CoCoALib - Bug #1726

Dangling references to temporaries

08 Mar 2023 21:57 - Jerry James

Status:	Closed	**Start date:**	08 Mar 2023
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Portability	**Estimated time:**	3.75 hours
Target version:	CoCoALib-0.99850	**Spent time:**	3.70 hours

Description

The Fedora Linux project has built all packages with GCC 13, which will be released soon. I see warnings of this form in the cocoalib build logs:

```
PolyRing.C: In function 'CoCoA::RingElem CoCoA::FixedDivisor(ConstRefRingElem)':
PolyRing.C:306:17: warning: possibly dangling reference to a temporary [-Wdangling-reference]
  306 |     const ring& k = CoeffRing(P);
      |                 ^
PolyRing.C:306:30: note: the temporary was destroyed at the end of the full expression 'CoCoA::Coe
ffRing(CoCoA::PolyRing((* & P)))'
  306 |     const ring& k = CoeffRing(P);
      |                     ~~~~~~~~~^~~
```

In each case, a method is called that returns a constructed value, rather than a reference to an existing value. GCC creates a temporary to hold the constructed value, computes a reference to it, stores the reference in the indicated variable ... and then destroys the temporary. The reference now points to invalid memory. If you are lucky, the stack space for the temporary is not reused for some other purpose elsewhere in the calling function. If you are unlucky, it is reused and now you have a weird, difficult to diagnose memory corruption bug.

I will attach a patch that eliminates all such warnings by changing the affected variables to hold values instead of references. The patch is against the 0.99800 release.

Related issues:

Related to CoCoALib - Feature #1793: Use ErrorContext instead of string FnName	**In Progress**	**16 Mar 2024**

History

#1 - 09 Mar 2023 09:43 - John Abbott

Thanks for letting us know! We are about to release a new version of CoCoA... hopefully tomorrow or early next week.

I am puzzled by this. At the moment I do not see what is causing trouble here -- which temporary will go out of scope?
CoeffRing returns a reference to an internal data member of P which is a local variable with lifetime/scope longer than that of k.

#2 - 09 Mar 2023 20:16 - John Abbott

- Category set to Portability

- Status changed from New to In Progress

- Target version set to CoCoALib-0.99850

- % Done changed from 0 to 10

I am still perplexed by these warnings. I'm pretty sure the code is safe: all it is doing is giving a convenient short name to a data member of some "long-lived" object.

Currently I have Ubuntu 22.04 with gcc/g++ 11.3.0. I have also tried compiling with clang++ (v 14.0.0), *no such warnings were produced* even with **-Wall -pedantic** flags ... other warnings were produced, I admit.

I would definitely like to know why the warnings were issued. It is possible that the compiler is mistaken (but that is rather unlikely). Perhaps I have misunderstood some aspect of C++, or the definitions have changed?

I'll see if I can find someone to reproduce the warnings... a colleague runs Arch Linux, so he probably has the latest version of gcc.

#3 - 07 Mar 2024 21:18 - John Abbott

Maybe I have understood what is upsetting the compiler.
The variable P is a ring&, but CoeffRing expects an argument of type const PolyRing&. There is an automatic conversion from const ring& to PolyRing, which does indeed create a new (temporary) object whose member function myCoeffRing is called, which returns a reference to the coeff ring object inside the poly ring object referred to by the "smart pointer" of type PolyRing. So the "smart pointer" is a temporary object referring to a non-temporary data-structure, and the value obtained is a reference to part of the non-temporary data-structure.

So the operation is safe, but it is not surprising that the compiler felt that a warning should be issued.

To be honest, I'm not sure what a good/clean way to resolve this is... I'll look at your patch.

#4 - 11 Mar 2024 10:48 - John Abbott

After discussing, we think my approach makes sense. So I must look at all messages in the patch.

#5 - 14 Mar 2024 20:46 - John Abbott

- Assignee set to John Abbott

- % Done changed from 10 to 60

This is one of those mind-numbing tasks. I think I have made most of the changes, but there is still plenty of cleaning up to do. By accident I ran all the CoCoALib tests with debugging on... they all passed! So that's perhaps a happy accident!

#6 - 15 Mar 2024 20:08 - John Abbott

- Status changed from In Progress to Resolved

- % Done changed from 60 to 80

Resolved, and checked in. But still need to so some cleaning... (sigh!)

#7 - 16 Mar 2024 11:07 - John Abbott

- Related to Feature #1793: Use ErrorContext instead of string FnName added

#8 - 18 Mar 2024 21:48 - John Abbott

Hi Jerry James
Could I send you a TGZ of the latest sources, so that you can try compiling, and see whether there are still worrisome error messages? The TGZ would be about 4Mbytes long. We are hoping to make a new source release soon, and it would be handy to have advance feedback in case of problems.

#9 - 22 Mar 2024 09:35 - John Abbott

- Status changed from Resolved to Feedback

- % Done changed from 80 to 90

#10 - 25 Mar 2024 18:05 - John Abbott

- Status changed from Feedback to Closed

- % Done changed from 90 to 100

- Estimated time set to 3.75 h

I believe I have made all necessary changes. So closing -- it works fine for us...

Files

cocoalib-dangling-ref.patch	12.2 KB	08 Mar 2023	Jerry James