Compilers: a human-machine translator

As computer science students, the first programming language we are often introduced to is C++. C++ is a general purpose programming language and is a branch of C, like its other siblings such as C# and Objective-C. The languages are designed for different use cases, different devices and have different features, but they all have one thing in common: a compiler.

What is a compiler? When we create a program and type code into an IDE, we are writing what is known as “source code.” In C++, the source code is often saved in “.c” files. The “source code” is a high-level language, which means that it is designed to be human readable and can be easily edited in any text editor. However, computers do not understand “source code”, and a compiler is required to convert it into “machine code,” a low level language. The compiler takes the “.c” file and spits out an “.exe” file, an executable file that contains the “machine code.” This file can then be read by the computer and be executed.

Although compilers look simple and are fairly straightforward, building a good and efficient compiler is extremely hard. The compiler needs to read every single line of code, check syntax for errors and identify them, find the best way to manage resources and then convert them to binary “machine code.” In addition, the compiler needs to do those tasks as quickly as possible. Furthermore, the compiler also needs to be updated frequently to remove bugs or to keep up with its programming language whenever new features are added.

Sources

<https://www.softwareengineerinsider.com/programming-languages/cplusplus.html>

<https://whatis.techtarget.com/definition/compiler>

<https://www.quora.com/Why-are-compilers-so-hard-to-write>