

Re: Infinite Loops and Explicit Exits

Source: <http://coding.derkeiler.com/Archive/Cobol/comp.lang.cobol/2004-11/0262.html>

From: Richard (*riplin_at_Azonic.co.nz*)

Date: 11/06/04

Date: 6 Nov 2004 14:22:31 -0800

Robert Wagner <spamblocker-robert@wagner.net> wrote

> > *Of course that does stifle progression, but the company is there to*
> > *run the business, not write books on style.*
>
> *I disagree. Style can be refined in an EVOLUTIONARY (not*
> *revolutionary) way without deprecating what came before. In the world*
> *of cinema, such progression is called 'auteur theory'.*

""The auteur theory was, of course, not a theory of film at all, but a method of evaluation which enabled people knowledgeable about film, film history and particularly individual directors to show their knowledge and superiority over the uneducated masses who did not know what John Ford's cavalry trilogy was or the importance of cigarette lighting in the films of Howard Hawks.""

While both programming and movies are evolutionary, the old ones are not updated as part of a routine. Though 'The Jazz Singer' had audio added and 'Casablanca' was coloured, these were not common practice, remakes are usually failures. If movie companies spent their time 'updating' old movies then they would not have time to make new ones and would go broke.

In much the same way with Cobol systems. Cost is a factor, they can spend their time constantly revising old systems just to make them prettier with no added functionality or they can write new one to extend the function of IT into new areas.

> *If change is gradual, the same themes appear in old and new code.*
> *There isn't a scism or fracture.*

Only if the systems are continually worked on for other reasons. In many cases systems may continue to be used for many years without any need for change. You seem to suggest that they should be changed for no other reason than to change the code. This, of course, also introduces risk. Unnecessary risk.

- > *A weak but current example is*
- > *replacing GO TO THE-EXIT with EXIT PERFORM.*

- 1) The only compiler that has EXIT PERFORM in out-of-line in Chuck's
- 2) in-line EXIT PERFORM is not standard '85 and is only available in a couple of compilers.
- 3) Out-of-line EXIT PERFORM is not even '02 standard
- 4) EXIT PERFORM may, or may not, be equivalent to GO TO X-exit

- > >> *Cause: failure to use .. typedef.*
- > >
- > >*How can you possibly criticise them for not using a feature that is*
- > >*not available in their compilers ?*
- >
- > *Typedef is available on Micro Focus and Fujitsu.*

Yes, how many mainframes run the PC compilers that support this ?

- > >*Decisions and actions must always be made in different places, it*
- > >*can't all be in one line of code.*

- > *IF condition*
- > *action*
- > *END-IF*

You are implying that this cannot be:

```
IF condition
  PERFORM action
END-IF
or
IF condition
  CALL action
END-IF
```

- > >*Tax rates may depend on where the*
- > >*delivery is made to, you can't decide where the delivery is to be in*
- > >*the tax calc, nor would you calculate the tax in the address code.*
- >
- > *So you join delivery-address, taxable-sales-amount and tax-table in*
- > *one place that isn't subordinate to any of the three. In OO, it would*
- > *be a method of tax-amount.*

- > *I said "far away .. in the same or another program."*

Merely being in another program makes it 'far away'.

- > >> *Cause: unstructured programming.*
- >
- > *A well-written program should log and bypass cases it doesn't know how*
- > *to handle. They represent a 'bug' in the specs.*

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

Exactly. A bug in the specs has nothing to do with 'programming style' 'well written' or 'structured'. It may well be that the programmer should make a note to the systems analyst/designer and have code that reports cases not specified, but it does not relate in any way to 'unstructured'.