CoCoALib - Design #891

Replace auto_ptr in preparation for C++11?

17 Jun 2016 22:17 - John Abbott

Status:	Closed	**Start date:**	17 Jun 2016
Priority:	High	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Portability	**Estimated time:**	3.33 hours
Target version:	CoCoALib-0.99650 November 2019	**Spent time:**	3.20 hours

Description

Compiling CoCoALib with --std=c++11 produces lots of warnings about std::auto_ptr being deprecated

Reportedly std::auto_ptr will be removed from C++17.

Consider ways to make CoCoALib source code compatible (or easily upgradable) with C++03 and the newer standards.

Related issues:

Related to CoCoA-5 - Design #83: C++11 compatibility questions	**In Progress**	**26 Jan 2012**
Related to CoCoALib - Feature #82: C++11 compatibility questions	**Closed**	**26 Jan 2012**
Related to CoCoALib - Design #1225: Move to C++14 (skipping C++11)	**In Progress**	**06 Sep 2018**

History

#1 - 17 Jun 2016 22:17 - John Abbott

- Related to Design #83: C++11 compatibility questions added

#2 - 17 Jun 2016 22:23 - John Abbott

Obviously we are not the first to encounter this problem.

My understanding is that std::unique_ptr is intended as a "drop in replacement" for std::auto_ptr (provided that the latter has been used cleanly).

I would strongly prefer a solution which does not use #define.

I'm not sure if typedef can be used easily with templated types.
If so, we could just use a typedef for CoCoA::UniquePtr to refer
to either std::auto_ptr or std::unique_ptr. This could be done
inside config.H perhaps?

#3 - 17 Jun 2016 22:24 - John Abbott

A quick check we should do is to string-replace all auto_ptr with unique_ptr and see if the code compiles cleanly (in --std=c++11 mode).

#4 - 21 Jun 2016 13:12 - John Abbott

Here are the files which use std::auto_ptr:
GlobalManager.H
MemPool.H
SparsePolyRing.H
SugarDegree.H
TmpJBMill.H
TmpPBMill.H
TmpUniversalInvolutiveBasisContainer.H

FractionField.C

FreeModule.C
MemPool.C
PPMonoidEv.C
PPMonoidEvOv.C
PPMonoidOv.C
PPmonoidSparse.C
QuotientRing.C
RingDenseUPolyClean.C
RingDistrMPolyClean.C
RingDistrMPolyInFpPP.C
RingDistrMPolyInlPP.C
RingFp.C
RingFpDouble.C
RingFpLog.C
RingFqLog.C
RingFpVec.C
RingQQ.C
RingTwinFloat.C
RingWeyl.C
RingZZ.C
SugarDegree.C
TmpHilbert.C
TmpJBMill.C
TmpPBMill.C
TmpToric.C ???
TmpUniversalInvolutiveBasisContainer.C

Also in src/server/ the following files:
CoCoAServer.C
GlobalIO.C
RegisterServerOps.C
RegisterServerOpsFrobby.C
RegisterServerOpsUser.C

#5 - 22 Jun 2016 13:24 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

I have just globally replaced auto_ptr with unique_ptr.
Just 1 line in TmpJBMill.C needed to be changed to get a clean compile (barring a couple of innocuous warnings).

I had to edit ex-PolyInput1.C and ex-PolyInput2.C as they used improper methods for testing whether an ifstream had been opened successfully.

For CoCoA-5 I had to edit OnlineHelp.C agains because of improper testing whether an ifstream had been opened successfully.

Now everything compiles, and all tests pass :-)

#6 - 22 Jun 2016 13:26 - John Abbott

Just a quick note about doing a global replacement of auto_ptr by unique_ptr. Open all files in emacs, then use M-x replace-regexp (rather than query-replace-regexp).

The replacement must be done for include/CoCoA/*.H, src/AlgebraicCore/*.C, src/server/*.C. That should be all.

#7 - 22 Jun 2016 16:15 - John Abbott

- Assignee set to John Abbott

- % Done changed from 10 to 20

I have looked on the internet for "clever" ways to make our code compatible with both C++03 and C++11 (using auto_ptr in the former case, and unique_ptr in the latter). However, I found no solution which I liked.

So I now think the best approach is just to leave the code as it is -- after all it does compile with the option --std=c++11 with the only downside being numerous "obsolescent" warnings.

When we finally relinquish C++03 compatibility (when??? soon?), then we can simply replace all auto_ptr by unique_ptr.

#8 - 26 Jun 2018 15:21 - John Abbott

- Related to Feature #82: C++11 compatibility questions added

#9 - 06 Sep 2018 16:54 - John Abbott

- Related to Design #1225: Move to C++14 (skipping C++11) added

#10 - 08 Feb 2019 21:33 - John Abbott

- Priority changed from Normal to High

- Target version changed from CoCoALib-1.0 to CoCoALib-0.99650 November 2019

#11 - 25 Mar 2019 15:45 - John Abbott

- Status changed from In Progress to Closed

- % Done changed from 20 to 100

- Estimated time set to 3.33 h

Summary:

- update effected
- **NOT C++03 COMPATIBLE** (not worth it)