0.3 — Introduction to C/C++

BY ALEX ON MAY 27TH, 2007 | LAST MODIFIED BY ALEX ON AUGUST 22ND, 2017

Before C++, there was C

The C language was developed in 1972 by Dennis Ritchie at Bell Telephone laboratories, primarily as a systems programming language. That is, a language to write operating systems with. Ritchie's primary goals were to produce a minimalistic language that was easy to compile, allowed efficient access to memory, produced efficient code, and did not need extensive run-time support. Thus, for a high-level language, it was designed to be fairly low-level, while still encouraging platform-independent programming.

C ended up being so efficient and flexible that in 1973, Ritchie and Ken Thompson rewrote most of the UNIX operating system using C. Many previous operating systems had been written in assembly. Unlike assembly, which ties a program to a specific CPU, C has excellent **portability**, allowing UNIX to be recompiled on many different types of computers and speeding its adoption. C and Unix had their fortunes tied together, and C's popularity was in part tied to the success of UNIX as an operating system.

In 1978, Brian Kernighan and Dennis Ritchie published a book called "The C Programming Language". This book, which was commonly known as K&R (after the authors' last names), provided an informal specification for the language and became a de facto standard. When maximum portability was needed, programmers would stick to the recommendations in K&R, because most compilers at the time were implemented to K&R standards.

In 1983, the American National Standards Institute (ANSI) formed a committee to establish a formal standard for C. In 1989 (committees take forever to do anything), they finished, and released the C89 standard, more commonly known as ANSI C. In 1990 the International Organization for Standardization adopted ANSI C (with a few minor modifications). This version of C became known as C90. Compilers eventually became ANSI C/C90 compliant, and programs desiring maximum portability were coded to this standard.

In 1999, the ANSI committee released a new version of C called C99. It adopted many features which had already made their way into compilers as extensions, or had been implemented in C++.

C++

C++ (pronounced see plus plus) was developed by Bjarne Stroustrup at Bell Labs as an extension to C, starting in 1979. C++ adds many new features to the C language, and is perhaps best thought of as a superset of C, though this is not strictly true as C99 introduced a few features that do not exist in C++. C++'s claim to fame results primarily from the fact that it is an object-oriented language. As for what an object is and how it differs from traditional programming methods, well, we'll cover that in chapter 8 (Basic object-oriented programming).

C++ was ratified in 1998 by the ISO committee, and again in 2003 (called C++03). Two updates to the C++ language (C++11 and C++14, ratified in 2011 and 2014 accordingly) have been made since then, adding additional functionality to the language. Relevant features from both of these updates will be discussed in these tutorials.

C and C++'s philosophy

The underlying design philosophy of C and C++ can be summed up as "trust the programmer" -- which is both wonderful and dangerous. C++ is designed to allow the programmer a high degree of freedom to do what they want. However, this also means the language often won't stop you from doing things that don't make sense. There are quite a few pitfalls that new programmers are likely to fall into if caught unaware. This is one of the primary reasons why knowing what you shouldn't do in C/C++ is almost as important as knowing what you should do.

Note that you do *not* have to learn to program in C before doing these tutorials. We'll teach you everything you need to know (including pitfalls to avoid) along the way!