

Re: Boost Workshop at OOPSLA 2004

Source: http://coding.derkeiler.com/Archive/C_CPP/comp.lang.cpp/2004-08/1941.html

From: David Abrahams (dave_at_boost-consulting.com)

Date: 08/13/04

Date: 13 Aug 2004 12:49:41 -0400

"Andrei Alexandrescu \((See Website for Email)\)" <SeeWebsiteForEmail@moderncppdesign.com> writes:

> *"David Abrahams" <dave@boost-consulting.com> wrote in message*
> *news:uzn51rbml.fsf@boost-consulting.com...*
> > *"Andrei Alexandrescu \((See Website for Email)\)"*
> > > *Today Boost uses a "preprocessor library", which in turn (please*
> > > *correct me if my understanding is wrong) relies on a program to*
> > > *generate some many big macros up to a fixed "maximum" to overcome*
> > > *preprocessor's incapability to deal with variable number of*
> > > *arguments.*
> >
> > *That's a pretty jumbled understanding of the situation.*
> >
> > *The preprocessor library is a library of headers and macros that allow*
> > *you to generate C/C++ code by writing programs built out of macro*
> > *invocations. You can see the sample appendix at*
> > *<http://www.boost-consulting.com/mplbook> for a reasonably gentle*
> > *introduction.*
>
> *Ok, it's a half jumbled understanding of the situ, coupled with a half*
> *jumbled expression of my half-jumbled understanding :o).*
>
> *First, I've looked in my boost implementation to see things like*
> *BOOST_PP_REPEAT_1_0 to BOOST_PP_REPEAT_1_256 and then BOOST_PP_REPEAT_2_0 to*
> *BOOST_PP_REPEAT_2_256 and so on. My understanding (which I tried to convey*
> *in my post) is that such macros are generated by a program.*

Yes, but as I mentioned none of that is required in std C++.

<http://sourceforge.net/projects/chaos-pp/> doesn't use any program-generated macros.

> *That program is admittedly not part of the library as distributed (I*
> *believe it is part of the maintenance process), but I subjectively*
> *consider it a witness that a more elegant approach would be welcome.*

Yeah, I'd rather be using Chaos everywhere instead of the current Boost PP lib. Too bad it isn't portable in real life.

- > *Then I've looked again over the PP library (this time through the link*
- > *you've sent), and honestly it reminds me of TeX macro tricks more than any*
- > *example of elegant programming.*

Where are the similarities with TeX macro tricks?

- > *As such, I'd find it hard to defend it with a straight face, and I*
- > *am frankly surprised you do.*

You're surprised I defend the PP library based on the fact that it reminds `_you_` of TeX macros?

The PP lib provides me with an expressive programming system for code generation using well-understood functional programming idioms. In the domain of generating C++ from token fragments, it's hard to imagine what more one could want other than some syntactic sugar and scoping.

- > *But then I understand the practical utility, as you point out below.*
- > > *Bjarne's plan for that is to gradually make the capabilities of the*
- > > *existing PP redundant by introducing features in the core*
- > > *language... and then, finally, deprecate it.*
- >
- > *It's hard to introduce the ability to define syntactic replacement*
- > *(which many people consider useful) in the core language.*

Right. I personally think the PP will always have a role. That said, I think its role could be substantially reduced.

- > > *I doubt even with Boost backing that the community at large is likely*
- > > *to easily accept integrating another tool into its build processes.*
- > > *The big advantage of the C++ PP is that it's built-in... and that's*
- > > *one of the biggest reasons that the PP `_lib_` is better for my purposes*
- > > *than any of the ad hoc code generators I've written/used in the past.*
- >
- > *Practicality, and not elegance or suitability, is about the only*
- > *reason that I could agree with.*

Practicality in this case is elegance. My users can adjust code-generation parameters by putting `-Dwhatever` on their command-line.

FWIW, I designed a sophisticated purpose-built C++ code generation language using Python and eventually scrapped it. Ultimately the programs I'd written were harder to understand than those using the PP lib. That isn't to say someone else can't do better... I'd like to see a few ideas if you have any.

- > > *I think Bjarne's approach is the best way to do that sort of*
- > > *replacement. As long as the PP's functionality is really being*

> > replaced by a textual preprocessor (or a token-wise one as we have
> > today) it's going to suffer many of the same problems. Much of those
> > jobs should be filled by a more robust metaprogramming system that's
> > fully integrated into the language and not just a processing phase.
>
> I think here we talk about different things. One path to pursue is
> indeed to provide better means for template programming and another
> is to provide syntactic manipulation. To me, they are different and
> complementary techniques.

Metaprogramming != template programming. In meta-Haskell, they actually manipulate ASTs in the core language. As I understand the XTI project, it's going in that sort of direction, though a key link for metaprogramming is missing.

> > > Maybe namespaces that are so badly designed, you'd think they are
> > > inherited from C?
> >
> > Wow, I'm impressed; that's going to piss off both the hardcore C _and_
> > C++ people!
>
> Heh heh... I knew this is gonna be taken that way :o). What I meant was,
> many shortcomings of C++ root in a need for compatibility with C. With
> namespaces and export, there's no C to blame :o).

I don't know about that. Isn't C's inclusion model a big part of the reason that namespaces are not more like modules?

> > I've never seen a serious proposal for better namespaces, other than
> > <http://boost-consulting.com/writing/qn.html>, which seems to have been
> > generally ignored. Have you got any ideas?
>
> That's a good doc solidly motivated; I am sorry it does not get the
> attention that it deserves.

Thanks. Maybe I should re-submit it.

> > > So, a proposal for a new preprocessor would be great.
> >
> > If that's your point, I think it's an interesting one, but somehow I
> > still don't get how it could be appropriate for a workshop on C++
> > libraries.
>
> Ok, I'll drop it.

If you have PP library ideas, by all means bring those up.

> May I still bicker about it on the Usenet? :o)

It's your dime ;-)

comp.lang.c++. Re: Boost Workshop at OOPSLA 2004

--

Dave Abrahams

Boost Consulting

<http://www.boost-consulting.com>

[See <http://www.gotw.ca/resources/clcm.htm> for info about]
[comp.lang.c++.moderated. First time posters: Do this!]