

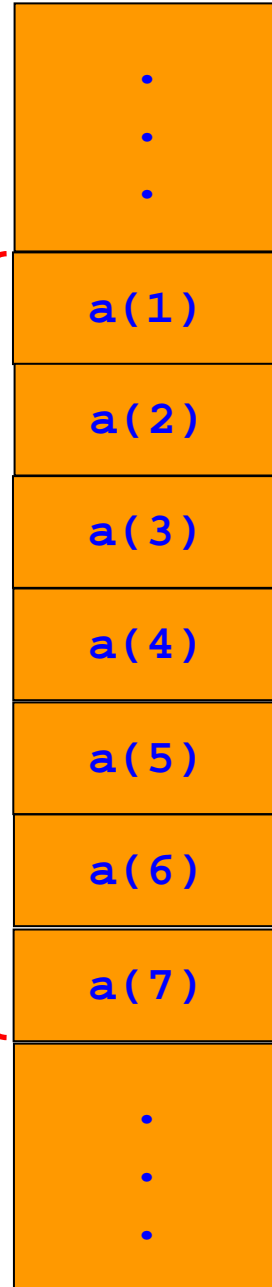
Goals for This Lecture:

- Learn about how to avoid out of bounds array references via compiler bounds checking
- Learn about whole array operations
- Learn about scalar-array operations
- Learn about elemental functions
- Learn about array subset referencing
- Learn about vector subscripts

Out-of-bounds Array Subscripts

- What happens if we reference an array element that is below the lower bound of the array or above the upper bound of the array?
- Example: `a(9)`
- These array elements do not exist
- We refer to this as an out-of-bounds array reference and it is an illegal operation
- Most compilers can be told to build in checking for such operations during execution
 - Intel Compiler: add the `-CB` flag to the compilation command
 - Example: `ifort -CB -o array_read2 array_read2.f90`
- This should always be done while debugging code.
- Bounds checking can slow the execution of the code significantly

Array a



Example: Reading data into arrays

! Purpose: Show how to read data into an array with an integer parameter defining the size

! Author: F. Douglas Swesty

! Date: 10/17/2005

```
program array_read2
```

```
implicit none      ! Turn off implicit typing
```

```
integer, parameter :: lun1=11  ! Define an LUN for the output file
```

```
integer, parameter :: npts=10  ! Number of data points
```

```
real :: x(npts), y(npts)  ! Define arrays to hold data
```

```
integer :: i              ! Loop index
```

```
integer :: ierror  ! I/O error status variable
```

```
open(unit=lun1,file='experiment.dat',status='OLD',iostat=ierror)  ! Open the file
```

```
if(ierror /= 0) then
```

```
    write(*,*) ' Could not read in file!'
```

```
    stop
```

```
endif
```

```
do i=1,npts
```

```
    read(lun1,*) x(i),y(i)  ! Read in (x,y) pairs of data into arrays
```

```
enddo
```

```
close(unit=lun1)          ! Close the file
```

```
stop                      ! Halt execution
```

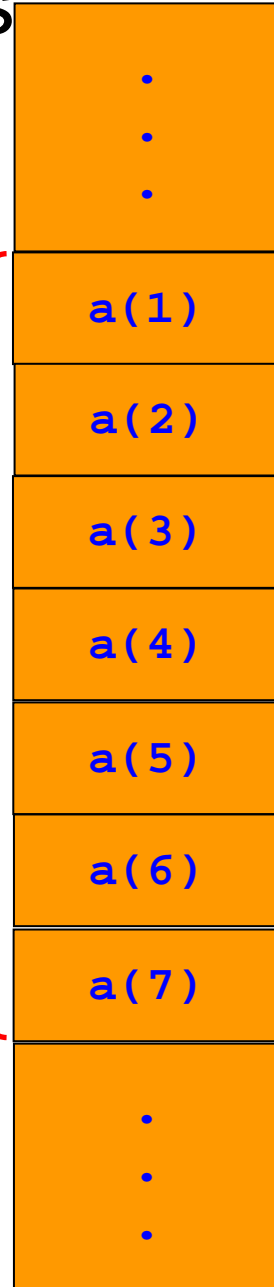
```
end program array_read2
```

Array Elements are Just Variables

- Each element of an array is just like an ordinary variable
- Array elements can be used in any expression where a variable would be used
- Examples:

