Subject: EXIT from any labeled construct
From: Van Snyder
Reference: 03-258r1, section 2.1.2, 04-109, 04-156

1 Number

TBD

2 Title

EXIT from any labeled construct.

3 Submitted By

J3

4 Status

For consideration.

5 Basic Functionality

Allow EXIT from any labeled construct.

6 Rationale

Some algorithms cannot be expressed in Fortran 2003 without GOTO statements or extra tests, but they could be expressed with EXIT if it could be applied to any labeled construct. For example, here's a routine that says "call R if X is not an element of the set S, which is represented by elements of `A(1:num_in_set)`." With an extra test:

```
   do i = 1, num_in_set
      if ( x == a(i) ) exit
   end do ! i
   if ( i <= num_in_set ) call r
```

or, with GOTO:

```
   do i = 1, num_in_set
      if ( x == a(i) ) go to 10
   end do ! i
   go to 20
10 call r
20 continue
```

or, with a more general EXIT:

```
o: if ( .true. ) then
      do i = 1, num_in_set
         if ( x == a(i) ) exit o
      end do ! i
      call r
   end if o
```

7 Estimated Impact

Trivial to minor — a few lines in 8.1.6.4.4. Estimated at meeting 169 to be at 3 on the JKR scale.

8 Detailed Specification

Allow an EXIT statement with a *do-construct-name* that is the same as the name of any enclosing construct, not just a DO construct. When the EXIT statement is executed, execution of the named construct is terminated. For compatibility, an EXIT statement without a *do-construct-name* continues to refer to the nearest enclosing DO construct.

8.1 Suggested editorial changes

These suggested editorial changes illustrate the magnitude of the project.

Replace *do-construct-name* in R844 with *construct-name*. Allow it to be the name of any construct that encloses the EXIT statement. Add a new subclause 8.1.7 that describes the EXIT statement but not loop termination. Specify there that the EXIT applies to the construct named by the *construct-name*. Do not change the interpretation of an EXIT statement that doesn't mention a *construct-name*.

It would be helpful if a construct existed that had no purpose other than to have a construct label.

Here's an example of a new 8.1.6.4.4

8.1.6.4.4 Loop termination

A loop terminates, and the DO construct becomes inactive, when any of the following occurs:

(1) Determination that the iteration count is zero or the *scalar-logical-expr* is false, when tested during step (1) of the above execution cycle,
(2) Execution of an EXIT statement that belongs to the DO construct,
(3) Execution of an EXIT statement or a CYCLE statement that is within the range of the DO construct, but that belongs to an outer construct,
(4) Transfer of control from a statement within the range of a DO construct to a statement that is neither the *end-do* nor within the range of the same DO construct,
(5) Execution of a RETURN statement within the range of the DO construct,
(6) Execution of a STOP statement anywhere in the program, or
(7) Termination of the program for any other reason.

When a DO construct becomes inactive, the DO variable, if any, of the DO construct retains its last defined value.

Here's an example of a new subclause about the EXIT statement:

8.1.7 EXIT statement

The EXIT statement provides one way of terminating a construct.

R844 *exit-stmt* **is** EXIT [*construct-name*]

C829 (R844) If an *exit-stmt* refers to a *construct-name*, it shall be within the range of that construct; otherwise, it shall be within the range of at least one *do-construct*.

An EXIT statement belongs to a particular construct. If the EXIT statement refers to a construct name, it belongs to that construct; otherwise, it belongs to the innermost DO construct in which it appears.

When an EXIT statement that belongs to a DO construct is executed, it terminates the loop (8.1.6.4.4). When an EXIT statement that belongs to a non-DO construct is executed, execution continues with the first executable construct after the END statement for the construct to which the EXIT statement belongs.

9 History