Subject: Define "component order" term, do more work on constructors
From: Van Snyder

1 Introduction

Section **4.5.6 Construction of derived-type values** doesn't work for extended types. Section **4.5.3.1 Inheritance** defines the order of components of an extended type, for purposes of derived-type value construction and intrinsic input/output, but doesn't define the term.

This paper defines the term for nonextensible, base and extended types, and uses the term for value construction and intrinsic input/output.

2 Edits

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 (before) the indicated line. Remarks for the editor are noted in the margin, or appear between [and] in the text.

[Editor: Delete "For purposes..." to the end of the paragraph.] 53:13-16

4.5.3$\frac{1}{2}$ Component order 55:0+

[Editor: Insert "component order" into the index.]

The **component order** of the components of a derived type is the component order of the components inherited from the parent type, if the type is an extended type and the parent type has components, followed by the order of the declarations of components declared in the derived type definition.

The component order of the ultimate components of a derived type is the order of the ultimate components inherited from the parent type, if the type is an extended type and the parent type has components, followed by the order of the declarations of components that are of intrinsic type, and the ultimate components that result from declarations of components of derived type, taken in the order the declarations appear in the derived type definition.

The structure constructor for any derived type may be in **flattened form**, in which values may be provided for components inherited from the parent type, if any. The structure constructor for an extended type may be in **nested form**, which allows providing a single value for the parent subobject. 55:28+

[Editor: Replace text in 00-007 and text added within this area by 00-148 by the following:] 55:31-56:3

Constraint: The type name shall be accessible in the scoping unit containing the structure constructor.

Constraint: In the flattened form, there shall be at most one *component-spec* corresponding to each component of the type. In the nested form, there shall be at most one *component-spec* corresponding to each component declared for the extended type.

Constraint: In the flattened form, there shall be exactly one *component-spec* corresponding to each component of the type that does not have default initialization. In the nested form, there shall be exactly one *component-spec* corresponding to the parent subobject of the type, and exactly one *component-spec* corresponding to each component declared for the extended type that does not have default initialization.

Constraint: The *keyword* = may be omitted from a *component-spec* only if the *keyword* = has been omitted from each preceeding *component-spec* in the constructor.

Constraint: In the flattened form, each *keyword* shall be the name of a component of the type. In the nested form, each *keyword* shall be the name of a component declared for the extended type, or the name of the parent subobject.

Constraint: In the nested form, there shall not be a *keyword* that is the same as the name of any component inherited from the parent type.

Constraint: In the nested form, the parent type and all components declared for the extended type shall be accessible in the scoping unit containing the structure constructor. In the flattened form, all components of the type shall be accessible in the scoping unit containing the structure constructor.

If the first *component-spec* has no keyword and the type of the *expr* is the same as the parent type, or if there is a *component-spec* with a keyword that is the same as the name of the parent subobject, the constructor is in nested form. Otherwise, the constructor is in flattened form.

In the nested form, in the absence of a component name keyword, the first *expr* is assigned to the parent subobject, the second *expr* is assigned to the first component declared in the derived type definition, and each subsequent *expr* is assigned to the sequentially corresponding component declared in the derived type definition.

In the flattened form, in the absence of a component name keyword, each *expr* is assigned to the corresponding component of the type, with the components taken in component order (4.5.3$\frac{1}{2}$).

If the keyword is the same as the name of the parent subobject, the *expr* is assigned to the parent subobject; otherwise the *expr* is assigned to the component named by the keyword.

[Editor: Replace text in 00-007 and text added within this area by 00-148 by the following:] 56:6-19

The value that corresponds to the parent subobject is assigned to the parent subobject using intrinsic assignment.

For nonpointer components, the corresponding value is assigned to the corresponding component using intrinsic assignment (7.5.1.4).

Note 4.44$\frac{1}{2}$

The rules for intrinsic assignment apply to the value and component.

The previous semantics were "converted according to the rules of intrinsic assignment to a value that has the same type and type parameters as the corresponding component. The shape of the expression shall correspond to the shape of the component." Since this didn't say it did intrinsic assignment, there's some question how it handles pointer and allocatable components of a derived type component value. The revision clarifies this, and also allows a scalar *expr* to be assigned to an array component.	*Note to J3*

For pointer components, the corresponding *expr* shall evaluate to an object that would be an allowable target for such a pointer in a pointer assignment statement (7.5.2), and it is assigned to the component using pointer assignment.

[Editor: Delete.] 57:1-3

[Editor: Replace "in the same ... type" by "in the component order (4.5.3$\frac{1}{2}$) of the ultimate components." 184:23-24

[Editor: Replace "components ... comprise" by "effective items (9.5.2) that result from expanding".] 189:37