Subject: Map application of function onto derived-type components
From: Van Snyder

1 Number

TBD

2 Title

Map application of function onto derived-type components.

3 Submitted By

J3

4 Status

For consideration.

5 Basic Functionality

Map application of function onto derived-type components.

6 Rationale

It is occasionally useful to apply a function to all of the components of an object of derived type, or to all corresponding componenents of several such objects.

7 Estimated Impact

Minor — tending toward the large end thereof depending on how much of what's described here is done — and well concentrated. The most important one is the intrinsic function.

8 Detailed Specification

8.1 A MAP intrinsic "function"

13.7.69$\frac{1}{2}$ MAP (F, A1 [, A2, ...])

Description. Apply a function to all of the components of an object of derived type, or to corresponding componens of several such objects.

Class. Transformational function.

Arguments.

F shall be the name of a specific function with explicit interface, a generic function, an operator symbol, or a reference to a function that returns a function pointer with explicit interface. It may be elemental, pure or impure.

A1 [, ...] may be of any type. At least one of the arguments A1, ... shall be of derived type. If more than one is of derived type, those that are of derived type shall all be of the same type and rank, and corresponding kind type parameters and length type parameters shall have the same values.

Result Characteristics. The type, type parameter values and rank are those of the arguments that are of derived type.

Result Value. The value of each component of the result is the value that results from applying F to the corresponding components of those arguments that are of derived type, and to the other arguments.

Examples. Consider the type POINT defined in Note 4.54.

Case (i): The result of MAP(+, POINT(1.0, 2.0), POINT(3.0, 4.0)) is POINT(4.0, 6.0).

Case (ii): The resulf of MAP(*, 5.0, POINT(1.0,2.0)) is POINT(5.0,10.0). This illustrates that not all of A1, A2, ... need be of derived type.

Case (iii): The result of MAP(F, T(5.0, 7, ”Name”, .FALSE.)) is T(F(5.0), F(7), F(”Name”), F(.FALSE.)). This illustrates that F could be generic.

8.2 New thing-o to put in an interface block

When one develops a data structure, one develops not only the derived type to represent objects of the data structure, but also operations on that type. One way that operations are developed is to write functions that perform them. If the operations can be implemented by applying a function or operation, perhaps a generic function, to every element of objects of the type, or to corresponding components of several objects, one nonetheless needs to write a function to carry out those applications. The MAP function would help, but it would be more convenient not to need to spell it out explicitly every time. This could be avoided by a specification that implies the mapping is applied. Here’s a proposal based on interface blocks.

Allow a MAP statement of the following form in an interface block, perhaps a generic one, including one that defines an operation:

Add the following to R1202:

R1202	*interface-specification*	**is**	*map-stmt*
R1206$\frac{1}{3}$	*map-stmt*	**is**	MAP (*map-spec*, *type-spec-list*)
R1206$\frac{2}{3}$	*map-spec*	**is**	*defined-operator*
		or	*function-name*
		or	*generic-name*

C1200$\frac{1}{3}$ (R1206$\frac{1}{3}$) At least one *type-spec* shall specify a derived type. If more than one specifies a derived type, all that specify a derived type shall specify the same derived type, with the same values of corresponding kind type parameters.

C1200$\frac{2}{3}$ (R1206$\frac{2}{3}$) If *map-spec* is *generic-name*, every specific interface of that generic interface shall be a function.

Specification of functionality would be similar to 13.7.69$\frac{1}{2}$. It may be necessary to prevent recursive reference to the *generic-spec*.

8.3 New binding to a derived type

Add the following to R450

R450	*proc-binding-stmt*	**is**	*map-binding*
R452$\frac{1}{4}$	*map-binding*	**is**	MAP (*binding-map-spec*, *type-spec-list*)
R452$\frac{2}{4}$	*binding-map-spec*	**is**	*map-spec*
		or	*binding-name*
R452$\frac{3}{4}$	*map-spec*	**is**	*defined-operator*
		or	*function-name*
		or	*generic-name*

C464$\frac{1}{3}$ (R452$\frac{1}{4}$) At least one *type-spec* shall specify the type being defined. All *type-spec*s that specify a derived type shall specify the type being defined.

C464$\frac{2}{3}$ (R452$\frac{3}{4}$) If *map-spec* is *generic-name*, every specific interface of that generic interface shall be a function.

There should only be one definition of *map-spec* if both 8.2 and 8.3 are done.

Specification of functionality would be similar to 13.7.69$\frac{1}{2}$. If should be allowed for a *generic-spec* to be a stand-alone one or one bound to the same type.

9 History