CoCoA-5 - Design #473

Multiline string literals - useful or obsolescent?

13 Mar 2014 15:29 - John Abbott

Status:	Closed	**Start date:**	13 Mar 2014
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Parser/Interpreter	**Estimated time:**	5.00 hours
Target version:	CoCoA-5.1.0 Easter14	**Spent time:**	4.00 hours

Description

For the same reasons I do not like multiline comments, I have grave doubts about multiline string literals. They seem to be "useless" -- we never use them!

Related issues:

Related to CoCoA-5 - Bug #182: Unescaped double quote inside superstring	**Closed**	**08 Jun 2012**
Related to CoCoA-5 - Slug #875: Interpreter is too slow reading a big polynomial	**In Progress**	**03 May 2016**
Related to CoCoA-5 - Feature #1587: Multiline string literals (again)	**Closed**	**02 Apr 2021**

History

#1 - 13 Mar 2014 15:33 - John Abbott

We could define juxtaposition of normal string literals to mean concatenation with a newline inserted between the two strings. To my eye there is no significant difference in readability.

So instead of

```
str := """abc
def
ghi""";
```

we could get the same result writing

```
str := "abc" "def" "ghi";
```

NB the 3 literals abc def and ghi should be on separate lines, but there's an irritating bug in redmine which eats the whole post except for def

#2 - 13 Mar 2014 16:52 - John Abbott

My suggestion about juxtaposition of string literals is **not backward compatible** because juxtaposition is already defined to be simple concatenation (I didn't know that!!).

I think the current defn is not very useful since the + operator also effects simple concatenation. There *may* be a technical difference: it could be that the parser effects the concatenation of juxtaposed string literals whereas the + operator is surely effected by the interpreter.

NOTE 20140318 Anna tells me that I insisted on concatenation of juxtaposed string literals when we were designing CoCoA-5; it seems I've changed

my mind in the meantime! :-)

#3 - 18 Mar 2014 16:58 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

The source code relating to the recognition of string literals is in Lexer.C:423--511

```
Token Lexer::getStringLiteral(const CharPointer &begin, const ParserNS::ParserStatus &ps)
```

The fn above acquires the entire string literal (with appropriate checks for matching delimiters).

#4 - 18 Mar 2014 17:08 - John Abbott

Our options are:

1. remove all code related to multiline string literals
2. comment out all code related to multiline string literals
3. leave the code as is but remove all references from the C5 documentation
4. keep multiline strings as an integral part of C5

If I recall well, Anna insisted on multiline strings because that's how the on-line manual in C4 was implemented (and she wanted it to be easy to reflow text). However, the on-line manual in C5 is handled by C++ code, so there is no longer any special need for multiline strings.

My preference is for **(1)**; we could go there "softly" via **(2)**. I think **(3)** is slightly risky: someone who accidentally types three double-quotes will find C5 behaving in a strange(=undocumented) manner.

Option **(4)** is probably the easiest if the C5 doc has already been written; otherwise option **(3)** is easiest.

Can anyone justify wanting to keep multiline string literals?

#5 - 19 Mar 2014 12:34 - John Abbott

- % Done changed from 10 to 30

I have made a copy of Lexer::getStringLiteral which handles only normal string literals. It is 60 lines long compared to 88 lines for the version which handles multiline string literals as well. Note also that the parts which deal with multiline literals are quite involved (= convoluted = messy = hard to comprehend).

It seems very hard to justify the overhead of this extra "convoluted" code to handle a feature that has never yet proved useful!

Comments? Opinions? Ideas?

#6 - 19 Mar 2014 17:51 - John Abbott

- Status changed from In Progress to Resolved

- % Done changed from 30 to 70

I have just checked in the revised getStringLiteral (the old one is commented out).

Multiline strings were never documented, so no update is needed :-)

#7 - 02 Apr 2014 17:33 - Anna Maria Bigatti

- Target version set to CoCoA-5.1.0 Easter14

#8 - 04 Apr 2014 17:04 - John Abbott

- Target version changed from CoCoA-5.1.0 Easter14 to CoCoA-5.1.1 Seoul14

#9 - 06 May 2014 15:54 - John Abbott

- Status changed from Resolved to Closed

- Assignee set to John Abbott

- % Done changed from 70 to 100

- Estimated time set to 5.00 h

I have implemented option **(1)** *i.e.* eliminated all code to do with reading multiline string literals.

Documentation never mentioned multiline string literals.

The code has been working fine for over a month.
Closing.

#10 - 07 May 2014 17:32 - John Abbott

- Target version changed from CoCoA-5.1.1 Seoul14 to CoCoA-5.1.0 Easter14

#11 - 06 Jun 2016 23:52 - John Abbott

- Related to Slug #875: Interpreter is too slow reading a big polynomial added

#12 - 02 Apr 2021 10:59 - John Abbott

- Related to Feature #1587: Multiline string literals (again) added