CoCoALib - Design #925

MachineInt or long for args which are indices (yet again)

20 Sep 2016 18:58 - John Abbott

Status:	In Progress	**Start date:**	20 Sep 2016
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	50%
Category:	Safety	**Estimated time:**	0.00 hour
Target version:	CoCoALib-0.99880	**Spent time:**	2.65 hours

Description

This matter has already been rasied in #89 and #124.
Much to my surprise in matrix.H there are several SetEntry fns which have MachineInt args.

Resolve the matter.

Related issues:

Related to CoCoALib - Bug #89: MachineInt or long as fn arg type for indices	**Closed**	**09 Feb 2012**
Related to CoCoALib - Feature #124: change long args in matrices into Machine...	**Rejected**	**04 Apr 2012**
Related to CoCoALib - Bug #830: Use MachineInt instead of long for params to ...	**Closed**	**01 Dec 2015**
Related to CoCoALib - Design #934: MachineInt: change semantics?	**In Progress**	**30 Sep 2016**
Related to CoCoALib - Design #581: C++14: MachineInt	**Closed**	**04 Jul 2014**

History

#1 - 20 Sep 2016 18:59 - John Abbott

- Related to Bug #89: MachineInt or long as fn arg type for indices added

#2 - 20 Sep 2016 18:59 - John Abbott

- Related to Feature #124: change long args in matrices into MachineInt (?) added

#3 - 20 Sep 2016 19:19 - John Abbott

- Priority changed from Normal to High

Evidently I am not fully decided in my own mind whether it better to use MachineInt for safety or long for simplicity and speed.

Today I'm inclined to the view that anyone who uses unsigned integral types gets what he deserves if things go wrong... perhaps not unlike people who use raw pointers?

Anyway, the current version of the code is at odds with the rejection of #124.
Rectify the situation!

#4 - 20 Sep 2016 19:20 - John Abbott

- Related to Bug #830: Use MachineInt instead of long for params to ZeroMat, IdentityMat, MatByCols, MatByRows added

#5 - 20 Sep 2016 19:22 - John Abbott

Now I see that issue #830 went ahead even though it is incompatible with #89 and #124.
Oops!

What to do?

#6 - 21 Sep 2016 12:51 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

After some reflection and after speaking to Anna...
I now think that MachineInt is wrong for indexes; the type MachineInt was originally conceived for avoiding problems with conversion from C++ integral types to RingElem.

The clean approach would be to invent a new type (say, IndexInt) for representing indexes, and presumably also for representing the size of indexable containers. Strictly there is a slight problem (with using IndexInt to specify the size) in that it is not possible to specify the very biggest indexable object since the largest such object is one greater than the largest representable index value (assuming we adopt the C++ approach of indexing from 0).

How confusing would it be to have both MachineInt and IndexInt (or whatever name we choose)?

#7 - 21 Sep 2016 18:15 - John Abbott

- Target version changed from CoCoALib-0.99550 spring 2017 to CoCoALib-0.99560

#8 - 30 Sep 2016 11:28 - John Abbott

- Related to Design #934: MachineInt: change semantics? added

#9 - 08 Nov 2017 16:52 - John Abbott

- Target version changed from CoCoALib-0.99560 to CoCoALib-0.99600

#10 - 19 Jan 2018 14:33 - John Abbott

Should indices be constrained to be non-negative?
Note that symbol currently does allow negative "indices".

It would be good to decide this issue before too long!

#11 - 12 Jun 2018 16:31 - John Abbott

- Target version changed from CoCoALib-0.99600 to CoCoALib-0.99650 November 2019

#12 - 05 Apr 2019 16:06 - John Abbott

- Related to Design #581: C++14: MachineInt added

#13 - 30 Apr 2019 17:18 - John Abbott

- Priority changed from High to Normal

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

Postponing because there are too many other more pressing issues. Anyway the current code works adequately, so it is not that harmful to postpone.

#14 - 08 Feb 2021 15:29 - John Abbott

- % Done changed from 10 to 50

I checked that NTL happily uses long for indices into matrices.
So it is reasonable for us to use long for "indices" (perhaps in a wider sense).

I have just updated the code for symbol, and it has become slightly simpler/cleaner.

#15 - 08 Feb 2021 15:47 - John Abbott

- Target version changed from CoCoALib-1.0 to CoCoALib-0.99850

#16 - 21 Jan 2024 20:22 - John Abbott

- Target version changed from CoCoALib-0.99850 to CoCoALib-0.99880

#17 - 16 Apr 2024 22:29 - John Abbott

Indices into matrices (and other indexable objects, *e.g.* ModuleElem?) must be non-negative.
As already noted indices for symbols may be negative -- and the implementation uses long.

Proposal: use long for indexes into matrices (& other indexable data-structures):

- **PRO** coherent with indexes for symbol
- **CON** technically limits max size (but not a problem in practice); not coherent with C++ indexable data-structures

Proposal: use unsigned long or size_t for indexes into matrices (& other indexable data-structures):

- **PRO** coherent with C++; *unimportantly* does not limit max size
- **CON** not coherent with symbol

Note that if the caller tries to supply a negative index when the parameter type is unsigned long then it will still be recognized as "out of range" --- and a heuristic could also indicate that it was probably negative, but do we really want to distinguish "negative index" from "out of range"? When would the distinction ever be useful?