CoCoALib - Feature #385

Design new errors using inheritance

08 Jul 2013 11:10 - John Abbott

Status:	In Progress	**Start date:**	08 Jul 2013
Priority:	Urgent	**Due date:**	
Assignee:	John Abbott	**% Done:**	40%
Category:	Tidying	**Estimated time:**	35.00 hours
Target version:	CoCoALib-0.99900	**Spent time:**	13.35 hours

Description

So that C++'s **try...catch** mechanism can be used easily we should redesign CoCoALib's error codes so that they are in an inheritance hierarchy.

Recall that an exception performs two tasks:

1. allow programmers to catch certain exceptions and handle them;
2. produce helpful(?!) error messages if not caught & handled

We have little experience with the first use.

Related issues:

Related to CoCoALib - Feature #92: Error Codes	**In Progress**	**14 Feb 2012**
Related to CoCoALib - Design #427: Error names and error messages (current de...	**In Progress**	**28 Jan 2014**
Related to CoCoALib - Design #582: Error codes: use same code for "not poly r...	**New**	**07 Jul 2014**
Related to CoCoALib - Feature #721: CheckForInterrupt: string arg to specify ...	**Closed**	**29 May 2015**
Related to CoCoALib - Feature #714: Interrupt mechanism	**Closed**	**19 May 2015**
Related to CoCoALib - Feature #743: Better errors: give supplementary info ab...	**In Progress**	**30 Jun 2015**
Related to CoCoA-5 - Feature #1045: Error message from cocoalib to cocoa-5	**Closed**	**13 Apr 2017**
Related to CoCoALib - Design #1063: Catching an (expected) error	**Closed**	**09 May 2017**
Related to CoCoA-5 - Bug #1095: TimeOut not working as hoped/expected	**Closed**	**29 Aug 2017**
Related to CoCoALib - Design #1098: Ctors for exceptions/errors	**New**	**06 Sep 2017**
Related to CoCoALib - Design #1462: Change CoCoA_ERROR into CoCoA_THROW_ERROR	**Closed**	**16 Jun 2020**
Related to CoCoALib - Design #1807: Error codes: "Not..." for "blah must be	**New**	**26 Mar 2024**

History

#1 - 08 Jul 2013 11:18 - John Abbott

The current error mechanism allows for **error messages in different languages** -- and the user can change language arbitrarily often at run-time. **Do we want to maintain this capability?** [bear in mind that the we do not really use the existing multi-language mechanism at all]

Perhaps I should investigate how hard it would be to produce a design that could easily be extended to a multi-language impl if we chose to do so....

NOTE (2013-07-09) if error mesgs are to be composed of a main "generic" message with an (optional) supplementary message, there will have to be a way of **translating the supplementary messages** into the chosen language -- this looks to be **problematic**!

#2 - 08 Jul 2013 12:10 - John Abbott

The following URL http://stackoverflow.com/questions/9387377/how-to-design-exception-types-in-c suggests deriving all user exception types from std::runtime_error. It also suggests **avoiding needlessly fancy inheritance structures** -- the inheritance structure serves only to allow the caller to easily catch/handle some errors and not others.

NOTE (2013-07-10) we may find it convenient to use C++ inheritance in our implementation but **without** this inheritance being **publicly guaranteed** (*i.e.* no user should rely on it).

#3 - 08 Jul 2013 14:24 - John Abbott

BOOST exceptions are quite sophisticated. Since exceptions currently play a very minor role in CoCoALib, **I am inclined to ignore BOOST exceptions** (at the moment, anyway). This will allow CoCoALib to remain (largely) independent of BOOST.

BOOST exceptions do offer one tempting feature: they allow callers to add new information to exception objects as the stack unwinds.

The link in my previous post suggests that C++11 may offer a mechanism for adding info to an exception during unwinding (which may make the BOOST offering obsolete?).

#4 - 10 Jul 2013 15:28 - John Abbott

- Assignee set to John Abbott

- % Done changed from 0 to 30

After a long discussion (yesterday) with Anna, we concluded that:

1. we are too ignorant about catching/handling exceptions to make a good design decision
2. it is very hard to subdivide errors/exceptions "unanimously" into classes
3. we could not think of any convincing scenarios to inspire us
4. the number of different errors must be greatly reduced (there are several "duplicates")

#5 - 11 Jul 2013 17:08 - Anna Maria Bigatti

The (work in progress) list of errors is now in the documentation (cvs will help us getting to the final design)
doc/txt/error.txt
(and checked it in)

#6 - 01 Apr 2014 17:35 - Anna Maria Bigatti

- Target version set to CoCoALib-0.99533 Easter14

#7 - 07 Apr 2014 14:44 - John Abbott

- Status changed from New to In Progress

- Target version changed from CoCoALib-0.99533 Easter14 to CoCoALib-1.0

#8 - 30 Apr 2015 07:38 - Anna Maria Bigatti

```
--> ERROR: NOT YET IMPLEMENTED -- please be patient, we're working on it
```

This is pretty useless for a CoCoA-5 user (and then for us, debugging) because by default CoCoA-5 does not print the whole error report.
It would be nice to throw it like
(a) CoCoA_ERROR(ERR::NYI + "multivariate polynomial", FunctionName);
or
(b) @CoCoA_ERROR(ERR::NYI ("multivariate polynomial"), FunctionName);@

#9 - 30 May 2015 10:22 - John Abbott

I think it may be a cleaner design to use (impose) the convention that all exceptions CoCoALib are derived from a single class. Most exceptions report "internal" problems, but issue #714 is about an exception type for responding to external interrupts.

