CoCoALib - Design #1529

INPUT questions

31 Oct 2020 09:04 - John Abbott

Status:	Closed	**Start date:**	31 Oct 2020
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Safety	**Estimated time:**	3.11 hours
Target version:	CoCoALib-0.99800	**Spent time:**	3.10 hours

Description

I have some questions about what inputs should be accepted.

(1) when reading a BigRat we accept <Integer-literal><not-a-slash> and <integer-literal>/<integer-literal>. **Should we allow spaces around the slash?**
(2) if the istream is in oct or hex mode, do we read rationals in octal/hex? For instance, a/b is a valid hex rational.
(3) if the istream is in hex mode, and we want to read a symbol then a[a] is a valid symbol (meaning a[10] in decimal); also a is a valid symbol and a valid integer -- what to do?
(4) when reading an expression in hex mode x^a would mean x^10

I suppose the user should just be careful about setting a non-decimal mode...

Related issues:

Related to CoCoALib - Design #1523: Input fns: action when when istream is in...	**Closed**	**26 Oct 2020**
Related to CoCoALib - Feature #923: Fn to read symbol range (SymbolRange)	**New**	**16 Sep 2016**
Related to CoCoALib - Design #1538: RingElem from string (ReadExpr)	**Closed**	**13 Nov 2020**
Related to CoCoALib - Design #1547: Require decimal mode for ostream?	**Closed**	**05 Dec 2020**

History

#1 - 31 Oct 2020 09:05 - John Abbott

- Related to Design #1523: Input fns: action when when istream is in bad state? added

#2 - 31 Oct 2020 09:06 - John Abbott

- Related to Feature #923: Fn to read symbol range (SymbolRange) added

#3 - 04 Nov 2020 17:33 - John Abbott

One possibility would be to require that an integer literal start with a decimal digit: *e.g.* there is no problem in decimal or octal, but in hexadecimal one would write **0abc** instead of **abc**. If the number in hex already starts with 1-9 then it is not necessary to prepend a zero (but it is also harmless to do so).

I'm far from convinced that it is a good idea to use the istream (and ostream) base indicators; it is hard to imagine when it could be useful.
CoCoA could offer functions octal and hex (or hexadecimal) which convert an integer (and maybe a rational) into a string.... maybe?

Note that in C++ octal literals start with 0 and then have further octal digits; hex literals start with 0x and then have further hex digits.
So my suggestion above to start a hex literal with just 0 could be confusing to those who know C++... :-(

There is an output option in C++ **showbase** which forces output of the appropriate C++ standard prefix; not sure what it does for input (seemingly nothing).

#4 - 04 Nov 2020 17:41 - John Abbott

Regarding point **(1)**... what should input of a BigRat do with the following input streams?

- **1/2** -->JAA: ok
- **-1/2** -->JAA: ok
- **1/-2** -->JAA: err
- **-1/-2** -->JAA: err
- **- 1/2** -->JAA: ???
- **1 /2** -->JAA: ???
- **1/ 2** -->JAA: ???
- **1 / 2** -->JAA: ???
- **1/0** -->JAA: div-by-zero

If spaces are not allowed then - 1/2 will give error after the minus sign (and ready to read space-1-slash-2); 1 /2 will successfully read BigRat(1) and be ready to read /2;
1/ 2 will give error after the slash, and be ready to read 2 (space two); 1 / 2 is much like 1 /2.

Opinions?

#5 - 04 Nov 2020 20:06 - John Abbott

Ooops! I have had a look a what happens when a BigInt is read... and I found a bug :-(

For BigInt no space is allowed between a minus sign and the first digit.

Currently reading a BigInt not in decimal will produce a wrong result: the string of digits is correctly read, but BigIntFromString **assumes that the base is 10.**

#6 - 04 Nov 2020 20:25 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

The code for reading a BigRat can read it either as a fraction N/D or as a decimal Int.frac.

For the "decimal" form, no spaces are allowed around the dot.
The decimal form can be used for octal or hex input (so abc/def is a valid rational) -- not sure if this is really a good idea.

Since we don't allow spaces after minus sign for integer literals, and we don't allow spaces around the dot in "decimals", I am inclined to say that we don't allow spaces around slash (and the denom must be non-negative; zero denom throws div-by-zero).

#7 - 04 Nov 2020 20:41 - John Abbott

One mild concern is that the reading fns in C++ try to accept the maximum length initial string which gives "valid" input.

When reading a rational we could encounter an input stream containing 1/z. The max length initial string giving a valid rational is 1, but we can know this only after having read the slash and then seeing that the next char is not a valid digit; it is possible to "put back" the slash which has been read (and thereby mimic the behaviour of the input operators for built-in types).

However, we surely cannot mimic the behaviour of the input operators for built-in types when reading a symbol (see issue ???) because we might then need to put back an arbitrarily large number of characters (which is not portable).

Since we cannot do it for symbols, I am inclined to say that we should also not do it for rationals.

#8 - 13 Nov 2020 14:59 - John Abbott

- Related to Design #1538: RingElem from string (ReadExpr) added

#9 - 04 Dec 2020 11:29 - John Abbott

- Status changed from In Progress to Resolved

- Assignee set to John Abbott

- % Done changed from 10 to 70

Update:

- an exception is thrown if the istream is not in "decimal" mode
- no spaces are allowed inside a rational literal
- it is an error if there is a slash not followed by a positive denom
- no decimal part is required after a decimal point, but an integer part must be present

op>> returns the istream in a good state or throws an exception.

#10 - 05 Dec 2020 13:55 - John Abbott

- Related to Design #1547: Require decimal mode for ostream? added

#11 - 05 Dec 2020 14:02 - John Abbott

- Status changed from Resolved to Feedback

- % Done changed from 70 to 90

I have cleaned the code. All tests pass. Checking in.

#12 - 17 Feb 2021 11:02 - John Abbott

- Status changed from Feedback to Closed

- % Done changed from 90 to 100

- Estimated time set to 3.11 h

Closing after 2 months in feedback.