Conversion Tools - Bug #6308

post string literal alternative coding quirk in 4GL preprocessor

04/29/2022 01:46 PM - Greg Shah

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

History

#1 - 04/29/2022 01:49 PM - Greg Shah

The function issue is actually a 4GL preprocessor quirk. I've created a testcase and checked it into Old UAST Testcases (see testcases/preproc/post_string_alternative_coding_quirk.p). The fix will need to be in ClearStream (search on the usage of getAlternativeCoding()). The only tricky part is that we must know when we are adjacent to and following a string literal. We do get a notification of being inside a string, but we need to check that the notification occurs at exactly the right moment AND we would have to set an extra flag followsString and clear that flag on the next character. Since this was a typo in the customer code where it was found, work on the "bug" was deferred.

The testcase:

```
// outside of comments and strings there is a quirk of the "alternative codings"
// - the ;) alternative coding construct will be converted by the preproc to the } character
// - this occurs in comments and in regular code
// - no conversion occurs inside strings (either '' or "" style)
// - the quirk is that any ;) in code that immediately follows a string literal (either '' or "" style) will N
OT convert but will emit as a single ) char
// - no whitespace or other chars can be between the string literal and the ;), the quirk only happens if it i
s directly following
// - this quirk does not manifest in comments
// - since alternative codings don't convert in strings, this quirk also doesn't appear in strings

def var txt1 as char.
def var txt2 as char.
def var txt3 as char.
def var i as int extent 2.

function func0 returns int(input i as char).
end.

// in each of the following comments, the ;) will be converted into the } character

/* "";) */
// "";)
/* ";) */
// ";)
/* '';) */
// '';)
/* ';) */
// ';)

// preprocesses to:
// func0("").
func0("";).

// preprocesses to:
// func0('').
func0('';).

function func-garbage returns int (input t1 as char, input-output t2 as char, input t3 as char):
   return 14.
end.
```

```
// preprocesses to:
// i[1] = 3.
i;<1;> = 3.

// preprocesses to:
// func-garbage("whatever", txt2, "whatever").
func-garbage("whatever", txt2, "whatever";).

// preprocesses to:
// func-garbage("whatever", txt2, 'whatever').
func-garbage("whatever", txt2, 'whatever';).

txt3 = ";)".
txt3 = ';)'.

// preprocesses to:
// txt1 = (txt2 + "stuff").
txt1 = (txt2 + "stuff";).

// preprocesses to:
// txt1 = (txt2 + 'stuff').
txt1 = (txt2 + 'stuff';).

// OO support would also be affected if the ;) occurs after a string, but it is not specific
// to function parms or OO parms
// def var basic as oo.basic.BasicMethods.
// basic = new oo.basic.BasicMethods().
// basic.something(9, txt2, "";).
```