CoCoALib - Feature #92

Error Codes

14 Feb 2012 16:13 - Anna Maria Bigatti

Status:	In Progress	**Start date:**	14 Feb 2012
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:	Various	**Estimated time:**	40.00 hours
Target version:	CoCoALib-1.0	**Spent time:**	2.90 hours

Description

We've been talking a lot about how to classify errors and how many errors (error classes) we should have.
[see related issues below]

Here I have some cases:
DEFINE_ERROR(BadIndex, "Index out of range");
I think this should be called **IndexOutOfRange**

Another error I think would be useful in some occasions is "repeated index"

Related issues:

Related to CoCoALib - Design #308: Error: ERR::NotNonZero instead of ERR::Zer...	**In Progress**	**12 Feb 2013**
Related to CoCoALib - Feature #385: Design new errors using inheritance	**In Progress**	**08 Jul 2013**
Related to CoCoALib - Design #582: Error codes: use same code for "not poly r...	**New**	**07 Jul 2014**
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 #427: Error names and error messages (current de...	**In Progress**	**28 Jan 2014**
Related to CoCoALib - Design #1063: Catching an (expected) error	**Closed**	**09 May 2017**

History

#1 - 18 Oct 2012 15:15 - John Abbott

- Estimated time set to 40.00 h

The exception catching mechanism in C++ distinguishes the errors based on their types (respecting inheritance relationships). The current arrangement for errors in CoCoALib does not exploit inheritance at all, making it cumbersome for users who want to catch certain types of exception (they have to catch, do some comparisons, and possibly rethrow).

Redesign the CoCoALib error system to use inheritance; this implies organizing the errors into related groups. It may become less easy to add a new error (but this is probably a *feature* rather than a *bug* :-))

The hope is that we can also reduce the overall number of different exceptions CoCoALib throws.

#2 - 31 Oct 2012 14:17 - John Abbott

Here is note for accumulating ideas for structuring CoCoALib exceptions:
(1) create *CannotDivide* which has two subclasses: *InexactDivision* and *DivByZeroDivisor*
(2) create *IndexOutOfRange* with subclasses *BadRowIndex* and *BadColIndex*

#3 - 11 Jul 2013 14:40 - John Abbott

Note: our current thinking is to have only very few types of error, and allow each error object to contain a *supplementary message* giving more details about what went wrong. Here are a few examples:

- IndexOutOfRange with suppl mesg "matrix row index"
- UnsuitableValue with suppl mesg "value must be >= 0"
- BadType with suppl mesg "defined only for FractionFields"

Comparison of error objects will completely ignore the suppl mesg; so two IndexOutOfRange error objects with differing suppl mesgs will compare as equal. This is so that one can perform a check like if (ErrObj == IndexOutOfRange) ...; **WARNING** it may produce some unexpected results if error objects are used in std::map containers.

#4 - 11 Jul 2013 14:50 - John Abbott

What is a reasonable syntax for adjoining a suppl mesg to an error object?

(A) Anna suggests a fn-call style: **IndexOutOfRange("matrix row")**
(B) John suggests a binary op: **IndexOutOfRange + "matrix row"**
(B*) variant of proposal (B): **IndexOutOfRange += "matrix row"**
(C) BOOST uses a syntax like that for ostreams: **IndexOutOfRange << "matrix row"**

What happens if one tries to add several suppl mesgs?
(1) gives an error
(2) suppl mesgs are appended
(3) only the last suppl mesg is recorded
(4) only the first suppl mesg is recorded

JAA does not like option (1): getting an error while trying to prepare an error message is not ideal!! Option (2) is probably simplest to implement (and possibly the most natural) -- do we want to encourage/discourage appending multiple suppl mesgs? JAA thinks that the BOOST syntax encourages it.

NOTE (2013-09-11) if we want option (2) then JAA thinks syntax (B) or (B*) is more natural than syntax (A)

#5 - 01 Apr 2014 17:37 - Anna Maria Bigatti

- Target version set to CoCoALib-0.99533 Easter14

#6 - 04 Apr 2014 17:20 - John Abbott

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

#7 - 13 Apr 2017 17:17 - Anna Maria Bigatti

There are some notes about "classes of errors" in the CoCoALib documentation for error, section "=== new improved list of errors ==="

#8 - 13 Apr 2017 17:19 - Anna Maria Bigatti

- Related to Feature #743: Better errors: give supplementary info about the error added

#9 - 13 Apr 2017 17:29 - Anna Maria Bigatti

Anna Maria Bigatti wrote:

There are some notes about "classes of errors" in the CoCoALib documentation for error, section "=== new improved list of errors ==="

I suggest keeping the errors in error.C sorted as in that list (instead of the current alphabetical order): I believe that in this way, we would consider the problem of classifying the errors when we are thinking/looking for an error message, so when our brain is more likely to find a good strategy.

#10 - 13 Apr 2017 17:36 - Anna Maria Bigatti

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

#11 - 19 Apr 2017 08:31 - Anna Maria Bigatti

- Description updated

- Status changed from New to In Progress

#12 - 19 Apr 2017 08:32 - Anna Maria Bigatti

- Related to Design #427: Error names and error messages (current design) added

#13 - 09 May 2017 15:46 - Anna Maria Bigatti

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