****
J3/02-274r1

Date: November 2002
To: J3
From: Aleksandar Donev
Subject: MOLD in ALLOCATE
Reference: J3-007R2, J3/02-294r1
****
This paper continues discussion in 02-294r2 and restricts it to a specific proposal.

Summary

A MOLD argument should be added to ALLOCATE similar to the SOURCE argument, which would only give the type of the allocated data, but not the contents. This would allow one to allocate an array of a given (dynamic) type given only a scalar of that (dynamic) type. I also suggest we fix the current specification of SOURCE to make it an expression, and not a variable as mandated at present.

This would be a (limited) analog of the creation operators in other OOP languages, just like SOURCE in allocate provides a cloning operator (see 02-294r1). This is a point to be kept in mind by those who may say SOURCE is enough.

Motivation

Certain dynamic data-structures strain the memory allocator to a maximum, requesting allocation and deallocation of lots of nodes (i.e., small allocations), while in actuality not really changing the total allocated memory much. To deal with such situations efficiently across a variety of compilers and usage patterns, it is best to include as part of the dynamic data structure a memory "allocator".

In Fortran 2002, these types of dynamic data structured would be OO and have polymorphic nodes. However, implementing the custom allocator described above requires the following:
   Allocate an array of objects of a given dynamic type.
which is not really available in Fortran 2002.

Solution

To fix the above defficiency, I propose to add a MOLD argument to ALLOCATE, with the same semantics as SOURCE, but without the contents in the lines 111: 22 and 112: 8-10 given above. The mold variable will thus simply serve as a type template for the allocation, and can be a scalar even if the allocate-object is an array.

In summary,

```
CLASS(base_type), POINTER :: variable, expression
ALLOCATE(variable, MOLD=expression)
```

should be equivalent to:

```
ALLOCATE(dynamic_type_of_expression::variable)
```

when the type of the mold is accessible.

Edits

Optional:
107: Replace <source-variable> with <source-expression>

108: Replace <mold-variable> with <mold-expression>
everywhere in the document and in the edits below.

108: 15+ Add:
or MOLD = <mold-variable>

109: 4 Replace and add:
<source-variable> is <expr>
<mold-variable> is <expr>

109: 6 Replace:
", either <type-spec>, MOLD=, or SOURCE= shall appear"

109: 10 do the same as for 109:6

109: 19+ Add constraint:
(Cxxx) Both SOURCE= and MOLD= shall not appear.

109: 20 Add:
"IF SOURCE= or MOLD= appears," and
"with <source-variable> or <mold-variable>"

109: 23 Add:
"and <source-variable> or <mold-variable>"

109: 28
Delete ",source variable" if we make the source an expression.
[it is an expression and cannot be deallocated.]

109: 33 Add:
"if a <source-variable> or <mold-variable>"
109: 34 Add:
"the <source-variable> or <mold-variable>"

110:10+ Add:
"If MOLD= appears and the value of a nondeferred nonkind type parameter of
<allocate-object> is different from the value of the corresponding type parameter of
<mold-variable>, an error condition occurs."

! EOF