introduction: The C programming language is a general-purpose compiled language oriented towards system programming, classified as a member of the traditional procedural family of languages. The initial version of C was created in the period 1969-1973 for use in development of the UNIX operating system. Although C is often referred to as a lower-level language these days, it was considered a high-level language initially, as it offered a great deal of abstractions and human readability compared to the prevailing assembly languages most commonly used for the purpose of writing operating systems at the time of C's creation. After public release in the early 1970s a gradual rise in popularity led to it becoming one of the dominant languages in the late-80s and 90s. Today most of the world's computers run UNIX or UNIX-like systems, and the majority of the world's software is written in C or in a language descendant from C.

At the heart of a computer is the central processing unit (CPU). The CPU contains one or more arithmetic logic units (ALU) that perform arithmetic or logical operations on binary data stored in the internal registers. The data in the registers can be transferred to and from memory (RAM) and subsequently be read by software on the computer, perhaps translating a stored numerical value into a letter of the alphabet or the color of a single pixel in an image. A modern domestic CPU can perform billions of operations each second, enabling such feats as performing complex mathematical calculations in milliseconds or running intricate simulations of fluids in real-time. However, to instruct the CPU to perform a single operation it simply needs to receive a number. This number is referred to as an opcode, or instruction code, and the number tied to a certain operation will usually vary between different CPU models. A single or multiple of these opcodes in binary form are called machine code, and the history of programming languages begins with machine code as the method for the programmer to communicate with the machine. However, the human brain is not wired to write and comprehend hundreds, thousands, or even millions of binary numbers in sequence, therefore the necessity for a layer of abstraction between programmer and machine arose. This resulted in what is called assembly language. Where earlier the programmer might would have written the binary number 01101001 as an instruction for the CPU to add two numbers together, it could now be written as the instruction ADD. This could then be translated directly into machine code by an assembler program, resulting in highly efficient code, but it was still fairly difficult for humans to read, and the code was not portable, meaning the same program had to be written differently for every computer architecture.

By the late-1960s higher level languages with further layers of abstractions existed, but these languages were slow and inefficient compared to assembly code. Two of the main driving forces behind the development of the C language, Dennis M. Ritchie and Kenneth Lane Thompson, were working in the Computing Science Research Center at Bell Telephone Laboratories at the time. They had been involved in the Multics operating system project, and although the project was cancelled, the experience and ideas had inspired a desire for Thompson to create a new operating system. His vision was of a comfortable computing environment written in a higher-level language with high portability, meaning it could be adapted to different computer architectures with relative ease. Thompson wrote the original UNIX (Uniplexed Information and Computing Service) in PDP-7 assembly language. This consisted of a kernel, an editor, an assembler and a shell. The system was then self-supporting, and work began contemporaneously to develop a high-level programming language for the embryonic operating system. In the period 1969-1970 Thompson created the B language, derived directly from BCPL (designed by Martin Richards in the mid-60s), he also wrote an accompanying compiler for B. A compiler is a program that translates code written in a high-level language, for instance B, to machine code. The requirement going forward would be for the language and compiler to be small and fast enough to rival assembly language. Dennis M. Ritchie assisted Thompson in the creation of the UNIX operating system, and in 1971 Ritchie began work on extending

B. In a transition phase this became known as “new B”, but as development of the language progressed it evolved into C, and by 1973 the essentials of the language had reached a state of completion. After years of collaborative effort and hard labour a strong language and a effective compiler had been written into existence, consequently the core of the UNIX operating system was rewritten in C during the summer of that year.

The development of C and UNIX continued at a steady but less intensive pace in the following years. By 1977 The language was still associated exclusively with UNIX, however, development now entered a second period of heightened productivity, and a shift in focus took place, orienting progress towards portability and type safety for C, in conjunction with work going into producing a reference manual for the language. In 1978 Brian Kernighan and Dennis Ritchie completed the language reference book *The C Programming Language*. The focus on portability proved a fruitful endeavour, with the outcome of C being successfully ported to numerous types of computers, and this development, combined with the success of UNIX and the availability of a clear and concise reference manual, led to an explosive growth in popularity for the incipient programming language. Usage of C spread widely throughout the 1980s, with compilers becoming available on nearly every machine and operating system. In the mid-80s the American National Standards Institute (ANSI) set out to produce a definition of the language, resulting in a report produced in 1989 that would subsequently be accepted by the International Organization for Standardization (ISO) as *ISO/IEC 9899-1990*.

