Date:	12 August 1998
To:	J3
From:	Van Snyder
Subject:	Edits for explicitly typed allocations
References:	98-146, 98-160, 98-172r3

1 Background

In Fortran 95, character length cannot be deferred until allocation. In Fortran 2000, without change, one will in addition not be able to defer specifying parameters of parameterized derived types until allocation, nor will one be able to allocate a polymorphic object with a type extended from its declared type. This paper addresses those deficiencies.

Specifications and syntax were proposed in 98-172r2, and approved with minor amendments. A post-meeting 98-172r3 will include the amendments.

2 Edits

Edits refer to 98-007r2. 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.

[Editor: Change “...or assumed” to “...or be assumed or deferred”.] [30:28]

[Editor: Start a new section after note 4.4. Add “deferred” to the index.] [30:33+]

4.2.1 Deferred type parameters

A **deferred type parameter** is a nonkind type parameter of an entity for which specifying the value is deferred until the entity is allocated (6.3.1), associated to a target (7.5.2), or associated to an actual argument (12.4.1). The values of deferred type parameters of unallocated arrays or disassociated pointers are undefined. A deferred type parameter may be specified in a type declaration statement or a component definition statement. A deferred type parameter may be specified only if the entity has the POINTER or ALLOCATABLE attribute. A deferred type parameter is indicated by a colon. An entity that is declared with a deferred type parameter is a **deferred-parameter entity**.

[Editor: Replace clause beginning “it is required...” with the following:] [38:13-14]

it is required to specify values explicitly for all kind parameters of a derived type entity, and either to defer or specify values explicitly for all nonkind parameters of a derived type entity.

---

or : [48:38+]

Constraint: A colon may be used as a *type-param-value* only for a nonkind type parameter. [49:8+]

Constraint: A colon may be used as a *type-param-value* only if the declared entity has the POINTER or ALLOCATABLE attribute.

[Editor: Move the constraints at [49:4-6] to here, and make them ordinary normative text – so we can use *type-spec* in `allocate` statements, where type parameters are not required to be specification expressions.]

[Editor: Start a new paragraph. Add "deferred" to the index.] [49:16+]

A nonkind parameter of a derived type that is specified by a colon is a deferred type parameter (4.2.1). Specification of the values of deferred parameters is deferred until execution of an ALLOCATE statement (6.3.1), pointer assignment (7.5.2), or argument association (12.4.1).

[Editor: Delete the constraint. Replace "allocatable array" by "allocatable variable" everywhere it appears.] [54:6]

R510 *char-len-param-value*	**is**	*scalar-int-expr*	[57:2]
	or	:	[57:3+]

Constraint: If *char-len-param-value* is colon the declared entities shall also have either the POINTER or the ALLOCATABLE attribute. [57:8+]

If the character length parameter value is an integer expression it shall be a specification expression. [57:12.5+]

The reason to specify this using normative text instead of a syntax term at 57:2 is to allow *type-spec* to be used in `allocate` statements, where type parameters are not required to be specification expressions. *J3 note*

[Editor: Start a new paragraph. Add "deferred" to the index.] [57:18+]

A **deferred character length parameter** is a deferred nonkind type parameter (4.2.1) of a character entity; it is specified by a colon *char-len-param-value.* Specification of the length of the character entity is deferred until execution of an ALLOCATE statement (6.3.1), pointer assignment (7.5.2), or argument association (12.4.1).

in a type declaration statement, a component definition statement, or an ALLOCATE statement. Nonkind type parameters may be deferred. [62:40]

and type parameters may be specified in a type declaration statement, a component definition statement, or an ALLOCATE statement. Nonkind type parameters may be deferred. [63:2]

R623 *allocate-stmt*	**is**	ALLOCATE ([*type-spec* ::] *allocation-list* ■ ■ [, STAT = *stat-variable*])	[90:15]

[Editor: Change "array" to "variable". [90:21, 23] |

Constraint: The *type-spec* shall not be CLASS. [90:23+]

Constraint: If a *type-spec* is specified, allocate objects that are not polymorphic shall be of the type specified by *type-spec*, and allocate objects that are polymorphic (5.1.1.8) shall have a declared type that is an ancestor type (4.5.3) of the type specified by *type-spec.*

Constraint: If the shape of an allocate object is deferred, *allocate-shape-spec-list* shall be specified.

[Editor: Start a new paragraph.] [90:25+]

At the time an ALLOCATE statement is executed type parameters may be specified by expressions given in the *type-spec* in the allocate statement. A value shall be specified for every deferred type parameter. If a value is specified for a nondeferred type parameter, it shall be the same as the value specified in the object's declaration. Otherwise, an error condition exists.

Type parameter values are associated to type parameter names as specified in 4.5.5. If a type parameter value is associated by position to a type parameter name, a type parameter value shall be associated by position to all type parameter names that appear earlier in the type parameter name list (including type parameter names inherited from an extensible type's parent type).

Subsequent redefinition or undefinition of any entity within any expression that provides a value for a deferred type parameter does not affect the value of the parameter.

If a type is specified, allocation of a polymorphic object (5.1.1.8) allocates an object with the specified dynamic type; otherwise it allocates an object with the dynamic type of the declared type of the object.

If an *allocate-shape-spec-list* is specified for an array that is not a deferred-shape array, the bounds specified shall be the same as the bounds for the object; otherwise, an error condition exists.

[Editor: Change "bound" to "bound or type parameter value".] [90:26]

[Editor: Change "array" to "deferred-shape array".] [90:34]

[Editor: Delete (it was moved to 90:25+).] [91:1-2]

[Editor: Start a new paragraph] [124:7+]

If *pointer-object* has deferred nonkind parameters, the values of those parameters are assumed from the values of the corresponding parameters of *target*.

J3 note Do we need to say anything about the relation of explicitly specified nonkind parameters of *pointer-object* and *target*? Character length is apparently not a problem, but are the values of other explicitly specified nonkind type parameters of *pointer-object* required to be the same as corresponding nonkind type parameters of *target*?

[Editor: Change "character length is assumed" to "character length is assumed or deferred"] [216:18]

[Editor: Change "or a target" to ", a target, or has a deferred type parameter".] [217:22]

[Editor: Replace "type parameters" by "non-deferred type parameters".] [226:20-21]

[Editor: Replace "type parameters" by "non-deferred type parameters".] [227:21]

[Editor: Add new paragraph after Note 12.21] [228:11+]

If a dummy argument is a deferred-parameter entity and does not have INTENT(OUT) the values of the deferred type parameters are assumed from the corresponding type parameters of the actual argument. If the dummy argument has INTENT(OUT), the corresponding actual argument is a pointer and its pointer association status (14.6.2.1) is disassociated, or the corresponding actual argument is allocatable but not allocated, the deferred type parameters are undefined.

If a dummy argument does not have INTENT(IN) the corresponding actual argument shall have deferred the same type parameters as the dummy argument.

J3 note There really isn't any point to the combination of INTENT(IN) and deferred parameters, since one could in this case use assumed parameters. It is, however, easier to allow than to prohibit the combination.

---

deferred type parameter (4.2.1) A derived type parameter specified by a colon. [343:25+]

deferred-parameter entity (4.2.1) A derived type or character entity having a deferred type parameter.