Subject: Eigen and UPP? (STL question?)
Posted by GaroRobe on Fri, 13 May 2011 01:43:16 GMT
View Forum Message <> Reply to Message

Hi, %all%!

Gotta make a remark - I haven't been coding on C++ for, literally, YEARS and never actually used STL, thus all this stuff I'm trying to grasp while having to do an actual job using this is pretty confusing.

Currently I've got to implement some pretty complex CCTV-like system from scratch and I'll need to use some heavy math on images. Surfed a toolkit called Eigen (http://eigen.tuxfamily.org) that seems to fit my needs.

Trying to build the very first Eigen code example:

```
#include "Test_1.h"
#include <Eigen/Dense>

using Eigen::MatrixXd;

CONSOLE_APP_MAIN
{
 MatrixXd m ( 2, 2 );
 m ( 0, 0 ) = 3;
 m ( 1, 0 ) = 2.5;
 m ( 0, 1 ) = -1;
 m ( 1, 1 ) = m ( 1, 0 ) + m ( 0, 1 );
 Cout() << m;
}
```

and fail:

Quote:c:\upp\uppsrc\core\String.h(400) : error C2039: 'ToString' : is not a member of 'Eigen::Matrix<_Scalar,_Rows,_Cols>'

which is not very surprising, but unpleasant nevertheless.
My guess is that m (in original example "std::cout << m << std::endl;") can't be converted to String, but I don't get how to solve this. Could you please give me a hand here?

Subject: Re: Eigen and UPP? (STL question?)
Posted by dolik.rce on Fri, 13 May 2011 06:43:22 GMT
View Forum Message <> Reply to Message

Hi GaroRobe!

Welcome to U++ Forum!

First, regarding your remark: Not using STL in past will actually make it simpler for you to get familiar with U++, since you won't have to get rid of the old habbits

Now to the code: Simply put, the operator<< in U++ uses by default function AsString(), which transform the given object to Upp::String. This is in most cases done by just calling the objects ToString() method, but in cases of foreign types, like those from Eigen, you have to make a specialization of AsString, which would understand it. The simplest posible code to make this work (altough not the best, but I didn't have much time to study Eigen internals) can be this: #include <Core/Core.h>
#include <Eigen/Dense>

```
using namespace Upp;
using Eigen::MatrixXd;

NAMESPACE_UPP
template<>
String AsString(const MatrixXd& m) {
 std::stringstream tmp;
 tmp << m; // we just use eigen classes capability to write to std::ostream
 return tmp.str(); // here the std::string is 'magicaly' converted to Upp::String
}
END_UPP_NAMESPACE

CONSOLE_APP_MAIN
{
 MatrixXd m ( 2, 2 );
 m ( 0, 0 ) = 3;
 m ( 1, 0 ) = 2.5;
 m ( 0, 1 ) = -1;
 m ( 1, 1 ) = m ( 1, 0 ) + m ( 0, 1 );
 Cout() << m << '\n';
}
```

Hopefully this will give you an idea

Best regards,
Honza

Subject: Re: Eigen and UPP? (STL question?)
Posted by GaroRobe on Fri, 13 May 2011 11:18:48 GMT
View Forum Message <> Reply to Message

It worked, but... hey, what did it do?

I mean... what does "NAMESPACE_UPP" means and why can't I define similar template for std::stringstream& instead of MatrixXd&?

Well, anyway I'll have to do some serious learning on strings, values and containers in U++ (reading once obviously just wasn't enough)

Seems like I'll have quite a number of questions along the way

---

Subject: Re: Eigen and UPP? (STL question?)
Posted by mr_ped on Fri, 13 May 2011 13:10:55 GMT
View Forum Message <> Reply to Message

Eigen has it's own Matrix type, which has it's own "to std::string" conversion routine.
Which makes std::cout << Eigen::MatrixXd to compile without problem.

Cout() is not std::ostream (or what's the base output stream in STL/C++, I have no idea), it's Upp::output stream (not sure about exact class name either).

So the compiler can't find the conversion function from Eigen::MatrixXd to Upp::String (which is most common input format for "<<" for Upp::ostream.

And dolik did just supply you with one conversion function for such situation.

