CoCoALib - Design #1682

swap for new classes

31 May 2022 13:28 - John Abbott

Status:	In Progress	**Start date:**	31 May 2022
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	10%
Category:	Improving	**Estimated time:**	0.00 hour
Target version:	CoCoALib-0.99900	**Spent time:**	1.25 hour

Description

The description at cppreference strongly suggests that user-defined classes should be accompanied by their own **swap** functions: see https://en.cppreference.com/w/cpp/algorithm/swap

We should review all classes in CoCoALib in light of this.

This is really a matter of efficiency (rather than correctness).

Related issues:

Related to CoCoALib - Design #1685: RingBase::mySwap needed?	**In Progress**	**08 Jun 2022**

History

#1 - 31 May 2022 13:30 - John Abbott

I presented the template class **factorization** to my students, and asked whether it automatically had a "smart" swap capability.
Investigating led us to the page on the cppreference web-site.

I have not yet tested to see whether swapping factorization objects is costly (& exc safe) or not.

#2 - 15 Sep 2022 14:12 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

There is an efficient swap function for RingElem.

There is also a virtual mem fn RingBase::mySwap which is different from swap.
mySwap with its current interface has to be virtual because it receives as input
copies of the raw pointers to the values. Instead it should receive references
to the pointers, this in turn requires that the accessor functions myRawPtr return
references (which indeed they do). An alternative would be to access the data mem
myValuePtr directly (but that is poor encapsulation).

mySwap is called surprisingly often. Perhaps some of these can be changed to
calls to std::swap?

#3 - 15 Sep 2022 14:13 - John Abbott

- Related to Design #1685: RingBase::mySwap needed? added

#4 - 16 Feb 2024 09:35 - John Abbott

- Target version changed from CoCoALib-0.99850 to CoCoALib-0.99900