WG5 Document N1194 Dresden, July 1996

To : WG5
From : Steve Morgan
Date : 4th July 1996
Subject : Parameterised Derived Types (PDTs) - Draft Progress Report

---

Following the last WG5 meeting in San Diego in November '95, X3J3 produced a list of concerns about the draft TR. There was some discussion (but not much) on the PDT mailing list and I produced a new Draft TR and a Rationale document which tried to dispel some popular myths about the proposals and also address the concerns that X3J3 had raised.

I attended X3J3 meeting in Las Vegas, in May, and presented the Rationale. It was clear from this meeting, and further discussion on the PDT discussion list, that consensus on the approach in the first draft TR would be difficult to attain.

The main bone of contention was the form of the parameter inquiry. The discussions largely rejected the single function inquiry approach (INQUIRE(object, param-name)) and the %% selector approach as being ugly and likely to produce contorted syntax (for example, since inquiries can appear in initialization and specification expressions).

The two remaining possibilities which seem to be viable, and have support from a significant number of people on both sides (two non-intersecting sets!), are the generic inquiry function approach (i.e. param-name(object)) based on the name of the parameter and the component selector approach which uses the component selection syntax (i.e. object%param-name).

The main objections to the former were against the so-called "invasion of namespace" caused by the function inquiry approach. This is deemed to cause a name management problem for the user since a module which uses parameterized derived types will necessarily introduce new generic function names into the scope of the USE statement.

In the latter approach parameters are treated like components of their parent derived type. There is some concern with this approach that the similarity with components will imply a similar implementation model for parameters. Such an implementation model, although valid, will not be the most efficient in many cases.

On other issues such as declaration of types and structure constructors, although there were some concerns, there did not seem to be so much controversy. Renaming of parameters on USE statements caused some concern. Some did not like the renaming of parameter keywords.

The idea of having 'static' parameters which can be used to resolve generic function references was popular in Las Vegas, and subsequent discussions, so I have implemented it for both versions of the TR. It would be easy to remove it if it became unpopular again.

So... what I have ended up doing is producing a Draft TR which hopefully addresses the concerns discussed in Las Vegas and implements the component selection approach. The Draft TR also provides alternative (shaded) text in certain places so that the original Draft TR can be restored should that be desired.

Thus there would be two approaches to discuss at the next WG5 meeting in Dresden with (effectively) two draft TRs ready on the blocks should a decision to proceed with either be made.

I have put the new Draft TR on the WG5 server. It is document N1193.

In the following I have presented through the, by now, well-known (to me anyway) example of the matrix parameterised derived type, how the example would look should either of the possible TRs be accepted. This is only intended to give a **brief** idea of how the two approaches differ.

N.B. A full appreciation of the two approaches can only be obtained from the rationale and especially the detailed edits presented in the Draft TR.

In the text I have called the 'component approach' Approach 1 and the 'generic inquiry function approach' Approach 2.

Definition of the type

Approach 1

```
TYPE matrix(wkp,dim)
  INTEGER :: wkp, dim
  REAL(KIND=wkp)  :: elem(dim,dim)
ENDTYPE
```

Declaration of parameters is mandatory as it is for other components. Only default integer declarations are allowed for parameters. Explicit declaration avoids any confusion with implicit typing rules.

By default parameters which are subsequently used to specify the kind of a component are static and others are nonstatic unless they are specified with the STATIC attribute as in:-

```
INTEGER, STATIC :: param
```

Approach 2

```
TYPE matrix(wkp,dim)
  REAL(KIND=wkp)  :: elem(dim,dim)
ENDTYPE
```

No declarations mandated . Parameters are default integer.

Parameters can be explicitly declared if required but such declarations are redundant except in cases where it is required to define a parameter as static when it is not subsequently used to determine the kind of a component.

Static parameters can thus be declared in a statement such as

```
INTEGER, STATIC :: dim
```

This would make dim static even though it is not used to specify the kind of a component.

PLEASE note that 'INTEGER, STATIC' can have any spelling (e.g. STATIC). It does not materially affect approach 2.

Structure Constructor

Approach 1

matrix(<parameter-expr-list>,<component-expr-list)

This order is sensible since it will usually correspond with the order in which the parameters and components are declared.

Approach 2

matrix(<component-expr-list>, <parameter-expr-list>)

This form is chosen to be compatible with the ordering of the parameters in REAL(A,KIND).

In both approaches the TR provides edits to implement a simple constructor without keyword arguments and also an additional set of edits to make constructors generic function references. The latter edits allow keywords to be used for both parameter and component arguments.

As currently defined the arguments of the constructor can be in any order in their grouping but all the parameter ones must come before or after all the component ones (depending on the approach). I don't see much harm in this since users will probably think of components and parameters being separate anyway. However this restriction could be easily lifted by introducing some extra BNF with constraints and explanation. I think this would be easier to do with approach 1.

Type parameter value inquiry

For the matrix object M defined as,

TYPE(matrix(wkp=4,dim=20) :: M

In the first approach we would have M%wkp and M%dim and in the second approach we would have wkp(M) and dim(M) to inquire on parameter values.

The former expressions cannot appear in contexts which assign them a value. That is, the first approach treats parameters as read-only components.

In order to regularise the language associated with inquiries, the ability to inquire on intrinsic parameters with syntax such as numeric_object%KIND and character_object%LEN has been introduced. [A proposal to make the LEN() and KIND() inquiries obsolescent could be tentatively made but is not in the TR. It is probably best left for Fortran 2000.]