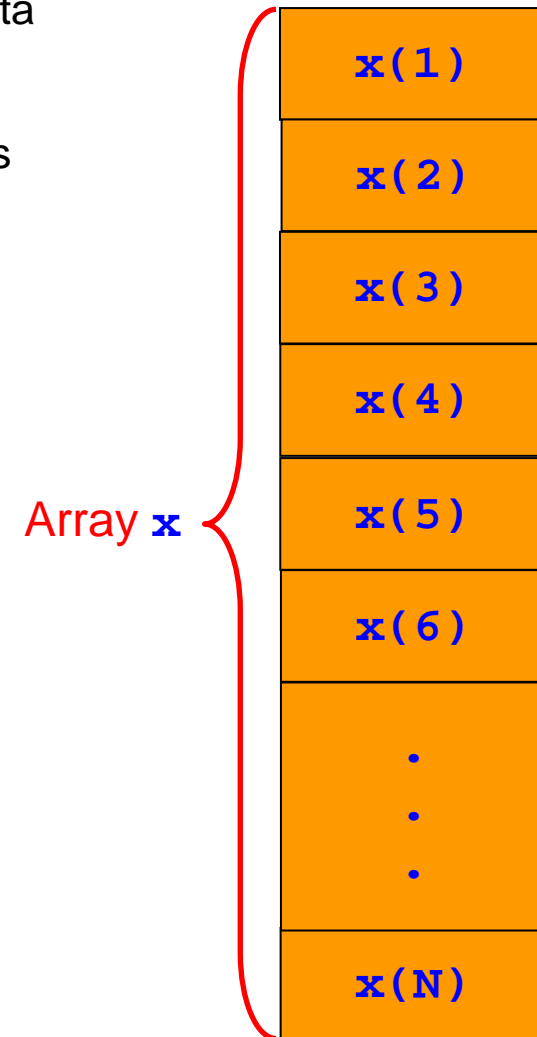
```
a(9) = 3.14 + x**2  
xmid = 0.5 * (x(i) + x(i+1))  
y = 5.0 * sin(theta(n))  
write(*,*) ' v(k) = ', v(k)
```

Array a



An example of array usage: max & min of data set

- Top down design steps:
 - Objective: Find the maximum and minimum of a set of real data and the location of these extrema in the list of data
 - Data needed: an array of data x , a loop index, i/o error status maximum value, minimum value, max location, min location
 - Algorithm steps:
 1. Read the N data items in from a file into an array $x(N)$
 2. Initialize maximum value to $x(1)$
 3. Initialize minimum value to $x(1)$
 4. Set maximum location to 1
 5. Set minimum location to 1
 6. For $i=2, \dots, N$
 7. If $x(i) > \text{maximum}$ then
 8. set maximum to $x(i)$
 9. set maximum location to i
 - 10.
 11. If $x(i) < \text{minimum}$ then
 12. set minimum to $x(i)$
 13. set minimum location to i
 14. End of loop
 15. Write out maximum & location of minimum
 16. Write out minimum & location of minimum
 17. Stop



```

! Purpose: Find max & min values of a data set
! Author:  F. Douglas Swesty
! Date:    10/19/2005
program maxmin
implicit none      ! Turn off implicit typing
integer, parameter :: lun1=11  ! Define an LUN for the
                               output file
integer, parameter :: npts=10  ! Number of data points
real :: x(npts)           ! Define array to hold data
integer :: i              ! Loop index
integer :: ierror         ! I/O error status variable
real :: max_val, min_val   ! Maximum & minimum
                               values
integer :: max_loc, min_loc ! Maximum & minimum
                               location

                               ! Open the file
open(unit=lun1,file='experiment.dat',status='OLD',iostat=ierror)
if(ierror /= 0) then
  write(*,*) ' Could not read in file!'
  stop
endif

do i=1,npts
  read(lun1,*) x(i) ! Read in data into array
enddo

close(unit=lun1)      ! Close the file

