CoCoA-5 - Bug #1595

Bad input causes crash

13 May 2021 21:59 - John Abbott

Status:	Closed	**Start date:**	13 May 2021
Priority:	High	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	bug	**Estimated time:**	2.50 hours
Target version:	CoCoA-5.4.0	**Spent time:**	2.35 hours

Description

Even after my latest update CoCoA can still crash with bad input:

```
/**/ +-
--> ERROR: Using two unary operators in a row is not allowed; please check your expression and par
enthesize its inner sub-expression if this is really what you want
--> +-
--> ^^
[[waiting for semicolon]] /**/ +*
--> ERROR: Using two unary operators in a row is not allowed; please check your expression and par
enthesize its inner sub-expression if this is really what you want
--> +-
-->  ^
CoCoAInterpreter: /usr/local/include/boost/smart_ptr/intrusive_ptr.hpp:199: T* boost::intrusive_pt
r<T>::operator->() const [with T = const CoCoA::LexerNS::Line]: Assertion `px != 0' failed.
```

Related issues:

Related to CoCoA-5 - Bug #1594: Parser bug: missing close square bracket	**Closed**	**08 May 2021**

History

#1 - 13 May 2021 21:59 - John Abbott

- Related to Bug #1594: Parser bug: missing close square bracket added

#2 - 13 May 2021 22:00 - John Abbott

Separate issue because I think the cause is different; at least the error message seems to be different.

How tedious :-(

#3 - 14 May 2021 17:14 - John Abbott

```
+-
-+
```

```
- -
-x
```

#4 - 16 May 2021 10:47 - John Abbott

- Status changed from New to In Progress

- Priority changed from Normal to High

- % Done changed from 0 to 10

Here is an entertaining failing input:

```
-
-
-
```

CoCoA's response is:

```
/**/ -
[[waiting for semicolon]] /**/ -
--> ERROR: Using two unary minus operators in a row is not allowed because it looks suspiciously like as a mis
typed single-line comment; please parenthesize the inner expression or remove the blank(s) to make your intent
ions clear
--> -
--> ^
--> -
--> ^
[[waiting for semicolon]] /**/ -
--> ERROR: Using two unary minus operators in a row is not allowed because it looks suspiciously like as a mis
typed single-line comment; please parenthesize the inner expression or remove the blank(s) to make your intent
ions clear
--> -
--> ^
CoCoAInterpreter: /usr/local/include/boost/smart_ptr/intrusive_ptr.hpp:199: T* boost::intrusive_ptr<T>::operat
or->() const [with T = const CoCoA::LexerNS::Line]: Assertion `px != 0' failed.

Process cocoa5 aborted (core dumped)
```

The problem seems to be when a syntax error has been detected on one line, and then the next line triggers another syntax error... BOOM!

NOTE: however the amusing example does not crash with + instead of - (in all 3 cases).

#5 - 20 May 2021 17:02 - John Abbott

- % Done changed from 10 to 20

I have not found the real bug, but do have a workaround.

I have changed **Parser::checkThereIsntAnotherUnaryPlusOrMinus** (in Parser.C around lines 2163--2170)
Now it just indicates the second unary minus (or plus), rather than trying to underline both unary operators (and all intervening space).

I do not know why the previous version of the code was troublesome: somewhere in the loop in DefaultErrorReporter::outputUnderlinedChars
a null pointer is encountered. Source is in Lexer.C around lines 653--664.

#6 - 27 May 2021 14:28 - John Abbott

How should the error be reported?

In the fn Parser::checkThereIsntAnotherUnaryPlusOrMinus the error is reported by calling this->reportError,
but in many other functions errors are signalled by throwing an exception, *e.g.*

```
throw UnexpectedTokenException("Using two unary minus operators in a row is not allowed because it looks suspi
ciously like as a mistyped single-line comment; please parenthesize the inner expression or remove the blank(s
) to make your intentions clear", t);
```

What is the difference?
It could be that this->reportError allows a range of input to be specified (and this is what actually caused trouble).

Which method should we use here?

#7 - 27 May 2021 22:31 - John Abbott

- % Done changed from 20 to 30

I have just tried a slightly modified test, namely

```
-
-
-
3;
```

If the code calls reportError then the message about "2 unary minus" appears twice.
Instead if the code throws then the mesg appears just once.

#8 - 10 Jun 2021 21:37 - John Abbott

- Assignee set to John Abbott

- % Done changed from 30 to 50

JAA now thinks that explicitly throwing an exception is probably neater than calling reportError.
I'll revise the code, test it, and check in.

#9 - 21 Jun 2021 15:44 - John Abbott

- Status changed from In Progress to Closed

- % Done changed from 50 to 100

- Estimated time set to 2.50 h