Dynamic Memory Allocation

FORTRAN 90/95 also includes a ways to allocate memory dynamically at execution time: allocatable arrays and Pointers. Allocatable arrays are arrays whose rank is specified at compilation time, but whose shape is not specified until the program is executed. Pointers are variables that contain the address in memory of another variable where data is actually stored. Pointers can also be used for dynamic memory allocation. We shall restrict to allocatable arrays.

ALLOCATABLE ARRAYS

The structure of a typical array declaration with the ALLOCATABLE attribute is

```
REAL, ALLOCATABLE, DIMENSION(:,:)  :: arr1
```

Note that colons are used as placeholders in the declaration, since we do not know how big the array will actually be. The rank of the array is declared in the type declaration statement, but not the size of the array.

When the program executes, the actual size of the array will be determined with an ALLOCATE statement. General form of an ALLOCATE statement is

ALLOCATE (list of arrays to allocate, STAT = status)

For example, ALLOCATE (arr1(100, 0:10), STAT = status)

This statement allocates a 100 × 11 array arr1 at execution time. The STAT = clause is optional. If it is present, it returns a integer status. The status will be 0 for successful allocation and a compiler-dependent positive number if the allocation process fails. The most common source of failure is not having enough free memory to allocate the array. If the allocation fails and the STAT = clause is not present, then the program will abort. You should always use the STAT = clause so that the program can terminate gracefully if there is not enough memory available to allocate the array.

When the allocatable array is no longer needed in the program, you should deallocate the memory to make it available for reuse with a DEALLOCATE statement. The structure of DEALLOCATE statement is

DEALLOCATE (list of arrays to deallocate, STAT = status)

For example,

```
DEALLOCATE (arr1, STAT = status)
```

where the status clause has the same meaning as it has in the ALLOCATABLE statement. After a DEALLOCATE statement is executed, the data in the deallocated arrays is no longer available for use. You should always deallocate any arrays when you are finished with them.