CoCoALib - Slug #969

Output to bad stream (operator<< and myOutput): just return immediately

10 Nov 2016 16:31 - John Abbott

Status:	Closed	**Start date:**	10 Nov 2016
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Improving	**Estimated time:**	4.40 hours
Target version:	CoCoALib-0.99700	**Spent time:**	4.00 hours

Description

Functions for outputting values should return immediately if the output stream is bad.
This is potentially important if the conversion to output form can be costly.

Related issues:

Related to CoCoALib - Feature #962: General verbose mode?	**Closed**	**07 Nov 2016**

History

#1 - 10 Nov 2016 16:31 - John Abbott

- Related to Feature #962: General verbose mode? added

#2 - 10 Nov 2016 16:38 - John Abbott

The motivation behind this issue is the idea of using an ostream with its bad-bit set to simulate /dev/null.

The current works correctly, but can "waste time" converting a value to output form when the output channel is bad. For example, I wrote a test program which created a table containing factorial(1) to factorial(9999), then I measure how long it took to "print" these values to an ostream with bad-bit set. The current impl took about 2.6s, but inserting an initial check whether the ostream is bad (and immediate exit if so) reduced the "printing" time to 0.000s.

It is probably "good practice" to perform the initial check anyway in most/all output fns. It is just 1 line:

```
  if (!out) return out;
```

#3 - 11 Nov 2016 15:10 - John Abbott

- Status changed from New to In Progress

- Assignee set to John Abbott

- % Done changed from 0 to 50

I have added the relevant line to all instances of operator<< (for ostream, but not for OpenMath output fns).

I am undecided whether an equivalent check should also be made in myOutput member fns.
Not adding the check to myOutput means that a direct call to the member fn (*i.e.* not passing through the corresponding operator<<) with a bad ostream will perform any "acrobatics" needed to convert the value to printable form, but then the characters of the printed form will simply be discarded. In summary: the harm is a just waste of computation time (if the check is not added).

Frankly, I expect it to be very rare that myOutput is called directly (without passing through some operator<<); so adding the checks is probably a waste of programming time (and possibly future maintenance time?)

I would almost be more tempted to add CoCoA_ASSERT(out); :-)

#4 - 26 Jan 2020 09:44 - John Abbott

- Status changed from In Progress to Feedback

- Target version changed from CoCoALib-1.0 to CoCoALib-0.99700

- % Done changed from 50 to 90

Here is a simple program I have used for testing (just printing of BigInt):

```
    const int N = 10000;
    vector<BigInt> F;
    F.push_back(BigInt(1));
    for (int i=1; i < N; ++i)
      F.push_back(i*F[i-1]);

    std::cout.setstate(std::ios_base::badbit);
    double t0 = CpuTime();
    for (int i=0; i < N; ++i)
      cout << F[i] << endl;
    double t1 = CpuTime();
    clog << t1-t0 << endl;
```

With the call to setstate the program takes less than 0.01s; if there call to setstate is removed then it takes about 2.4s.

NOTE: Removing setstate but using the shell to redirect output to /dev/null still takes about 2.4s (since that is the value-to-printed-form conversion time).

Note: not really relevant here, but there is a do-nothing ostream in BOOST: boost::iostreams::null_sink

#5 - 11 Feb 2020 18:07 - John Abbott

- Status changed from Feedback to Closed

- % Done changed from 90 to 100

- Estimated time set to 4.40 h

I have decided also to put "short-cuts" for bad ostreams inside the mem fns which produce output.
This is not really necessary for some very simple mem fns, but I do not think it is harmful, and it does remind the programmer who "copies" existing fns to do the same...

All tests pass; checked in. Closing.