Subject: Re: C++ FQA
Posted by cbpporter on Wed, 07 Nov 2007 13:57:59 GMT
View Forum Message <> Reply to Message

Quote:
Yeah, I like it too but there is a paper about modules in new C++ too http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2006/n207 3.pdf . I do not know how much similar (or not) they are to D modules but it seems to be a step in the right direction.

Quote:
I think new C++ should break compatibility and be more like D. I don't understand why it cannot be since all current/old apps can be developed with old compilers. This way C++ will be fatter and fatter (and more complicated) with each standard revision.

That sound great and is pretty similar with the modules from D, with the only difference that they are already present in D, and by the time these features will be included in C, we'll all be retired programmers. I hate to be pessimistic, but if history has though us anything is that those behind C++ are a bunch of close minded people who have no intention to break compatibility and will gladly bloat the language with absolutely retarded and redundant features. Just have a look at C++0x, and see that most stuff just tries to fix old problems with new redundant features and by keeping the old ones. And even if they do introduce modules, by the time there will be a compliant compiler, I hope that C++ will be less popular for real-life application development.

Quote:
I prefer RAII approach and U++ is a very good example that this really works. Frankly if you use NTL or STL (and follow the RAII way) there is a rare situation when you have to worry about memory management. I don't know why this still is an issue.

It's probably because I'm not as skilled with high level C++ features, but the issues with memory management are quite present for me in U++.
While the language has deep copy, value references, copy constructors, etc., I think that it is impossible not to have such issues. And in U++ I spend a great deal of time just messing with const correctness and other low level details which I shouldn't be forced to worry about.

Quote:
Again, I just don`t like an idea of uncontrolled garbage collection. Developing some time-cricical programms for industrial automation, I dislike the fact that my time-critical code can be interrupted or slowned down by some uncontrolled process of garbage collection.

Well in D you can always chose not to use garbage collection for a class or even just a part of your code. You can use C style pointers, and even call malloc if there is a need to. But I consider this some kind of a myth. Has there been a documented case where garbage collection had serious impact on performance. It is true that such applications sometimes have the bad habit of hanging for 2-3 seconds when a garbage collection cycle starts, but this shouldn't be an issue in non GUI applications. And U++ and even STL does a lot of "garbage collection", only it is more deterministic, but not necessarily faster.