

Regarding EVALUATE TRUE

Source: <http://coding.derkeiler.com/Archive/Cobol/comp.lang.cobol/2007-08/msg00523.html>

- *From:* "klshafer@xxxxxxx" <klshafer@xxxxxxx>
 - *Date:* Wed, 15 Aug 2007 12:03:07 -0700
-

Usually I follow the Doc's prescription for whatever ails me in COBOL, as in...

'What is **this** stuff? EVALUATE TRUE WHEN cond-1 imperative statement...

you call this COBOL?!?'

'Oh, please, Mr Standards-and-Practises Reviewmeister, it is exactly what

is allowed by the ANSI '85 Standard.'

But allow me to put forth something about EVALUATE TRUE which causes me, uh, discomfort... Is it that I nearly swallowed too much of the Doc's medicine :-)? More likely, the prescription was just a lil' too hard for me to read... gotta upgrade those glasses from a 1.75 power to a 2.00 power, ya know...

Anyway, consider the following –

two distinct "indicator" data elements IND-SNOW and IND-SKY, each with multiple 88-levels, and a numeric TEMP.

Looking at –

```
EVALUATE TRUE
WHEN TEMP > 80
  PERFORM 100-SWEAT
WHEN SKY-IS-BLUE (an 88 on IND-SKY)
  PERFORM 200-SMILE
WHEN SNOW-IS-FALLING (an 88 on IND-SNOW)
  PERFORM 300-EXTEND-UMBRELLA
WHEN OTHER
  PERFORM 400-STAY-HOME
END-EVALUATE
```

Because there is an implicit exit from the EVALUATE after a when-imperative is executed, the above code has a coding-order-dependency for the WHEN's and their imperatives.

In other words, the above is NOT equivalent to –

```
EVALUATE TRUE
WHEN SKY-IS-BLUE (an 88 on IND-SKY)
  PERFORM 200-SMILE
```

Regarding EVALUATE TRUE

```
WHEN TEMP > 80
PERFORM 100-SWEAT
WHEN SNOW-IS-FALLING (an 88 on IND-SNOW)
PERFORM 300-EXTEND-UMBRELLA
WHEN OTHER
PERFORM 400-STAY-HOME
END-EVALUATE
```

Using the data-element version of EVALUATE accommodates a clean partitioning where the order does not matter...

```
EVALUATE TEMP
WHEN 0 THRU 20
PERFORM 100-SHIVER
WHEN 21 THRU 75
PERFORM 200-COMFORTABLE
WHEN 75 THRU 100
PERFORM 300-SWEAT
WHEN OTHER
PERFORM 999-CHECK-LIFE-SUPPORT
END-EVALUATE
```

The above is equivalent to –

```
EVALUATE TEMP
WHEN 21 THRU 75
PERFORM 200-COMFORTABLE
WHEN 0 THRU 20
PERFORM 100-SHIVER
WHEN 75 THRU 100
PERFORM 300-SWEAT
WHEN OTHER
PERFORM 999-CHECK-LIFE-SUPPORT
END-EVALUATE
```

Now I'm OK with EVALUATE TRUE if care is taken to construct a clean and exhaustive WHEN partitioning... like...

```
EVALUATE TRUE
WHEN TEMP < 21
PERFORM 100-SHIVER
WHEN (TEMP >20 AND TEMP < 76)
PERFORM 200-COMFORTABLE
WHEN (TEMP > 75 AND TEMP < 101)
PERFORM 300-SWEAT
WHEN OTHER
PERFORM 999-CHECK-LIFE-SUPPORT
END-EVALUATE
```

because there is no order dependence; the following is equivalent –

```
EVALUATE TRUE
WHEN (TEMP >20 AND TEMP < 76)
PERFORM 200-COMFORTABLE
```

Regarding EVALUATE TRUE

Regarding EVALUATE TRUE

```
WHEN TEMP < 21
PERFORM 100-SHIVER
WHEN (TEMP > 75 AND TEMP < 101)
PERFORM 300-SWEAT
WHEN OTHER
PERFORM 999-CHECK-LIFE-SUPPORT
END-EVALUATE
```

But consider this real-world example, where the conditions are 88-levels on DIFFERENT data elements...

```
EVALUATE TRUE
WHEN TRANSACTION-IGNORED
PERFORM I000-APPEND-MESSAGE-TEXT
WHEN NO-OUTPUT-TRAN
PERFORM I100-FORMAT-OUTPUT-TRAN
WHEN OTHER
CONTINUE
END-EVALUATE
```

This is not at all equivalent to –

```
EVALUATE TRUE
WHEN NO-OUTPUT-TRAN
PERFORM I100-FORMAT-OUTPUT-TRAN
WHEN TRANSACTION-IGNORED
PERFORM I000-APPEND-MESSAGE-TEXT
WHEN OTHER
CONTINUE
END-EVALUATE
```

If the original programmer's intention was that TRANSACTION-IGNORED is to "trump" whether or not we have an output-transaction already, I prefer to make the "exit" explicit, where the code is at the outer-level-most nesting, rather than making the "order precedence" in the EVALUATE do that –

```
IF TRANSACTION-IGNORED
PERFORM I000-APPEND-MESSAGE-TEXT
GO TO 1000-PARA-EXIT
END-IF
IF NO-OUTPUT-TRAN
PERFORM I100-FORMAT-OUTPUT-TRAN
END-IF
continue with output transaction processing...
```

To me, the above is a little more illustrative that the date-elements are, in fact, independent, and consequently their 88-levels are independent, and because the TRANSACTION-IGNORED appears first in the sequence, it trumps NO-OUTPUT-TRAN.

Yes, I know that the language definition of EVALUATE includes the order of the WHEN statements; but depending on the ordering in an

Regarding EVALUATE TRUE

EVALUATE just seems to me to be not in the spirit of what a "case" construct should be about.

All in all, I think that it is a very bad idea to use EVALUATE TRUE with WHEN condition-1 and WHEN condition-2 and WHEN condition-3 where condition-1, condition-2, and condition-3 refer to separate data-elements.

Ken

.