

CoCoALib - Design #1063

Catching an (expected) error

09 May 2017 15:18 - Anna Maria Bigatti

Status:	Closed	Start date:	09 May 2017
Priority:	Normal	Due date:	
Assignee:	Anna Maria Bigatti	% Done:	100%
Category:	Documentation	Estimated time:	0.00 hour
Target version:	CoCoALib-0.99560	Spent time:	1.55 hour
Description			
Decide how to catch and expected error, for example a "Cannot reconstruct rational"			
Related issues:			
Related to CoCoA-5 - Bug #1062: IsRadical bug?		Closed	09 May 2017
Related to CoCoALib - Feature #92: Error Codes		In Progress	14 Feb 2012
Related to CoCoALib - Feature #385: Design new errors using inheritance		In Progress	08 Jul 2013

History

#1 - 09 May 2017 15:23 - Anna Maria Bigatti

- Category set to Documentation
- Target version set to CoCoALib-1.0
- % Done changed from 0 to 10

I had some trouble deciding **how** to catch an error:

I was expecting a possible problem in applying a partial hom, and I did not want to write explicitly the message string.

I wrote it this way, but I do not like it much.

```
catch (const CoCoA::ErrorInfo& e)
{
    if (message(e) == message(ErrorInfo(ERR::BadRingHomArg2, "")))
    {
        VERBOSE(80) << "    UGLY PRIME: going to next prime" << std::endl;
        continue;
    }
    throw;
}
```

Is there a better way than creating an ErrorInfo to compare the errors.

Maybe yes, but I didn't find it. I keep searching: if I find it I'll write it here ;-)

#2 - 09 May 2017 15:42 - Anna Maria Bigatti

- Status changed from New to Feedback

- Assignee set to Anna Maria Bigatti
- Target version changed from CoCoALib-1.0 to CoCoALib-0.99560
- % Done changed from 10 to 90

Found it (I had been confused by another unexpected error).
Easy: BTW I changed name to the error

```
catch (const CoCoA::ErrorInfo& err)
{
    if (err == ERR::BadPartialRingHomArg)
    {
        VERBOSE(80) << "    UGLY PRIME: going to next prime" << std::endl;
        continue;
    }
    throw;
}
```

#3 - 09 May 2017 15:44 - Anna Maria Bigatti

- Related to Bug #1062: IsRadical bug? added

#4 - 09 May 2017 15:46 - Anna Maria Bigatti

- Related to Feature #92: Error Codes added

#5 - 10 May 2017 14:12 - John Abbott

- Related to Feature #385: Design new errors using inheritance added

#6 - 11 May 2017 13:15 - John Abbott

I am a bit uneasy about using the following paradigm more than absolutely necessary:

```
try
{
    ans = SomeComputation(args);
}
catch (const CoCoA::ErrorInfo& err)
{
    if (err != ExpectedError) throw;
    // ignore expected error
}
```

We do use this approach for InsuffPrec with twin floats (since there is no reasonable alternative).
In the cases Anna has mentioned (here on redmine) in the last few days, I am less convinced that there is no reasonable alternative.

#7 - 08 Nov 2017 14:24 - John Abbott

- *Status changed from Feedback to Closed*

- *% Done changed from 90 to 100*