CoCoA-5 - Feature #1587

Multiline string literals (again)

02 Apr 2021 10:59 - John Abbott

Status:	Closed	**Start date:**	02 Apr 2021
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Parser/Interpreter	**Estimated time:**	1.49 hour
Target version:	CoCoA-5.4.2	**Spent time:**	1.45 hour

Description

Now that we have more comprehensible "context-sensitive" prompts, one of the reasons for wishing to avoid multi-line string literals has largely been obviated.

A putative session could look like this:

```
/**/ str := "abc
[[in multi-line string]] /**/ def";
/**/ len(str);
7
```

Discuss whether we want to reintroduce multiline string literals.

Related issues:

Related to CoCoA-5 - Design #473: Multiline string literals - useful or obsol...	**Closed**	**13 Mar 2014**
Related to CoCoA-5 - Feature #1431: Juxtaposition of string literals	**Rejected**	**03 Mar 2020**
Related to CoCoA-5 - Bug #182: Unescaped double quote inside superstring	**Closed**	**08 Jun 2012**
Related to CoCoA-5 - Feature #1599: ConcatStr	**Closed**	**12 Jun 2021**

History

#1 - 02 Apr 2021 10:59 - John Abbott

- Related to Design #473: Multiline string literals - useful or obsolescent? added

#2 - 02 Apr 2021 10:59 - John Abbott

- Related to Feature #1431: Juxtaposition of string literals added

#3 - 02 Apr 2021 11:06 - John Abbott

If we do not allow multi-line string literals, but we do allow concatenation in the parser then the example in the description could look like:

```
/**/ str := "abc\n"
[[waiting for semicolon]] /**/ "def";
/**/ len(str);
7
```

Or if concatenation (in the parser) were denoted by ampersand then it would look like:

```
/**/ str := "abc\n" &
[[waiting for semicolon]] /**/ "def";
/**/ len(str);
7
```

Maybe the informative prompt could even say something like **[[waiting for string]]** since only a string literal may follow the ampersand... could be tedious to impl.

#4 - 02 Apr 2021 11:06 - John Abbott

- Related to Bug #182: Unescaped double quote inside superstring added

#5 - 02 Apr 2021 11:24 - John Abbott

A possible motivation for wanting to allow multi-line strings is...
CoCoA is slow at reading polynomials with many terms, but the function RingElem(P,str) is tolerably fast, so reading of large polynomials is best done via strings;
but editing files containing very long lines is often awkward (*e.g.* Emacs becomes very slow... not sure about vim), so it would be convenient to break the strings
into smaller pieces which are then (automatically?) reconcatenated.

Another thought: I wonder how the **sum** function works when given many strings to concatenate? Performance is disappointing:

```
N := 2000;
L := [sprint(factorial(N+i)) | i in 1..N];
t0 := CpuTime(); str := sum(L); TimeFrom(t0);
println len(str);
```

The above took about 5s (18Mbytes) on my computer, but increasing to N := 4000; took 120s (80Mbytes)

Maybe we need a ConcatStr function?

#6 - 05 Apr 2021 13:02 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

As mentioned in #1431#note-9 I have made a first impl of ConcatStr, and the result is promising.

Would it make sense to have a variant of ConcatStr which inserts newlines between successive strings?

```
// Whitespace on lines inside strings is not ignorable!
str := "abc
def
ghi";
```

```
str := ConcatStrNewline(["abc",
                         "def",
```

```
                          "ghi"]);
```

#7 - 05 Apr 2021 13:09 - John Abbott

My original thought (that multiline string literals are not so dangerous if we have helpful prompts) is not so helpful inside a CoCoA script.
They have the same potential to be confusing as multiline comments: one may have to look arbitrarily far back before knowing whether one is reading "active" source code, a comment, or a string literal. And obviously nasty things would happen if someone incautiously closed the string literal without meaning to.

In much the same way that I prefer single-line comments over multiline comments, I think I prefer ConcatStr (or ConcatStrNewline) over a multiline string literal.

Comments, thoughts?

PS: of course, another big advantage of not allowing multiline string literals is that I do not have to delve into the interpreter.

#8 - 05 Apr 2021 15:04 - John Abbott

If we do adopt the fn ConcatStr and/or ConcatStrNewline then maybe there should be a modified form of fold which produces output that can easily be made into input?

If we do make a modified form of fold, in view of KISS, I'd say that the first version does not do anything clever about escaping chars.

Here is an outline prototype in CoCoA-5 language **(proper version should be in C++)**: prints a long string as a list of strings

```
define NewFold(str, opt W)
  out := OpenOString();
  width := 60; // default value
  if IsDefined(W) then width := W; endif;
  print fold(str,W) on out;
  lines := close(out);
  input := OpenIString(lines);
  out := OpenOString();
  first := true;
  print "[" on out;
  while not(IsAtEOF(input)) do
    line := GetLine(input);
    if not(first) then print ",\n" on out; else  first := false; endif;
    print "\"", line, "\"" on out;
  endwhile;
  print "]" on out;
  return close(out);
enddefine; -- NewFold
```

PS might want to allow 2nd arg to say how long each substring is.

#9 - 12 Jun 2021 14:57 - John Abbott

- Related to Feature #1599: ConcatStr added

#10 - 21 Jun 2021 17:47 - John Abbott

- Assignee set to John Abbott

- Target version changed from CoCoA-5.4.2 to CoCoA-5.4.0

- % Done changed from 10 to 60

I am still inclined **against** multi-line string literals.
NewFold above ConcatStrings seem like a reasonable compromise.

I'm not convinced about the real utility of ConcatStrNewLine; if we find it would be useful later on, it will be easy to add it.

I did impl a quick version of "fold" which produced a list of strings, but it was disappointingly slow (no doubt because append makes useless copies).
If we can fix append, it may be worth having this version of "fold.

I'll put the new fold fn into a package: the name is cumbersome FoldToListInput.

#11 - 03 Feb 2022 19:33 - John Abbott

- Target version changed from CoCoA-5.4.0 to CoCoA-5.4.2

#12 - 14 Mar 2023 21:52 - John Abbott

- Status changed from In Progress to Closed

- % Done changed from 60 to 100

- Estimated time set to 1.49 h

fold and **ConcatStrings** are a reasonable compromise. We'll revisit the topic if a specific use-case comes up.
Closing.