**Main differences between C and C++**

Comparing C++ vs. C reveals a few **differences** between these two programming languages:

* C is a **procedural language**, while C++ is **object-oriented**. This feature refers to the programming style that developers follow. For instance, procedural programming follows step-by-step guidelines of functions, while object-oriented programming focuses on objects, inheritance, etc.
* C++ has **a well-designed exception handling** (Try and Catch blocks), which makes the debugging process easier than in C. This feature is especially useful for finding difficult errors. In C, **error handling occurs through functions**.
* **C++ also supports information** hiding (closely related to encapsulation).
* **Data is more secure in C++** than in C because C++ offers modifiers to limit user access.
* **C++ supports function overloading**, which means that a function with the same name can be declared for different purposes.
* **C++ also uses namespaces**, which let you organize code according to the desired scope. For instance, grouped entities can be put into a narrower scope referred to as namespace scope. C does not support this feature.
* Specialists relate C++ to the concept of **multi-paradigm**. Even though we classified C++ as an object-oriented language, it**has features of a procedural one as well**. Therefore, **C++ is more flexible than C** since C only follows the procedural logic.

Therefore, the comparison of C VS C++ syntax rules leads to a few important conclusions:

* **The use of C and C++ differs in a way that you will follow different programming approaches.**
* With C++, developers **can follow both procedural and object-oriented programming.**
* **C allows only procedural programming**.
* C++ offers more features such as **error handling, data security, scope management, information hiding, etc.**
* However, for beginners,**C language might be more straightforward** and helpful in terms of understanding the main concepts of low-level programming.

As you can see, the languages are quite different.

**Comparing Objective-C to C++**

Objective-C is a **general-purpose programming language** that enhances C with the Smalltalk-style messaging. It is mainly for **creating applications for iOS and**[**OS X**](https://en.wikipedia.org/wiki/MacOS)**operating systems**. It is another variation of the C programming language.

The question of Objective-C VS C++ emerges due to the fact they are **both object-oriented languages**that derived from C. Here is a comparison of these languages:

* **Objective-C does not offer multiple inheritance** while C++ does.
* There are differences in the way C++ and Objective-C **code looks**. For instance, C++ uses **true and false for bool**, while Objective-C works with **YES and NO**for BOOL.
* Objective-C has a **Smalltalk feature**, which is a messaging paradigm in which you transfer so-called messages to objects through functions or selectors.
* While C++ manages structs and classes the same, **Objective-C treats them completely differently.**
* Objective-C is **more dynamic**than the static C++ programming language.
* In theory, Objective-C **is slower than C++** because of the dynamic method dispatch.
* Objective-C is **mostly for creating applications for Apple products**, while **C++ is a more diverse language** applied in numerous development fields.

**Which should you learn?**

You might now ask: **should I learn C or C++?** Learning C++ first might seem like the best option because **it has more to offer than** C. However, C helps you understand how hardware, especially [**CPU**](https://en.wikipedia.org/wiki/Central_processing_unit), works due to the straightforwardness of this language.

You will get familiar with the low-level programming concept, learn about**pointers,**and **proper memory allocation**. Furthermore, C has fewer features that beginners will need to analyze.

However, the decision to learn C VS C++ first depends on **the way you want to work**. There is **no actual need to learn C before C++**. Even if you start by learning C++, you will probably encounter some of the similar elements in syntax and semantics.

Developers that start with C often complain that they need to get rid of the habits they learned in C to use C++ correctly. Learning C first **might make developers stick to procedural programming without exploring the object-oriented side**.

For instance, if your life-long dream is video game development, you are better off **starting to**[**learn C++**](https://www.bitdegree.org/goon/edx-c-plus-plus-course)**immediately.**

**Performance comparison**

Comparing C VS C++ in terms of performance usually leads to the fact that **C is faster than C++**. However, in certain situations, C++ can win this race as well. Indeed, **interpreted languages will never be faster than compiled ones**. However, determining the speed difference of the two compiled languages is tricky.

C VS C++ **speed highly depends** on the produced code overall. A well-written C++ code can perform better or the same as a well-written C code.

For instance, more robust programming will probably **be quicker in C++ than in C.**Therefore, specialists do not state that one language is faster than the other one. In the case of C VS C++ performance, it is often stated that C++ is a match for C.

C++ indeed offers many features that might lead to **poorer performance**, but it is not necessary to use them. Even if you do, the speed difference thanks to modern compilers probably won’t be significant enough to favor one programming language.