CoCoA-5 - Feature #1431

Juxtaposition of string literals

03 Mar 2020 15:08 - John Abbott

Status:	Rejected	**Start date:**	03 Mar 2020
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	CoCoA-5 function: new	**Estimated time:**	1.99 hour
Target version:	CoCoA-5.4.0	**Spent time:**	1.90 hour

Description

We used to allow this, but then disallowed it in issue #576.

I raise it again because Martin Kreuzer wants to have some old CoCoA-4 packages transferred to CoCoA-5.
Those old packages tended to have a function called **Man** which printed out a very long, multi-line string.
This can be seen in the revitalized package thmproving.

Juxtaposing strings may help here. Discuss!

Reactivation appears to be a trivial matter of uncommenting the commented out code.

Related issues:

Related to CoCoA-5 - Design #576: Disallow juxtaposition for string literals?	**Closed**	**25 Jun 2014**
Related to CoCoA-5 - Feature #781: Option to "fold" long lines?	**Closed**	**28 Sep 2015**
Related to CoCoA-5 - Slug #1363: Emacs UI: slow with long lines	**In Progress**	**13 Nov 2019**
Related to CoCoA-5 - Feature #1587: Multiline string literals (again)	**Closed**	**02 Apr 2021**
Related to CoCoA-5 - Feature #1599: ConcatStr	**Closed**	**12 Jun 2021**

History

#1 - 03 Mar 2020 15:14 - John Abbott

The soln in thmproving.cpkg5 was to have a long succession of PrintLn commands:

```
PrintLn "DESCRIPTION";
PrintLn "";
PrintLn "Given a putative theorem in Euclidean geometry, we introduce Cartesian";
PrintLn "coordinates in the Euclidean plane, and translate the hypotheses and the...";
```

But in CoCoA-4 it was permitted to have newlines inside strings (more a curse than a mixed blessing).
With concatenation by juxtaposition, we could have something like:

```
PrintLn "DESCRIPTION\n"
        "\n"
        "Given a putative theorem in Euclidean geometry, we introduce Cartesian\n"
        "coordinates in the Euclidean plane, and translate the hypotheses and the...";
```

Is this a good idea?

I did also on one occasion actually want to have a long string literal... forgotten precisely why.
UPDATE I think I wanted to read in a large polynomial (*e.g.* 1Mbyte long), and it was inconvenient in Emacs to have such a long line (because Emacs becomes slow with long lines).

#2 - 03 Mar 2020 15:30 - John Abbott

- Related to Design #576: Disallow juxtaposition for string literals? added

#3 - 03 Mar 2020 17:29 - John Abbott

The relevant source code is around line 2100 in Parser.C, in the function Parser::parsePrimary.

#4 - 04 Mar 2020 15:33 - John Abbott

- Related to Feature #781: Option to "fold" long lines? added

#5 - 04 Mar 2020 21:15 - John Abbott

- Related to Slug #1363: Emacs UI: slow with long lines added

#6 - 19 Mar 2021 15:06 - John Abbott

Another possible justification for wanting to re-allow string literal juxtaposition is that we now have a fold function which can break a long string into pieces separated by newlines. The juxtaposition trick would make it easier to read in such a split string.

Why did I write in comment 1 that this is a "curse"? What problems does it cause?
I suppose a user could make the mistake to forget a comma after a string literal in a list of string literals then CoCoA will silently concatenate the literals.
Here is an example:

```
L := ["one" "two", "three"];
```

Or maybe string literal concatenation could be achieved by a special operator symbol? If so, what? (maybe **&**, ampersand)

Comments, ideas, suggestions?

#7 - 02 Apr 2021 10:59 - John Abbott

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

#8 - 02 Apr 2021 11:24 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

A potential disadvantage of something like **"abc"+"def"+"ghi"** is that a naive impl will take quadratic time. I suppose we could do geobuckets for string concatenation ;-)

The example in comment 6 with a "typo" (missing comma) does rather discourage allowing juxtaposition (but several other computer languages allow it, presumably without causing too much trouble for new programmers). Requiring an explicit operator would avoid that problem, but does involve delving to the parser... mmmm.

We could re-enable multi-line string literals... I'll create a new issue (#1587) to discuss this

#9 - 05 Apr 2021 12:53 - John Abbott

- Assignee set to John Abbott

I have just looked at the source code (Parser.C around lines 2135--2150); the old code for juxtaposed strings is still there. I do not really understand how it worked.
Presumably the interpreter knows how to "read" what the parser produced (see Interpreter.C around line 3250; search for stringliteral).

As suggested in #1587#note-5 I have implemented a new fn ConcatStr (not sure that that name is the best).
Repeating the example given in that note, but using ConcatStr instead of sum took about 0.12s; so about 1000 times faster.
That is acceptably fast.

An advantage of a fn such as ConcatStr is that it can be called on lists created at run-time, in contrast to a "smart" way of concatenating string literals.

#10 - 19 Apr 2021 13:55 - John Abbott

Another point in favour of ConcatStr is that it could insert newlines between the strings in a list, rather than requiring that the user put in \n explicitly at the end of each string.

```
println "First line\n"
        "second line\n"  "last line";

println ConcatStrWithNewlines(
    [ "First line",
      "second line", "last line"]);
```

#11 - 12 Jun 2021 15:04 - John Abbott

- Related to Feature #1599: ConcatStr added

#12 - 14 Sep 2021 11:45 - John Abbott

- Status changed from In Progress to Rejected

- % Done changed from 10 to 100

- Estimated time set to 1.99 h

Given the example in comment 6 (above), and the fact that **ConcatStrings** is now quite fast. We have decided to reject this issue.