

## Re: ALLOCATABLE arrays

**Source:** <http://coding.derkeiler.com/Archive/Fortran/comp.lang.fortran/2004-01/0175.html>

---

**From:** Jugoslav Dujic (*jdujicREMOVE\_at\_uns.ns.ac.yu*)

**Date:** 01/06/04

Date: Tue, 6 Jan 2004 18:01:37 +0100

Gus Gassmann wrote:

| Tony Jay wrote:

|

||| Do I read this right?

|||

||| Suppose I declare

|||

||| subroutine test

||| integer :: i

||| dimension i(k)

|||

||| where k is set in a module somewhere. Is it then the case that

||| a) i gets allocated automatically on entry to test and deallocated

||| on exit

||| b) this operation takes (essentially) zero time?

|||

||| Is that really all there is to it? I'm afraid to look a gift horse

||| in the mouth, but this sounds too good to be true!

|||

|||

|| Gus,

||

|| This technique can cause problems if you have insufficient stack size

|| (especially if you add in a second array at a later date)

||

|| My understanding is ...

||

|| If try to create a very large array on the stack and you do not have enough

|| stack (set up in the linking step of the compiler i think) then your program

|| will fail at run time.

||

|| Allocating on the heap gives you access to a hell of a lot more memory (well

|| as much as your OS can find !)

||

|| The typical allocate on the stack problem is to test with small problems and

|| then end users use large problems bomb out. You then look a fool (speaking

|| from personal experience)

|

comp.lang.fortran: Re: ALLOCATABLE arrays

| Hmmmh. 'Nother question, then: Do I get any control in where the array is  
| allocated?

Not really. Note that the Standard does not even know about terms "stack"  
and "heap" (and there probably are/were some computers which don't/didn't  
have one or another) so it's up to the implementation (compiler).

| Here is my current understanding from reading the various  
| responses, many of which are over my head, I'm afraid:

| Allocatable arrays are allocated on the heap. They can be pretty damn large,  
| and I can inquire whether things went OK. The downside is that it is slow.  
| Automatic arrays are always allocated on the stack. They are very fast, but  
| if the stack overflows, I have no recourse, so I am SOOL. So why use them,  
| except for very small arrays?

Depends on what you call "very small arrays". In my book, 100,000 elements  
(=400 kB), which is probably a bulletproof margin, is not a "very small array",  
and I find it more than enough for my purposes. Your mileage may vary, of  
course.

| Is there a compiler directive that moves automatic arrays to the heap?

Depends on your compiler (which I don't recall you mentioned). But even if there  
is one, what's the point then? Automatic arrays (with all their limitations) are  
implemented on the stack in order to be fast; if speed is not an issue for you,  
then use an allocatable; if you're worried about stack overflow, increase the  
stack size (/stack:0xNNNNNN switch in MS Linker). Usually, one can predict the  
order of magnitude of memory needed.

--

Jugoslav

---

[www.geocities.com/jdujic](http://www.geocities.com/jdujic)