```

```

max_val = x(1)      ! Initialize maximum
min_val = x(1)      ! Initialize minimum
max_loc = 1         ! Initialize maximum location
min_loc = 1         ! Initialize minimum location

do i=2,npts        ! Loop over rest of data set

  if(x(i) > max_val) then ! If x(i) is a new maximum
    max_val = x(i)       ! Reset maximum
    max_loc = i          ! Reset maximum location
  endif

  if(x(i) < min_val) then ! If x(i) is a new minimum
    min_val = x(i)       ! Reset minimum
    min_loc = i          ! Reset minimum location
  endif

enddo

                               ! Write out results
Write(*,*) ' Maximum ',max_val,' at ',max_loc
Write(*,*) ' Minimum ',min_val,' at ',min_loc

stop                               ! Halt execution

end program maxmin

```

Whole Array Operations

- Under some circumstances whole arrays can be used in numerical expressions
- If two arrays have the same shape (same rank & extents) then they can be combined with ordinary arithmetic operations which will be applied on an element by element basis

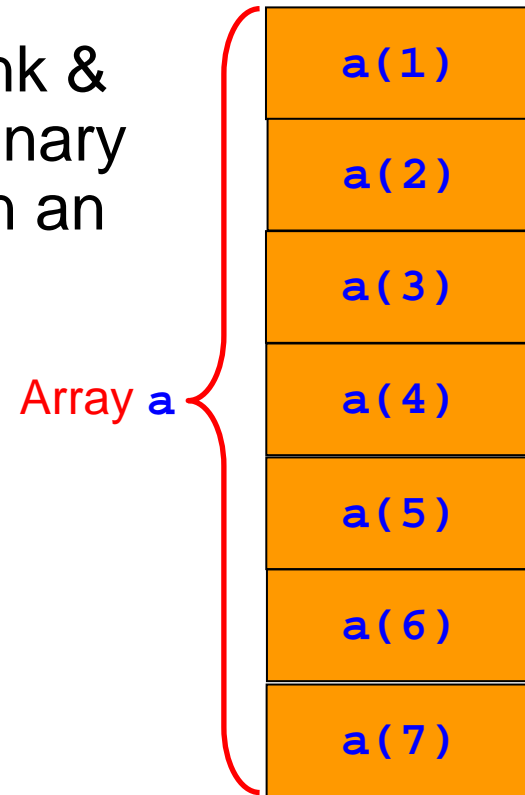
- Example (ignoring initialization):

```
real :: a(7), b(7), c(7), d(-1:5)
```

```
a = b+c
```

```
d = a*c
```

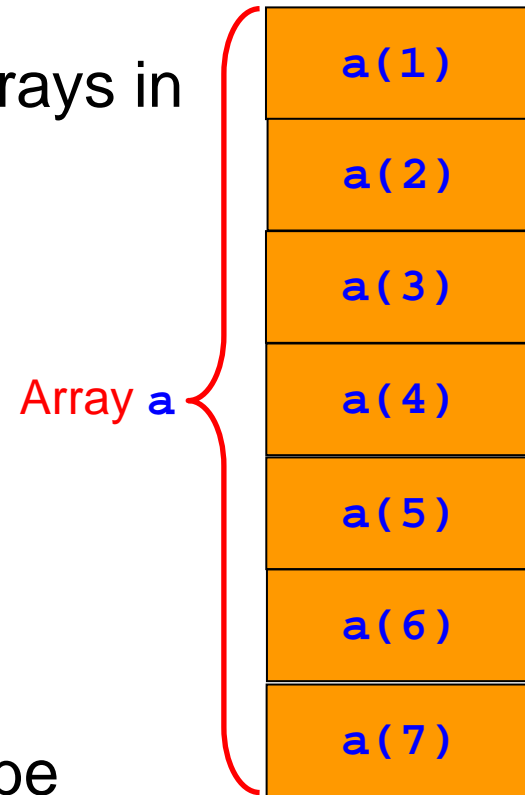
- Two arrays with the same shape are said to be conformable



Array-Scalar Operations

- Arrays are also conformable with scalars
- This means that scalars can combine with arrays in obvious ways
- Examples (ignoring initialization):

```
real :: b(7)= 1.0, c(7)=0.5
real :: a(7), d(-1:5)
a = 2.0*b+c/1.5
d = c+3.14
```
- Two arrays with the same shape are said to be conformable



Elemental Functions

– Certain intrinsic functions can accept arrays as arguments where the functional operation will be applied on an element-by-element basis

