Date: 26 June 1998
To: J3
From: Van Snyder
Subject: Explicitly typed allocations - Rationale, Specs, Syntax, Edits
References: 98-146, 98-160

# 1 Rationale

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.

# 2 Specs

Change the syntax of declarations to indicate that nonkind type parameters are deferred until allocation. Change the syntax of R624 *allocation* so that the type, and deferred nonkind type parameters in addition to dimensions, can be specified. Extend the semantics of `allocate` to allow specifying a type of a polymorphic object that is an extension of its declared type.

# 3 Syntax

In a declaration, allow ":" to stand for a deferred nonkind parameter. Paper 98-160 reports the deliberations of /data subgroup concerning the preliminary proposal, paper 98-146 *Discussion paper – explicitly typed allocations*. No conclusion was reached concerning the most desirable syntax to denote a deferred parameter, but using ":" appeared to be least objectionable. E.g.

```
character(len=:), allocatable :: char_arr(:)
```

Extend the syntax of `allocate` to specify the type and exactly all deferred parameters, using a *type-spec*. E.g.

```
allocate ( character(len=16) :: char_arr(23) )
```

A *type-spec* appearing in an `allocate` statement affects only the immediately succeeding *allocate-obj*.

# 4 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 attribute.

---

[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). *Redundant?*

---

**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+]

---

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

A **deferred character length parameter** is a deferred 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). *Redundant?*

---

[Editor: In the same paragraph] [62:40+]

Nonkind type parameters may be deferred.

---

Editor: In the same paragraph] [63:2+]

Nonkind type parameters may be deferred.

---

R624 *allocation* **is** [ *type-spec* :: ] *allocate-object* ■ [90:15]
■ [ ( *allocate-shape-spec-list* ) ]

---

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

Constraint: If the *allocate-object* is not polymorphic, the *type-spec* shall specify the same type as the type of the *allocate-object*; if the *allocate-object* is polymorphic (5.1.1.8) the *type-spec* shall specify a type that is an extension type (4.5.3) of the declared type of the *allocate-object*.

Constraint: The *type-spec* shall specify exactly all of the deferred parameters of the *allocate-object*. No other parameters shall be specified.

Remove "exactly" or replace it by "at least", and strike the last sentence? Replace "No other parameters" by "No other nonkind parameters"? Also see section 5. *Straw vote*

---

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

A *type-spec* in an *allocation* applies only to that *allocation*.

At the time an ALLOCATE statement is executed for a deferred-parameter entity the values of the deferred parameters are specified by expressions given in the *type-spec* in the *allocation*. 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.

---

[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*.

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 what about nonkind parameters of PDT's? *J3 note*

---

[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.

There really isn't any point to the combination of INTENT(IN) and deferred parameters, since one could in this case use assumed parameters. Well, maybe there is. I couldn't figure out the relation between INTENT and ALLOCATABLE. Is it similar to POINTER (refers to the allocatability), or does INTENT refer to the value? In any case, it seems easier to allow INTENT(IN) and deferred parameters than to prohibit the combination. *J3 note*

---

**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.

## 5 J3 question

Should J3 revisit the decision to require that all dimensions be deferred for an allocatable or pointer array? The present paper does not propose to require that if one nonkind parameter is deferred then all must be. This seems inconsistent to the situation with deferred shape.