Since the ANSI definition of 1989, the language has remained exceptionally stable, however some changes were made over the years, as defined by the succeeding ISO standards: “C99” (*ISO/IEC 9899:1999*) in 1999, “C11” (*ISO/IEC 9899:2011*) in 2011, “C17” (*ISO/IEC 9899:2018*) in 2018 and “C23” (*ISO/IEC 9899:2024*) in 2024. Furthermore, the language has spawned several descendants since it’s inception, for instance C++ and Objective C, as well as proven a great influence for almost every major programming language to come after it, such as Swift, Go, Odin, GLSL, PHP, Java, Javascript and C# to name but a few.

Conclusion: These days the use of C and the direct descendant C++ are quickly fading out of fashion. This is in large part due to a concern for memory-safety. One of the elements often considered hardest to master in C is the use of pointers, in addition to handling manual memory management correctly. These aspects reflect an emphasis on systems programming, allowing the user to get "closer" to the hardware, giving total control in the process of allocating and deallocating memory. This does however place a great deal of responsibility on the programmer, and as a consequence can lead to memory-leaks or safety issues if sufficient caution is not applied in coding practice. These safety concerns surrounding memory leaks led to the U.S. Cybersecurity and Infrastructure Security Agency (CISA) and the Federal Bureau of Investigation (FBI) co-publishing the report "Product Security Bad Practices" on October 16th 2024 with the recommendation for developers to stop using C and C++. This has in turn led to speculation in the industry that it might be a mark of the beginning of the end for "unsafe" languages such as C and C++, and that the time has come to switch to modern, memory-safe alternatives, such as Rust. It is nevertheless difficult to understate the importance of C and UNIX in computing history. C has proven a remarkable tool and seen use in mostly every computer architecture conceived and for almost every thinkable purpose: From building the kernel in the Windows operating system used to write this very text, to creating software used to explore distant terrestrial bodies. In addition to C powering a key part of Windows, it is worth mentioning that UNIX would branch off and evolve into most other operating systems that exist today: Linux, OS X and Android are all systems derived from UNIX. Furthermore, most programming languages in use at the time of writing are either descendants of C, languages built with C, or languages heavily inspired by, and often with a similar syntax to C. In conclusion C is a simple and small language, but a true giant in the history of computing.

Sources:

AT&T Tech Channel. (2012, February 22). *AT&T Archives: The UNIX Operating System* [Video]. YouTube. https://www.youtube.com/watch?v=tc4ROCJYbm0

Bergin, T. J. & Gibson, R. G. (1996). *History of programming languages II.* Addison-Wesley publ.

Computerphile. (2015, August 18). *"C" Programming Language: Brian Kernighan* [Video]. YouTube. https://www.youtube.com/watch?v=de2Hsvxaf8M

Computerphile. (2017, August 25). *Why C is so Influential* [Video]. YouTube. https://www.youtube.com/watch?v=ci1PJexnfNE

Cybersecurity and Infrastructure Security Agency & Federal Bureau of Investigation. (2024, October 16) *Product security bad practices.* https://www.cisa.gov/resources-tools/resources/product-security-bad-practices

Henderson, H. (2009). *Encyclopedia of computer science and technology* (Rev. ed). Facts On File.

ISO/IEC JTC1/SC22 - Programming languages and, operating systems, WG14 - C. (n.d.). Retrieved November 15, 2024 from https://www9.open-std.org/JTC1/SC22/WG14/

Kernighan, B. W. & Ritchie, D. M. (1988). *The C programming language* (2nd ed). Prentice Hall.

Lekanger, K. (2024, September 13). Frykter at C++ forsvinner – for en minnesikker løsning kan brekke alt. *Kode24.* http://www.kode24.no/a/81931243

National Academy of Engineering. (2016). *Memorial tributes: Volume 20.* The National Academies Press.