NAMESPACE_UPP is macro to open Upp:: namespace, so you are with the next code defining
Upp::String Upp::AsString(const Eigen::MatrixXd& m)
function.
Then later during compiling "Cout() << m" that new method is perfect fit for silent auto conversion.

Subject: Re: Eigen and UPP? (STL question?)
Posted by dolik.rce on Fri, 13 May 2011 14:41:10 GMT
View Forum Message <> Reply to Message

GaroRobe wrote on Fri, 13 May 2011 13:18It worked, but... hey, what did it do?

I mean... what does "NAMESPACE_UPP" means and why can't I define similar template for std::stringstream& instead of MatrixXd&?Well "NAMESPACE_UPP ... END_UPP_NAMESPACE" is just a fancy way to say "namespace Upp { ... }", it uses macro to allow compiling everything without namespace, when flag NONAMESPACE is used. Here it is necessary because AsString is templated function defined originally in Upp namespace.

Now what did it do Everything in Eigen uses this operator (defined in Core/IO.h in Eigen sources):template<typename Derived>
std::ostream & operator <<
(std::ostream & s,
 const DenseBase<Derived> & m)
{
  return internal::print_matrix(s, m.eval(), EIGEN_DEFAULT_IO_FORMAT);
}As you can see it uses std::ostream (and print_matrix() as well) which is not compatible with U++

streams. So we used this operator to output the matrix into stringstream (which inherits from ostream), then convert it to std::string by calling str() method and this string is then converted to Upp::String (using implicit conversion in the return statement). Now the whole machinery that lies behind the U++ operator<< can use the AsString specialization to convert the matrix to Upp::String, which can be used in the Cout << ... expression. I'm not sure if this is understandable, but looking at operator<<() definition (at Core/Stream.h:650) might help

I'm 90% sure there is a better solution, I just didn't have time to investigate Eigen deep enough. I am quite interested in having nice algebra toolkit such as eigen available in U++. It would be great if you could make a package that would provide some of the basic functionality needed to better integrate eigen into U++, such as this << operator etc... Of course, I'll offer you help and advice on this

GaroRobe wrote on Fri, 13 May 2011 13:18Well, anyway I'll have to do some serious learning on strings, values and containers in U++ (reading once obviously just wasn't enough)

Seems like I'll have quite a number of questions along the way Some parts of the U++ docs have to be read 10+ times to be fully understood It migh sound discouraging, but this forums members are very helpful and ready to answer the questions

Honza

EDIT: I see mr_ped once again answered faster than me... Thankfully, the his answer is basicaly the same as mine

---

Subject: Re: Eigen and UPP? (STL question?)
Posted by GaroRobe on Sun, 15 May 2011 06:15:50 GMT
View Forum Message <> Reply to Message

Okay, here comes the hell.

I expect using some math-related datastructures tightly bound with the need to print them out eventually ( ) thus I'll clearly need either a very generic implicit convertor like yours or a numerous (insane amount actually) of them.

I'm a noob at generic-C++ (just started reading Alexandrescu and don't expect to get skilled enough anywhere soon) and making sense of Eigen sources is nothing but a headache to me so far (yep, definetely nothing like Borland...).

But my question this time is "CAN'T I EASILY EXPLORE OUTSIDE CODE IN THEIDE?"

When I include Eigen headers in my scope I expect to jump to prototypes with Alt+J or get inline helper with Ctrl+Space, but for Eigen routines it just doesn't work.

Can I make it work somehow? It's very difficult otherwise.

Subject: Re: Eigen and UPP? (STL question?)
Posted by dolik.rce on Sun, 15 May 2011 09:59:09 GMT
View Forum Message <> Reply to Message

GaroRobe wrote on Sun, 15 May 2011 08:15Okay, here comes the hell.

I expect using some math-related datastructures tightly bound with the need to print them out eventually ( ) thus I'll clearly need either a very generic implicit convertor like yours or a numerous (insane amount actually) of them.That is what I was talking about when I said that better solution will be needed I already looked at the code in eigen and all the containers seem to be derived from Matrix<T>, so the best approach will probably be to make it work for this one class and let the inheritance do its work However, since it is a templated class, it is not easily possible to write specializations for templated functions such as AsString... I have few ideas how to trick it to work, but I haven't time to try it yet (maybe tonight).

