Subject: Generalization of vector-valued subscript
From: Van Snyder
Reference: 03-258r1, section 2.9.1

Number

TBD

Title

Generalization of vector-valued subscript.

Submitted By

J3

Status

For consideration.

Basic Functionality

Allow a single subscript of rank $k+1$ and extents $(r, n_1, ..., n_k)$ as a subscript for a rank-r array A.

Rationale

There are computations that result in collections of r subscripts for rank-r arrays. These cannot be used as subscripts in the same way that vector-valued subscripts can be.

If the specification for the shape of things that can be used for subscripts were generalized, this would be possible.

Estimated Impact

Minor.

Detailed Specification

Allow a single subscript of rank $k+1$ and extents $(r, n_1, ..., n_k)$ as a subscript for a rank-r array A. The elements of the rank-one sections in the first dimension of S are used consecutively as subscripts for A, resulting in a rank k array of extents $n_1, ..., n_k$. This provides a more general scatter/gather facility than the present vector subscript facility. This is not the same as using the elements of the rank-one sections in S as vector subscripts for A, which would result in a rectangular section of shape $(n, n, ..., n)$ (in the case S is of rank 2).

Example:

Suppose we have arrays A3 with dimensions (10,10,10) and S3 with dimensions (3,2). If we assume `S3 = reshape( (/3, 4, 5, 6, 7, 8/), (/3,2/) )` $= \begin{bmatrix} 3 & 6 \\ 4 & 7 \\ 5 & 8 \end{bmatrix}$, then `A3(S3)` is a rank-1 extent (2) array that can appear in a variable-definition context (except perhaps not as an actual argument associated with a dummy argument having INTENT(OUT) or INTENT(INOUT)); it specifies the same array as `(/ A3(3,4,5), A3(6,7,8) /)`, which cannot appear in a variable-definition context. This is different from A3(S3(1,:),S3(2,:),S3(3,:)), which can appear in a variable-definition context, but is an object with extents (2,2,2), not (2). The former is an arbitrary collection of elements, while the latter is a rectangular section.

As a degenerate case, allow a single subscript of rank one and extent equal to the rank of an array as the only subscript for that array. The elements of the first array are treated as the subscripts of the second array, resulting in accessing a single element of the second array. The result is a scalar, not an array of extent (1), or an array of extent (1,1,...,1).

Example:

Suppose we have arrays A3 with dimensions (10,10,10) and S3 with dimension (3). If we assume `S3 = (/ 3, 4, 5 /)`, then `A3(S3)` is the same as `A3(3,4,5)`, not `A3(3:3,4:4,5:5)`.

History