****

J3/02-274

Date: September 23 2002
To: J3
From: Aleksandar Donev
Subject: SOURCE versus MOLD in ALLOCATE
Reference: J3-007R2

****

Note: This paper refers to J3-007R2 since I still do not have access to 007R3. Since the subject discussed did not change from R2 to R3, I hope this is OK.

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 type given only a scalar of that type, which is needed to design custom allocators for various dynamic data structures.

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 my Fortran 95 implementation of Skip Lists, a probabilistic search linked-list data structure, I implemented such an allocator. The basic idea is for the allocator to allocate new nodes in batches, i.e. allocate arrays (pages) of nodes and then maintain a heap or stack of these pages. Deletions are handled so that the deallocated node is added to a stack of deallocated nodes (using the link pointers already present in each node). Attempts are made to deallocate from the least used page so as to free it completely if possible and release the whole page. Allocations are handled by popping nodes off the stack of free nodes if this stack is not empty, and allocating a whole new page of nodes otherwise. This has the effect of improving data locality, avoiding memory fragmentation, and reducing (slow) interactions with the OS to a minimum. This allocator almost certainly outperforms the run-time library (RTL) Fortran allocator in real applications as it is custom-made for a certain type of allocation pattern.

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.

At present, there is no way to do this in Fortran 2002. We can either specify the type in an ALLOCATE statement, which we cannot do here as we do not know the exact type, or we can use a SOURCE argument. The problem is:

p. 111: 22
(R623) The source-variable shall be a scalar or have the same rank as allocate-object

which means that one cannot allocate a whole array given just a scalar as a type template (mold).

Furthermore, allocation with a SOURCE argument really does a "clone" operation, in that it also copies the contents of the source variable:

p. 112: 8-10:
source-variable is then assigned to allocate-object by intrinsic assignment for objects whose declared type is the dynamic type of source-variable.

Clone operations are very important in OOP and need to be in the language. In personal conversations with Malcolm Cohen, he expressed conviction that cloning is the only important allocation with a dynamic type. However, I am documenting here an important class of applications which require the allocation of an array of a dynamic type given by a mold polymorphic variable, without copying the contents of the mold.

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.

It may also be possible to simply replace the SOURCE argument with the MOLD argument if certain constraints for intrinsic assignment are modified. However, I leave this to a future paper which will discuss issues related to typing and polymorphic variables in Fortran 2002.

Edits

Edits will be given in a revision paper once 007R3 is released. They will consist of adding MOLD in all places where SOURCE appears, other then the excluded lines given above.

! EOF