Subject: Re: C++ FQA
Posted by unodgs on Wed, 07 Nov 2007 12:15:07 GMT
View Forum Message <> Reply to Message

Quote:
1. True module support.
No more including hundreds of kilobytes if not megabytes of header files in each compilation unit. No more writing every ddecalration twice. Modules act like Java packages, can be fully qualified to avoid name clashes and can be combined to create a library with different access levels. And because each module is compiled only once, D is lightning fast. My entire project compiled from scratch almost instantly, and if I only modified the content of a function or other miner detail, truly instantly.

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:
2. A lot of built in features, like variable length array and hash maps, which combined with some extra functions, can be used to create types like Vector or Map, but even more easy to use and without templates. char[] and about 10 extra functions could make String and StringBuffer obsolete.
True. But I remember that D-people wanted to have String class. I prefere it too even if bulit-in char type is powerful and easy to use.
Quote:
3. Templates + mixins + static ifs are stronger than templates in C++ + preprocessor. I never used them, because you can do in D a lot more without templates than in C++.

Yes, some D coders (like Don Clugston for example) proved they are much more powerful than C++ ones.
Quote:
4. Optional (but defaulted to true) garbage collection.
As much as you could dislike the idea of garbage collection, memory management in C++ is a nightmare, and from this point of view, even U++ which has a lot less such issues, is not able to give such pain free management.

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.
D uses GC but fortunately it allows for deterministic destruction in "scope classes". At least in theory Must check it.

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.