– Examples:

```
integer :: i
```

```
real :: a(7)=(/(0.3*i,i=1,10)/),b(7),c(7)
```

```
b = sin(a)
```

```
c=abs(b)
```

Array a

a(1)

a(2)

a(3)

a(4)

a(5)

a(6)

a(7)

– A complete list of elemental functions can be found in appendix B

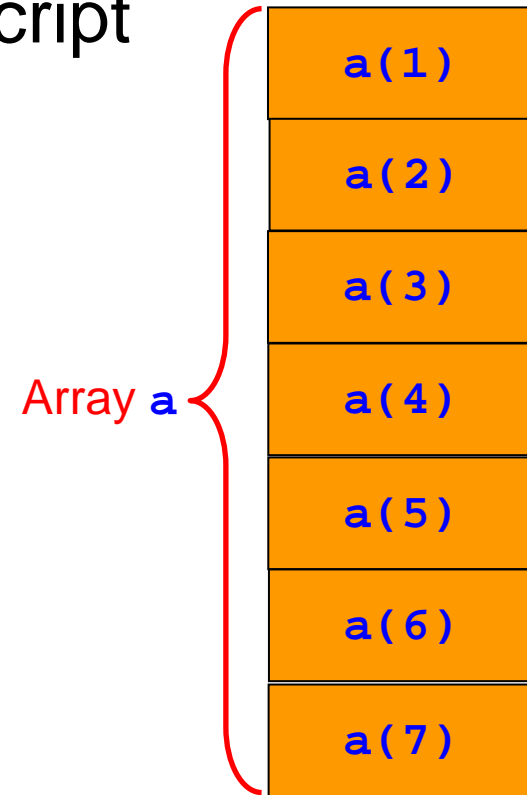
– wo arrays with the same shape are said to be conformable

Array Subsets

– Subsets, a.k.a. slices, of arrays can be referenced through the use of subscript triplets

– Examples:

$a(1:3)$ (elements 1-3)
 $a(1:7:2)$ (elements 1,3,5,7)
 $a(4:)$ (elements 4-7)
 $a(:3)$ (elements 1-3)
 $a(2::3)$ (elements 2,5)
 $a(:4:2)$ (elements 4,6)
 $a>::2$ (elements 1,3,5,7)
 $a(:)$ (all elements of a)



Vector Subscripts

- Subsets of an array can be referenced through the use of a vector subscript (an rank-1 integer array specifying the array elements to be referenced)

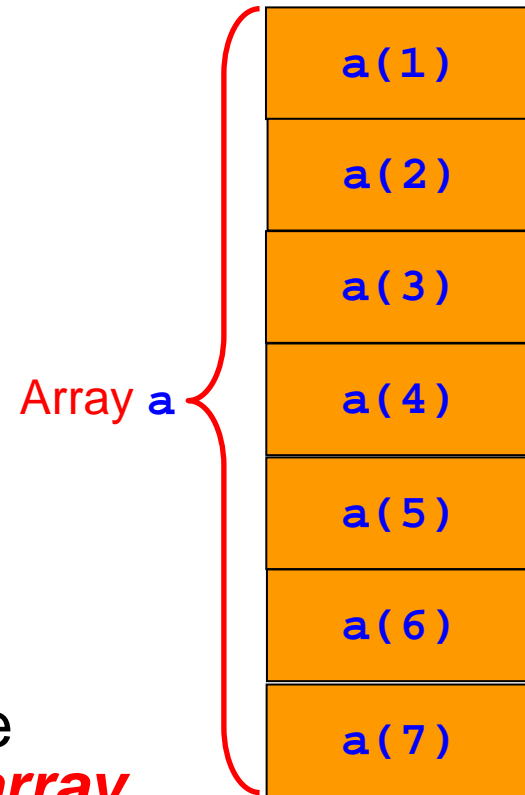
- Examples:

```
integer :: iv(2)=(/1,7/)
```

```
real :: a(7)=1.0
```

```
a(iv) = 0.0 ! Zero the ends of array
```

- If a vector subscript contains the same value more than once it is called a **many-to-one array section** and it cannot be used on the left-hand-side of an assignment statement



Reading Assignment

- Read Sections 6.4-6.7,8.1-8.7