

Re: possibility to forbid use of "this"?

Source: http://coding.derkeiler.com/Archive/C_CPP/comp.lang.cpp/2004-01/0796.html

From: tom_usenet (tom_usenet_at_hotmail.com)

Date: 01/07/04

Date: Wed, 07 Jan 2004 14:23:07 +0000

On Wed, 7 Jan 2004 14:53:45 +0100, "Ernst Murnleitner"
<mur-spam@awite.de> wrote:

>Dear Readers,
>
>Is it possible to forbid conversion from this or use of this in general
>except where it is explicitly wanted?
>
>Reason:
>
>I changed my program from using normal pointers to classes A, ...
>
>typedef A * APtr;
>
>to a shared pointer
>
>typedef boost::shared_ptr<A> APtr;
>
>Now, it crashes because of statements like this
>
>// call
>DoSomething(this);
>
>....
>// implementation
>void DoSomething(APtr a)
>{
>// do nothing with a
>}

That shouldn't compile – the constructor of shared_ptr taking a T* is explicit, so a pointer can't implicitly convert to a shared_ptr.

>Obviously, "this" is converted to a shared ptr locally. Outside the function
>DoSomething() the shared ptr is destroyed and hence it tries to delete the
>class where "this" points, too. This is clearly not wanted.

comp.lang.c++. Re: possibility to forbid use of "this"?

You have a few choices, depending on what DoSomething does. If DoSomething doesn't hold onto a reference to a, then you can change it to:

```
void DoSomething(A& aref)
{
}
```

If it does hold onto a reference to a, but you know it will release that reference before a is destroyed (which you probably can't guarantee), then you could do:

```
struct null_deleter
{
    void operator()(void const *) const
    {
    }
};

//...

DoSomething(APtr(this, null_deleter()));
```

The safe approach is to make a shared_ptr from the this pointer. This is only possible if the A object was created as a shared_ptr in the first place. Follow the instructions here:

http://www.boost.org/libs/smart_ptr/sp_techniques.html#from_this

Tom

C++ FAQ: <http://www.parashift.com/c++-faq-lite/>

C FAQ: <http://www.eskimo.com/~scs/C-faq/top.html>