Subject: FINAL procedures as type-bound procedures
From: Van Snyder
References: 99-108, 00-138 00-170

1 Introduction

This paper is based on 00-138, which was available but not discussed at meeting 152. The syntax is slightly different from what was adopted for final procedures in 99-108, and slightly different from what was proposed in 00-138. There is more work to be done for final procedures, especially specifying the order in which objects cease to exist, and therefore the order in which their final procedures are executed.

2 Edits for finalization

Edits refer to 00-007. Page and line numbers are displayed in the margin. Absent other instructions, a page and line number or line number range implies all of the indicated text is to be replaced by immediately following text, while a page and line number followed by + indicates that immediately following text is to be inserted after the indicated line. Remarks for the editor are noted in the margin, or appear between [and] in the text.

---

or PROCEDURE :: FINAL() => *procedure-name-list* 44:18+

44:25+

Constraint: The *procedure-name* shall be the name of an accessible module subroutine. It shall have one dummy argument with a declared type of *type-name* that is polymorphic if and only if *type-name* is extensible. This argument shall not have the ALLOCATABLE, ASYNCHRONOUS, OPTIONAL, INTENT(OUT), POINTER, VALUE or VOLATILE attribute. If the dummy argument is an array it shall have assumed shape. All nonkind parameters of the dummy argument shall be assumed.

Constraint: If several subroutines are bound to the type with *binding-attr* FINAL, they shall be distinguished according to the rules for unambiguous procedure references (14.1.2.3).

Constraint: If any final subroutines are specified for a type and the dummy arguments have a particular set of kind type parameters, at least one of them shall have a scalar dummy argument.

4.5.1.5.1 Final subroutine 49:30+

A procedure binding that specifies FINAL() is a **final subroutine** for objects of the type. The set of final subroutines that are bound to the type is a generic interface.

A final subroutine may be elemental.

When any object is deallocated (6.3.3, 6.3.3.1) or becomes undefined by the events specified by items (3) or (13)(c) in 14.7.6, if a final subroutine is selected as specified in 14.1.2.4.2$\frac{2}{3}$, it is invoked with the object as its actual argument. If the subroutine causes other objects of the same type and kind type parameters to be deallocated or to become undefined by the events specified by items (3) or (13)(c) in 14.7.6, it shall be recursive.

J3 internal note

Unresolved issue xxx
A specification when certain objects – e.g. variables with the SAVE attribute, module variables, ... – cease to exist is needed.

Immediately following execution of a final subroutine, the direct subobjects of the type, including the parent subobject are finalized.

J3 internal note

Unresolved issue xxx
The above depends upon the meaning of terminology, e.g. *finalized*, that is yet to be developed.

The set of final subroutines declared in the type is a generic interface. 346:22+

A final subroutine for an object is selected according to the generic resolution rules (14.1.2.4.1).

If the reference is to a deferred binding, an error condition occurs.