CoCoALib - Design #763

GlobalManager: initialization compatible with initialization of external libs

01 Sep 2015 11:38 - John Abbott

Status:	In Progress	**Start date:**	01 Sep 2015
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	10%
Category:	Portability	**Estimated time:**	0.00 hour
Target version:	CoCoALib-1.0	**Spent time:**	2.20 hours

Description

[during meeting in Aarhus]
Anders asked about the following scenario:
a program uses both CoCoALib and GfanLib, and during initialization of both libs both attempt to specify a memory manager for GMP. What happens?

Check that CoCoALib initialization behaves nicely with initialization of GfanLib (and others).

Related issues:

Related to CoCoALib - Feature #762: ExternalLib-GFan: first prototype	**Closed**	**28 Aug 2015**
Related to CoCoALib - Feature #482: Unique copies of rings -- smart ctor	**In Progress**	**19 Mar 2014**
Related to CoCoALib - Design #785: finite fields: global register of fields a...	**New**	**12 Oct 2015**

History

#1 - 01 Sep 2015 11:39 - John Abbott

- Category set to Portability

- Target version set to CoCoALib-0.99540 Feb 2016

There is no problem with libnormaliz because it does nothing fancy.

#2 - 01 Sep 2015 12:14 - John Abbott

Here is why the situation could be quite "delicate".

Prior to using CoCoALib and GfanLib some initialization must be completed: for CoCoALib one must create a GlobalManager object, and for GfanLib (currently) some initialization is automatic (via ctors for globals).

Specifically GfanLib sets special memory managers for GMP, and CoCoALib can also set special memory managers (given the right args to the ctor for GlobalManager). So in a program which uses both there is a clash as to which memory managers to use for GMP (the last to set the memory managers wins).

However, the ctor for GlobalManager also creates RingZZ and RingQQ which internally allocate their 0 and 1 elements, which in turn will allocate GMP values. These values will then be destroyed during the dtor for GlobalManager (to achieve proper clean up).

We have to be sure that the right memory managers are active when the GMP values allocated inside RingZZ and RingQQ are deallocated.

#3 - 21 Mar 2016 14:58 - John Abbott

- Target version changed from CoCoALib-0.99540 Feb 2016 to CoCoALib-1.0

This looks to be quite a "delicate" matter. Even if I persuade Anders not to use globals for initializing his library, but local variables (as CoCoALib does), then there could still be problems... Consider the following chronology:

[1] Initialize CoCoALib (assoc to CoCoA::GlobalMgr) -- creates RingZZ which allocates some GMP values using CoCoA::GMPAllocator.
[2] Initialize GFanLib (assoc to GFan::GlobalMgr) -- changes mem mgr fns for GMP.
(A) Create more elements of RingZZ, values now allocated using GFanLib mem mgr.
(B) Create ring ZZ[x]; this will create zero and one elements "inside" the ring.
[3] Destroy GFan::GlobalMgr -- resets GMP allocators to CoCoA::GMPAllocator.
[4] Destroy CoCoA::GlobalMgr -- destroys all surviving rings.

If the elements of RingZZ created in step **(A)** are in local variables then they should be destroyed before we reach step **[3]**, and no nasty surprises should occur.

When is the ring ZZ[x] created in step **(B)** destroyed?
If it too has "local scope" then it will be destroyed before reaching step **[3]**; if however it survives until after step **[3]** then a nasty surprise will occur when it is destroyed in step **[4]** since its zero and one elements were allocated using GFan's mem mgr, but GMP has now been told to use CoCoA::GMPAllocator. **OUCH!!!**

IMPORTANT NOTE this last observation suggests that having a global register of rings would be a bad idea (*i.e.* it would likely trigger the nasty surprise); see issues #785 and #482

#4 - 21 Mar 2016 15:14 - John Abbott

- Related to Feature #482: Unique copies of rings -- smart ctor added

#5 - 21 Mar 2016 15:14 - John Abbott

- Related to Design #785: finite fields: global register of fields already created? added

#6 - 21 Mar 2016 16:55 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

This is not really a bug... the extant mechanism for cleaning up global variables is likely to cause trouble in the scenario presented in comment 3 since the creating/destruction of values does not follow a proper "nested scoping" arrangement. That using global variables is problematic should come as no surprise; however the scenario above suggests that the problems could be quite subtle (and hard to debug).

Nevertheless, global variables may be a good solution in small stand-alone programs which use just CoCoALib (with GMP) and no other external library; in that situation there should be no problems.