

Re: Infinite Loops and Explicit Exits

Source: <http://coding.derkeiler.com/Archive/Cobol/comp.lang.cobol/2004-11/0049.html>

From: William M. Klein (wmklein_at_nospam.netcom.com)

Date: 11/02/04

Date: Mon, 01 Nov 2004 23:16:10 GMT

Response to first note in thread

As the person who wrote the initial proposal to add EXIT PARAGRAPH/SECTION syntax to the 2002 Standard (based on the JOD definition from the late 70's or early 80's),

I **like** the EXIT PERFORM <cycle> from within an out-of-line PERFORM for a reason that I don't think has been mentioned yet in the thread (but I haven't read it all yet).

Currently the EXIT PARAGRAPH/SECTION syntax can cause VERY unexpected results if you have code where PERFORM xxx-SECTION (still used more commonly in Europe than PERFORM xxx-para) and PERFORM xxx-para and GO TO statements. If a "generic" EXIT (out-of-line) PERFORM statement were added, this could be avoided and a "better" structured and maintainable code could be created.

Consider (for example)

Procedure Division.

PARA1.

 Perform Para2

 Go to Para2.

PARA2.

 Exit Paragraph.

Many more complex (real world) examples show how mixtures of the 3 types of control transfers can cause SERIOUS problems with the current EXIT statements.

--

Bill Klein

[wmklein <at> ix.netcom.com](mailto:wmklein@ix.netcom.com)

"Chuck Stevens" <charles.stevens@unisys.com> wrote in message
news:clu4lg\$1t2b\$1@si05.rsvl.unisys.com...

> For those who have been following the "perform forever" and "exit perform
> [cycle]" threads, I'm just tidying up the first drafts of two proposals that
> I hope to submit Real Soon Now to INCITS/J4 for evaluation and consideration
> for a future standard, if nothing else so that they can be added to the
> "candidates list" for future consideration.

comp.lang.cobol: Re: Infinite Loops and Explicit Exits

>
> One of these proposals relaxes the current restriction that an EXIT PERFORM
> or EXIT PERFORM CYCLE statement may only appear within a PERFORM ...
> END-PERFORM range.
>
> The other proposal is for PERFORM ... UNTIL FALSE. I didn't want to add a
> new reserved word and for that reason rejected PERFORM ... FOREVER. I was
> concerned about the potential semantic ambiguities that one might encounter
> in sequences like "PERFORM ... PERFORM ... UNTIL EXIT PERFORM ...
> END-PERFORM ...", which led me away from PERFORM ... UNTIL EXIT. I don't
> care for PERFORM ... WITH NO TEST because it really doesn't convey the sense
> of iteration. I am not wedded to the syntax I chose, but I think the
> technical reasons for avoiding the other three here presented are sound.
>
> Unless WG4 so directs, these capabilities can't be included in the proposed
> 2008 standard. But I can envision no technical barrier at all to the second
> of these, and the only barriers I can envision to the first are
> implementor-specific, WG4 *might* acquiesce so long as the additions do not
> cause a delay in the production of that next standard.
>
> At the very least this action on my part will ensure that these topics are
> on the standards committees' radar screens, and stimulate the discussion and
> clarification as to why EXIT PERFORM (with or without CYCLE) was limited to
> inline PERFORMs in the 2002 standard.
>
> -Chuck Stevens
>
>