CoCoALib - Design #1606

Return type with const

03 Aug 2021 11:04 - John Abbott

Status:	Closed	**Start date:**	03 Aug 2021
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Improving	**Estimated time:**	1.66 hour
Target version:	CoCoALib-0.99850	**Spent time:**	1.70 hour

Description

While improving/correcting binomial (see issue #1605) I noticed that many functions have a "funny" return type, namely **const BigInt** where one might have expected just **BigInt**.

There is an interesting discussion at https://stackoverflow.com/questions/12051012/should-i-return-const-objects
where it is pointed out that returning const objects prevent use of move semantics (which require non-const objects).

We originally used const to block users from doing silly things such as (a+b)++ or even (a+b) = 1; **[assignment!]**
However, since const now prevent use of move-semantics, it could thus force the
compiler to make copies of values where one might expect/hope that no copy was needed.

I propose now to remove most/all of these const qualifiers on return types.

History

#1 - 03 Aug 2021 11:20 - John Abbott

- Description updated

#2 - 03 Aug 2021 14:08 - John Abbott

- Status changed from New to In Progress

- Assignee set to John Abbott

- % Done changed from 0 to 10

I have modified BigIntOps and everything compiles and runs fine.

#3 - 03 Aug 2021 21:26 - John Abbott

- % Done changed from 10 to 50

I have changed all files which obviously need changing (after grepping for a suitable regexp).
It compiles, and all tests pass. So I hope to check in shortly (after some more *ad hoc* testing).

#4 - 14 Sep 2021 12:07 - John Abbott

Not yet ready: the awkward cases mentioned in the description are compiled... ooops :-(

We may need to change the defn of operator+ and operator= so that the following do not compile:

```
BigInt a(3);
(a+a)++;
(a+a) = 2;
```

#5 - 04 Feb 2022 22:09 - John Abbott

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

#6 - 18 Mar 2024 09:51 - John Abbott

John will investigate this, and hopefully close it.

#7 - 18 Mar 2024 20:29 - John Abbott

- Status changed from In Progress to Feedback

- % Done changed from 50 to 90

This was mostly done anyway. I changed some things in the involutive code (without really understanding it).

I have left the postfix operator++ and operator-- as returning const results; it is hard to imagine when they might be used.

I have checked that input such as **if (a+b = 0)** does actually produce a compile-time error [though the message could be more helpful]
Something like **(a+b) = a;** does compile, but it is not so important to block it... when would anyone write such code?

Also the advice is not to put unnecessary const qualifiers so that std::move can be used (and work in the expected way).

#8 - 21 Mar 2024 20:24 - John Abbott

- Status changed from Feedback to Closed

- % Done changed from 90 to 100

- Estimated time set to 1.66 h