Subject: Embedding conditionals in expressions etc.
From: Van Snyder
Reference: 03-258r1, section 2.8.1; 04-192, 04-357r1

1   Number

2   TBD

2   Title

4   Embedding conditionals in expressions etc.

3   Submitted By

6   J3

4   Status

8   For consideration.

5   Basic Functionality

10   Allow to embed decisions within expressions, as actual arguments, or as pointer-assignment targets.
11   Allow to compute whether an actual argument is present or absent.

6   Rationale

13 One sometimes needs to select one thing or another to use within an expression or as a pointer-assignment
14 target. At present, for the former usage, one creates a temporary variable, sets that variable with an
15 if-then-else construct, and then evaluates the expression using the temporary variable. It would be more
16 convenient if one could embed the "use A or B depending on C" decision within an expression. These
17 may be such things as "IF (A $>$ 0, log(A), -huge(0.0))" or "IF (present(A), A(:,i), B)" where B is of
18 rank one. In these cases, the value of one or the other of the outcomes is desired, but it's important
19 not to calculate the "wrong" one because it will raise an error condition. In cases such as "p => IF (
20 associated(A), A(:,i), NULL())," it's important not to try to take the section A(:,i) if A is not associated.
21 Even if A is associated, the value of A(:,i) is not needed.

22 One would like to be able to compute whether an actual argument that is to be associated with an
23 optional dummy argument is to be considered to be present or absent. This isn't a compelling desire in
24 the case of one of these beasts, but for n of them, one needs a 2^n-way if-elseif...-else-endif construct with
25 a different one of the 2^n possible combinations of present actual arguments in each branch. It would be
26 more convenient if one could use an actual argument of the form "IF(A, B)" meaning "if A then B is
27 the actual argument else the actual argument is absent." In these cases, it's important that B, not the
28 value of B, is the actual argument — at least in the INTENT([IN]OUT) case. An example of this might
29 be "IF(present(A), A(:,i))" meaning the actual argument is A(:,i) if A is present and it is absent if A is
30 not present, or something similar with "present" replaced by "associated" or "allocated."

31 When a procedure is invoked, one would like to be able to select one actual argument or another
32 depending on some condition. For one argument, this isn't too bad: Put two different invocations in two
33 branches of an IF construct. For n arguments, the ways to do this are with a 2^n-way if-elseif...-else-endif
34 construct, or to have n IF statements or constructs that associate a pointer with one or another of the
35 desired arguments (or nullify it, if that was one of the outcomes), followed by the invocation. It would
36 be more convenient to be able to write something of the form "IF(A, B, C)" for each argument. In these
37 cases, it's important that either B or C, not the value of one or the other, becomes the actual argument
38 — at least in the INTENT([IN]OUT) case.

1 No matter whether a functional or operational syntax is chosen, the entity behaves somewhat differently
2 from existing functions or operators, in that the "result" *is* one of the arguments/operands, not the value
3 of one of them. That is, these entities behave more like run-time macro substitutions than functions or
4 operators.

7   Estimated Impact

6 Small, both for standard and implementors. Judged at meeting 169 to be at 4 on the JKR scale.

8   Detailed Specification

8 Provide two new intrinsic functions, named IF here but the particular names are not important. In both
9 cases, the first argument is of type logical, and is evaluated before the function is "invoked." In the
10 three-argument case, the result is the second argument if the first is true, and the third argument if the
11 first is false.

12 In the two-argument case, a reference to which is permitted only as an actual argument associated with
13 an optional dummy argument, the result is the second argument iff the first is true, else it is an absent
14 actual argument.

15 Notice that the specification carefully specifies "the result is ...," not "the result is the value of ...."
16 For all other functions, the result is an entity distinct from its arguments. For these functions, the result
17 *is* one of the arguments. The "functions" behave more like run-time macro substitutions than functions.

8.1   Illustrative edits w.r.t. 04-007, to indicate the scope of the proposed change

19 C1220$\frac{1}{2}$ (R1217) A reference to the two-argument form of the IF intrinsic function shall not appear   266:16+
20   except as an actual argument corresponding to an optional dummy argument.

21 [Replace "it" by "any function other than the IF intrinsic function (13.7.51$\frac{1}{2}$)".]   276:3

13.5.17$\frac{1}{2}$ Conditional functions   298:2+

23 IF (MASK, TSOURCE, FSOURCE)	Result is TSOURCE or FSOURCE, depending
24	on MASK.
25 IF (MASK, TSOURCE)	Result is TSOURCE if MASK is true, else result
26	is an absent actual argument.

13.7.51$\frac{1}{2}$ IF (MASK, TSOURCE, FSOURCE) or IF (MASK, TSOURCE)   322:23+

28 **Description.** Embed a decision within an expression, or calculate whether an actual argument
29 is present.

30 **Class.** Transformational.

31 **Arguments.**

32 MASK   shall be of type logical and shall be scalar.

33 TSOURCE   may be of any type, and may have any type parameter values. Shall be TKR
              compatible (5.1.1.2) with FSOURCE in the three-argument case. It is not
              evaluated before the function is invoked. It may be undefined. If it is a
              pointer it need not be associated. If it is allocatable it need not be allocated.

34 FSOURCE   shall be TKR compatible with TSOURCE. It shall be polymorphic if and
              only if TSOURCE is polymorphic. It is not evaluated before the function is
              invoked. It may be undefined. If it is a pointer it need not be associated. If
              it is allocatable it need not be allocated.

35 **Result Characteristics.**

36 *Case (i):*   Three arguments: The result characteristics are the same as TSOURCE if MASK
37   is true, else the same as FSOURCE.

38 *Case (ii):*   Two arguments: The result characteristics are the same as TSOURCE if MASK
39   is true, else the result is an absent actual argument.

Result.

Case (i):   Three arguments: The result is the TSOURCE argument if the MASK argument is true, else it is the FSOURCE argument. The result, and therefore the function reference, may appear in a variable-definition context (16.5.7) if TSOURCE and FSOURCE are permitted to appear in a variable-definition context.

Case (ii):   Two arguments: The result is the TSOURCE argument if and only if the MASK argument is true. If MASK is false the result is undefined, and the actual argument consisting of the function reference is absent. The result, and therefore the function reference, may be associated with an argument that does not have INTENT(IN) if TSOURCE is permitted to appear in a variable-definition context (16.5.7).

Examples.

Case (i):   The result of IF (PRESENT(X), X, 0.0) is X if X is present, else it is 0.0.

Case (ii):   The result of IF (ASSOCIATED(P), P(::2), NULL()) is the array section P(::2), which is not a pointer, if P is associated, and NULL(), which is a pointer, if P is not associated. Both are valid targets in a pointer assignment.

Case (iii):   The result of IF (ASSOCIATED(P), P(::2)) is a present actual argument that is the array section P(::2) if P is an associated pointer, else it is an absent actual argument.

Case (iv):   The result of IF (PRESENT(D), D(:,J)) is a present actual argument consisting of the array section D(:,J) if D is a present dummy argument, else it is an absent actual argument.

9   History

Concept originally submitted in 04-192 at J3 meeting 167.