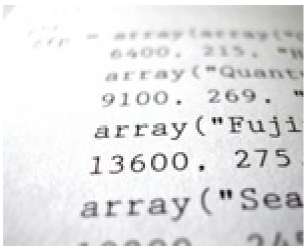
Why I Still Love C++

Author has been coding in C++ for a long time now. Even though from the start the language had certain warts he didn’t like, certain aspects of it were irresistable to him.

This article is actually a precursor to my article on the D programming language. I lay the foundations for my interest in D by talking about what I love about C++.



I have been coding in C++ for a long time now. Even though from the start the language had certain warts I didn’t like, certain aspects of it were irresistable to me and made it top dog.

1. An expressive language - Once learned, few languages are intrinsically as expressive as C++. It has a million nuances (like the English language) that somehow make expressing exactly what you want to say seem easier than other languages. Having true const support, multiple inheritance (in those rare circumstances where it makes sense) and references that can never be null are just a few of the unqiue ways that it supports saying exactly what you mean. Even if saying it sometimes may seem to the novice a little verbose and complicated. How many times have you checked for null in a Java or C# program and gritted your teeth knowing that the value should in fact NEVER be null. In C++ once you define a reference you have made a contract with the compiler expressing your intent. The language spec will not let you check for null.

2. Multi-paradigm - A Even though 10 years ago I jumped on the OO paradigm in a big way, single paradigm language never could quite cut it. Even though generics exist in other languages, few are as strong as C++. Simpler, yes. Powerful, no. Support for objects and generics is a must in my book. Supporting a simple functional style is important as well. Some concepts in the real world are both functional and freestanding. Granted - it is rare that I use freestanding functions, but there are times when they can reduce coupling in a big way. In object-insistant languages you are forced to wrap these guys in a class to “objectify” them. This is silly at best.

3. Supporting low level constucts - someday this will go away. But doing a good amount of 3D coding I still like to be able to optimise things pretty tightly sometimes. The day has come and gone when assembly or flat C was a must in this category. But C++ is still very useful in this way.

4. Huge opensource support - let’s face it C & C++ are still the reigning kings here.

5. Ability to interface with C & C++ effortlessly - given this “isn’t quite fair”. Of course C++ works well with C and C++. This is important because of #4. Many langauges support clumsy bindings to C++ and C and this adds a layer of inconvenience to “get at” the wealth of opensource code already out there.
Many other aspects of C++ have been absorbed by other languages. I like the availibility of generics. Being able to force strong typing when it is important is key to me - like const correctness. You know exactly what you want. Being able to also use generics and avoid strong typing is equally important. Your intent is clear. C# is a close runner up for me in many respects - the two main things that makes it difficult to fully embrace include clumsy bindings with C++ and its lack of ubiquity on other platforms. The thought of writing a large body of code in C# and then having to rewrite it for say the Wii or a handheld makes me cringe. Mono may help, but as far as I’m aware the jury is STILL out on how widely accepted it will be.

For the certain types of coding C++ is still essential to me. It’s a love hate relationship. I look forward to the day when everything I enjoy about C++ is in a language that removes the things I truly dislike.

In this article’s follow up I’ll talk about the D programming language. You might be pleasantly surprised by what it brings to the table. There I will mention the things that D does right and C++ clearly does not.