GaroRobe wrote on Sun, 15 May 2011 08:15But my question this time is "CAN'T I EASILY EXPLORE OUTSIDE CODE IN THEIDE?"

When I include Eigen headers in my scope I expect to jump to prototypes with Alt+J or get inline helper with Ctrl+Space, but for Eigen routines it just doesn't work.

Can I make it work somehow? It's very difficult otherwise.
This can be done by creating a package containing the Eigen sources. This is BTW a preferred way in U++, as it allows for better portability (you don't have to make sure Eigen is installed on other computers when you distribute your source code, just the exported package) and reliability (you know that the code will be always used with the same version of the library). See e.g. plugin/png if you want to see an example of this.

By the way, what version of Eigen are you using? I noticed some (rather important) differences between versions 2.x and 3.x...

Honza

Subject: Re: Eigen and UPP? (STL question?)
Posted by GaroRobe on Sun, 15 May 2011 11:43:24 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Sun, 15 May 2011 20:59That is what I was talking about when I said that better solution will be needed I already looked at the code in eigen and all the containers seem to be derived from Matrix<T>, so the best approach will probably be to make it work for this one class and let the inheritance do its work However, since it is a templated class, it is not easily possible to write specializations for templated functions such as AsString... I have few ideas how to trick it to work, but I haven't time to try it yet (maybe tonight).

Well, I grasped that much (though three-layered-eight-parameters-long-templated-classes-generated-with-three-folded-macroes are clearly beyond my comprehension), but what I really meant is that there are not only matrices out there - right? I hoped finding some common predecessor or

template param, which controls this behaviour, but failed miserably and scrapped this idea for the time being. Too difficult for me. Guess I'm stuck with basic workarounds for now (hate leaving stuff with a mark "refactor when I'm a guru"...).

dolik.rce wrote on Sun, 15 May 2011 20:59
This can be done by creating a package containing the Eigen sources. This is BTW a preferred way in U++, as it allows for better portability (you don't have to make sure Eigen is installed on other computers when you distribute your source code, just the exported package) and reliability (you know that the code will be always used with the same version of the library). See e.g. plugin/png if you want to see an example of this.
Now that's a plan ! Now I'll look into example, but could you give me some more solid docs on the subject? Like reference.

dolik.rce wrote on Sun, 15 May 2011 20:59
By the way, what version of Eigen are you using? I noticed some (rather important) differences between versions 2.x and 3.x...
3.0 package, using 3.x interface which I plan on using on.

Subject: Re: Eigen and UPP? (STL question?)
Posted by dolik.rce on Sun, 15 May 2011 12:15:53 GMT
View Forum Message <> Reply to Message

Well, judging from the code in IO.h it seems that only things derived from Matrix are capable of being outputted, unless there is something fairly well hidden in some obscure place And pretty much anything can be described as matrix in algebra, so it is not really surprising

I'll try to prepare the package and see if I can figure out how to integrated it with U++ in more general way tonight and post it here when I have something useful.

Also if you have any U++ or C++ related questions (e.g. about those heavily templated functions), don't hesitate to ask (in appropriate part of the forum).

Honza

Subject: Re: Eigen and UPP? (STL question?)
Posted by dolik.rce on Sun, 15 May 2011 15:52:02 GMT
View Forum Message <> Reply to Message

Here is the plugin package, just unpack the plugin directory into your MyApps directory and add package plugine/Eigen to your main package.

Very good news: Eigen is amazingly friendly to third party extensions. It lets you add anything to all of the base classes using EIGEN_*_PLUGIN macros. So I just added method ToString into the Matrix class, which result in all types of matrices to be easily printable. I believe that this mechanism will allow us to solve also most of the future incompatibility troubles you might

encounter.

Let me know when you hit next problem, I really enjoy this

Honza

EDIT: Added new version of the package, Eigen now respects SSE2 flag...

File Attachments

1) eigen.zip, downloaded 319 times

Subject: Re: Eigen and UPP? (STL question?)
Posted by GaroRobe on Mon, 16 May 2011 01:28:00 GMT
View Forum Message <> Reply to Message

Beautiful solution! Why did I not pay attention to *_PLUGIN construction?

Well, that greatly simplifies interaction with future logic implementation But I guess basics still must be learned.
