Subject: Non-null initial targets for pointers
From: Van Snyder
References: 03-258r1, section 2.12.2, 04-202, 04-351

1 Number

TBD

2 Title

Non-null initial targets for pointers.

3 Submitted By

J3

4 Status

For consideration.

5 Basic Functionality

Allow initial targets for pointers that are other than NULL().

6 Rationale

It would be useful to be able to initialize pointers to targets other than NULL(). This is especially true for procedure pointers.

7 Estimated Impact

Minor. It was estimated to be at 3 on the JKR scale at meeting 169, but it is now clear that the project is a bit more complicated than originally envisioned. It might now fairly be judged to be at 4 on the JKR scale.

8 Detailed Specification

Allow the initial target for a data pointer to be an accessible nonpointer nonallocatable variable with the SAVE attribute. Every expression within the variable shall be an initialization expression.

Allow the initial target for a procedure pointer to be an accessible external or module procedure, or an intrinsic procedure listed in subclause 13.6 and not marked with a bullet (•) (or the result of resolving a generic without invoking a procedure if the proposal in 04-391 succeeds).

The initial target shall satisfy all the requirements for pointer assignment (e.g. the TARGET attribute, type conformance, etc.).

This feature shall be available both for named pointers and for pointer components. Pointer components may be default initialized to have an initial target.

The target may be accessed by use or host association. If it is declared in the same scoping unit it need not have been previously declared; this facilitates initialization to a "sentinel" object. (See note $4.36\frac{1}{2}$ in section 8.1 below.)

If the proposal to provide for reinitialization on every invocation is adopted, extend this proposal mutatis mutandis. It will be necessary to take care to avoid circular dependence of on-every-invocation pointer reinitialization if a pointer's on-every-invocation initial target is allowed to be the target of another pointer.

8.1 Suggested edits

The following edits address only the first part of the proposal — The part not dealing with initialization on every invocation. They are intended to illustrate the magnitude of the project, but they may be complete or nearly so.

---

or *data-pointer-init* 50:14

[Editor: "*null-init* appears for a" ⇒ "*data-pointer-init* consisting of *null-init* appears for a data pointer component or *proc-pointer-init* consisting of *null-init* appears in *proc-decl* for a procedure".] 53:6

If *data-pointer-init* consisting of *variable* appears for a data pointer component, that component in any object of the type is initially associated with *variable* or becomes associated with *variable* as specified in 16.4.2.1.1. 53:7+ New ¶'s

If *proc-pointer-init* consisting of *procedure-name* appears in *proc-decl* for a procedure pointer component, that component in any object of the type is initially associated with *procedure-name* or becomes associated with *procedure-name* as specified in 16.4.2.1.1. The component and *procedure-name* shall be related in the same way as required for *proc-pointer-object* and *proc-target* in 7.4.2.2.

[This note illustrates that we should not require the non-null initial target of a pointer component to be previously declared.] 54:1-

NOTE 4.36$\frac{1}{2}$

A pointer component of a derived type may have an initially non-null target, so long as that target is accessible, has the SAVE attribute, does not have the POINTER or ALLOCATABLE attribute, has no expressions that are not initialization expressions (such as a variable subscript), and would be permitted as a target in a pointer assignment.

```
TYPE NODE
  INTEGER               :: VALUE = 0
  TYPE (NODE), POINTER :: NEXT_NODE => SENTINEL
END TYPE

TYPE(NODE), SAVE, TARGET :: SENTINEL
```

---

or *data-pointer-init* 72:16

R506$\frac{1}{2}$ *data-pointer-init* **is** *null-init*
or *variable*

[Editor: "with no arguments" ⇒ "that does not have an argument with a type parameter that is assumed or is defined by an expression that is not an initialization expression (7.1.7)". This tiny generalization makes the definition of "initialization expression" easier in the case of a structure constructor with a pointer component. Otherwise, C525$\frac{1}{2}$ would have to be repeated there.] 72:18-19

C525$\frac{1}{2}$ (R506$\frac{1}{2}$) The *variable* shall be declared in the same scoping unit or be accessible therein by use or host association, shall have the SAVE attribute or be declared in the main program, shall not have the POINTER or ALLOCATABLE attribute, every expression within it shall be an initialization expression, and it shall satisfy the requirements for a *data-target* in a pointer assignment statement (7.4.2) in which the *data-pointer-object* is the corresponding data entity. 73:16+

[This intentionally doesn't say "previously declared". That would prohibit initializing a "next" pointer component to a "sentinel" object.]

If *initialization* is => *null-init*, the initial association status of *object-name* is disassociated. If *initialization* is => *variable*, *object-name* is initially associated with the variable. 74:33-34

[An *object-name* is already required to be a pointer by C525. The first sentence of this edit may be useful even if this proposal does not proceed.]

---

or *data-pointer-init* 88:26

[Editor: "*null-init*" ⇒ "initial association status".] 89:12

[Editor: "*null-init*" ⇒ "*data-pointer-init*" twice.] 89:14,16

[Editor: "The" ⇒ "If *data-pointer-init* is *null-init*, the". "pointer *data-stmt-object*" ⇒ "data statement object" because the data statement object is already required to be a pointer and syntax terms don't have association status. Insert a new sentence at the end of the paragraph:] 89:15

If *data-pointer-init* is *variable* the corresponding data statement object is initially associated with the variable.

---

(3) A structure constructor where each *component-spec* corresponding to 126:27-29

(a) An allocatable component is a *null-init*,

(b) A data pointer component is a *data-pointer-init*,

(c) A procedure pointer component is a *proc-pointer-init*,

(d) Any other component is an initialization expression,

---

(6) A *null-init*, 127:4-6

R1214 *proc-decl* **is** *procedure-entity-name* [=> *proc-pointer-init*] 264:19

R1214$\frac{1}{2}$ *proc-pointer-init* **is** *null-init*
or *procedure-name*

C1216$\frac{1}{2}$ (R1214$\frac{1}{2}$) A *procedure-name* shall be the name of an accessible external or module procedure, or the name of a specific intrinsic function listed in 13.6 and not marked with a bullet (•). 264:30+

If => appears in a *proc-decl* in a *procedure-declaration-stmt* it specifies that the initial association status of the corresponding procedure entity is defined, and implies the SAVE attribute. The SAVE attribute may be reaffirmed by explicit use of the SAVE attribute in the *procedure-declaration-stmt*, by inclusion of the procedure entity name in a SAVE statement (5.2.12), or by the appearance of a SAVE statement without a *saved-entity-list* in the same scoping unit. If => *null-decl* appears, the procedure entity is initially disassociated. If => *procedure-name* appears, the procedure entity is initially associated with the procedure specified by *procedure-name*. The characteristics of *procedure-entity-name* and *procedure-name* shall be related in the same way as required for *proc-pointer-object* and *proc-target* in 7.4.2.2. 265:15-18

---

(3) The pointer is an ultimate component of an object of a type for which default initialization consisting of *variable* or *procedure-name* is specified for the component and
[Editor: Copy [414:26-30] to here.] 414:18+

[Editor: Insert "consisting of *null-init*" before "is".] 414:25

9 History