Does a single super-class make good sense? It would allow a user to catch any possible exception thrown by CoCoALib in a single catch statement -- I'm not sure when this might be useful...

#10 - 26 Jun 2015 17:10 - John Abbott

- % Done changed from 30 to 40

JAA has now implemented a simple superclass CoCoA::exception from which all exceptions which CocoALib could throw must be derived.

Implemented, documented, (mildly) tested, checked in. The design "feels right".

#11 - 23 Mar 2016 17:42 - Anna Maria Bigatti

- Target version changed from CoCoALib-1.0 to CoCoALib-0.99550 spring 2017

#12 - 16 Sep 2016 16:10 - John Abbott

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

#13 - 13 Apr 2017 17:37 - Anna Maria Bigatti

- Related to Feature #1045: Error message from cocoalib to cocoa-5 added

#14 - 18 Apr 2017 22:54 - John Abbott

It seems that about 2 years ago I already had the idea that all exceptions originating from CoCoALib should be derived from a single class (called CoCoA::exception). Note that if a CoCoALib function calls a function from some other library, and that function throws an exception then CoCoALib does not convert the exception to one derived from CoCoA::exception because the exception did not originate from CoCoALib.

I have had another quick look for guidance about designing exception classes, and found one page which suggested keeping KISS in mind. This seems to be good advice. So we can start with a very simple set of exception classes, and on the basis of future experiments this could perhaps be later refined.

My current thoughts are the following:

- **CoCoA::exception** base class for all exceptions originating from CoCoALib
- **CoCoA::error** concrete class for exceptions corresponding to run-time errors
- **CoCoA::InsuffPrec** concrete class for twin-floats
- **CoCoA::TimedOut** thrown when a time-out occurs
- **CoCoA::InterruptReceived** concrete class thrown when an interrupt has been detected.

While it is very tempting to want to subdivide CoCoA::error into finer classes, it not at all clear to me how this could be helpful to a CoCoALib user.

If we do decide to subdivide CoCoA::error here are some possible subclasses:

- BadArg
- IncompatibleArgs
- ProgrammingError (*e.g.* AssertFail, ObsolescentFn, SERIOUS, NYI, iterator off end)
- other

#15 - 19 Apr 2017 08:13 - Anna Maria Bigatti

From another point of view (kind of bottom-up).
What we have now shows a few of these situations

```
    DEFINE_ERROR(BadPwrZero, "Non-positive power of zero");
    DEFINE_ERROR(NotPositive, "Value is not positive");
    DEFINE_ERROR(NotPositiveGrading, "Grading must be positive");
```

called, for example by NewLexOrdering. As it is already done in some functions, we could have just one "MustBePositive" and extra information (if necessary!) in the second argument:

```
      CoCoA_ERROR(ERR::NotPositive, "NewLexOrdering(n) size of matrix");
```

#16 - 19 Apr 2017 08:28 - Anna Maria Bigatti

There are some notes about "classes of errors" in the CoCoALib documentation for error, section "=== new improved list of errors ==="
(see also issue #92 for parallel discussion)

#17 - 19 Apr 2017 13:38 - John Abbott

Following on from my comment 14... well, assuming that we adopt that approach...

This means that practically all errors are represented by exceptions which are instances of a single class, CoCoA::error (or whatever name we choose). This makes the situation very similar to the current nonstandard errors, namely that essentially all the details are in the descriptive string(s).

I also agree with Anna's comments in #427: specifically

- errors containing double negatives are confusing (*e.g.* "not non-negative")
- we should try to use fairly uniform messages for similar errors.

#18 - 10 May 2017 14:12 - John Abbott

- Related to Design #1063: Catching an (expected) error added

#19 - 01 Sep 2017 11:28 - John Abbott

- Related to Bug #1095: TimeOut not working as hoped/expected added

#20 - 01 Sep 2017 11:31 - John Abbott

Here are some old notes about errors/exceptions I found while tidying up:

- BadArg
- IncompatibleArgs
- ProgrammingError (AssertFail, Obsolescent, ShouldNeverGetHere, NYI)

- InsuffPrec
- IntrRecd
- TimedOut

Recently I wrote some ad hoc code and wanted to use an **ERR::NeverGetHere** error; is this the same as **SERIOUS**?

#21 - 06 Sep 2017 15:27 - John Abbott

- Related to Design #1098: Ctors for exceptions/errors added

#22 - 06 Nov 2017 14:04 - John Abbott

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

#23 - 25 Jun 2018 15:34 - John Abbott

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

#24 - 05 Apr 2019 16:31 - John Abbott

- Priority changed from Normal to Urgent

- Target version changed from CoCoALib-0.99650 November 2019 to CoCoALib-0.99700

This really the same as #743.

#25 - 08 Jan 2020 22:57 - John Abbott

- Target version changed from CoCoALib-0.99700 to CoCoALib-0.99800

#26 - 16 Jun 2020 16:55 - John Abbott

- Related to Design #1462: Change CoCoA_ERROR into CoCoA_THROW_ERROR added

#27 - 06 Oct 2020 15:31 - John Abbott

- Target version changed from CoCoALib-0.99800 to CoCoALib-0.99850

#28 - 09 Feb 2024 10:25 - John Abbott

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

#29 - 26 Mar 2024 17:17 - John Abbott

- Related to Design #1807: Error codes: "Not..." for "blah must be ..." -- change prefix added