Bits, Bytes, and Precision

- Bit: Smallest amount of information in a computer.
  - Binary: A bit holds either a 0 or 1.
  - Series of bits make up a number.
- Byte: 8 bits.
- Single precision variable: 4 bytes (32 bits)
  - Essentially 10 significant digits.
- Double precision variable: 8 bytes (64 bits)
  - Essentially19 significant digits

Example: Quality controlling hourly station reports on a local mesonet (25 stations)

- Read in temperatures and dew points from a file.
- Find the average temperature and dew point
- Print stations that have temperature and/or dew points greater than ±5 ºF from the mean, sorted so that stations with the largest deviation are printed first
- How do I accomplish this task?
- You could try and read and re-read the file over and over, but this is inefficient. What do you do?

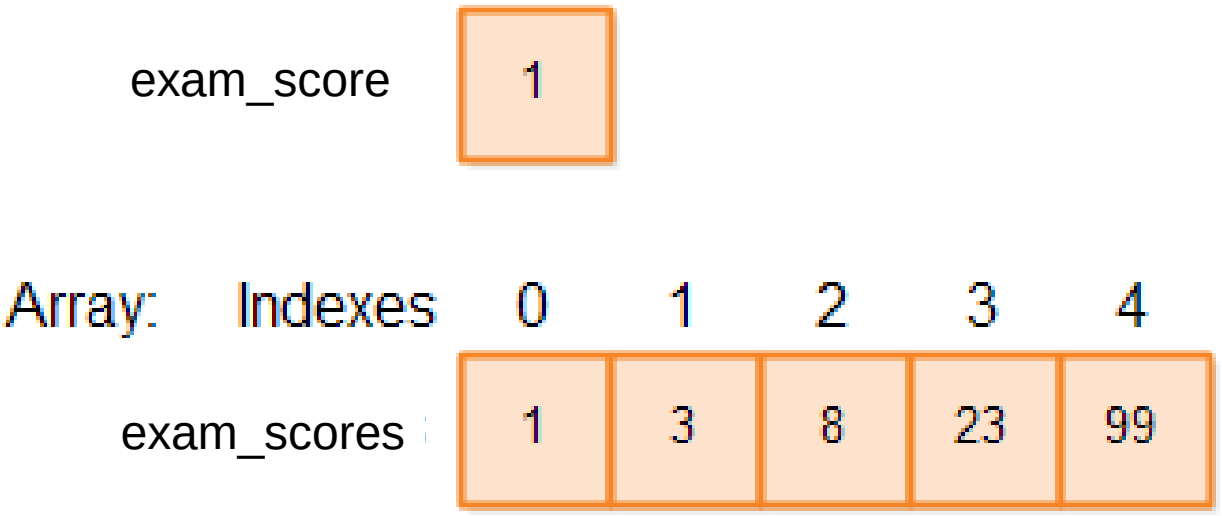
FORTRAN 90: Arrays

Meteorology 2270

Data Structures

- Data structure to store and organize the collection of temperatures and dew points.
- Structure should allow data storage and retrieval.
- Data should be stored in main memory.
- 1-D array: Stores a fixed number of data values, all of the same type.

Variables vs. Arrays



1-D Arrays

- In FORTRAN 90, the type statement would look like:
  - REAL, DIMENSION(25) :: TEMP
  - REAL, DIMENSION(25) :: DEW
  - REAL, DIMENSION(25) :: TEMP, DEW

- Creates two arrays named TEMP and DEW consisting of 25 memory locations for each in which reals can be stored.

Accessing an Array

- Two methods of accessing an array
- Use array-variable (TEMP, DEW) to refer to the entire array.
  - TEMP = 0
- Use a subscripting variable and an index to refer to an individual element
  - TEMP(5), DEW(6) is the 5th temperature and the 6th dew point.
  - Each subscript variable refers to a memory location.
- By default, FORTRAN indexing starts at 1 unless otherwise stated.
  - TEMP(1) is the first element, TEMP(2) is the second, etc.

Examples

- TEMP(5) = 68
  DEW(5) = 66
  DEPRESSION(5) = TEMP(5) – DEW(5)
  PRINT *, “Dew Point Depression = “, DEPRESSION(5)

- READ(10,*) TEMP(N), DEW(N)
  - How might this be used?

- DO N=1, 25
    READ(10,*) TEMP(N), DEW(N)
  END DO

- Mean-time to failure program in book.

Implied DO loops

- Can be used to simplify input and output
  - Useful, but does sacrifice readability.

- (list-of-variables, control-var = init-value, limit, step)

- READ(10,*) (TEMP(N), DEW(N), N=1,25)
  - This is equivalent to the loop on the previous slide.

Compile time and Allocatable Arrays

- What do we currently know?
- Size of arrays are fixed at compile time.
- Small dataset: wasted memory
- Large dataset: Too large to store and process correctly.

Compile time arrays

- type, DIMENSION(l:u) :: list-of-array-names
- type :: list-of-array specifiers
  - array-name(l:u)
- FORTRAN does allow a subscript to be any integer value, positive, negative, or zero
  - Must not fall outside of the range specified.
- INTEGER, PARAMETER ::LowerLimit_1= -1, UpperLimit_1 = 3, &
  LowerLimit_2 = 0, UpperLimit_2 = 5

  INTEGER, DIMENSION(LowerLimit_1 : UpperLimit_1) :: Alpha

  REAL, DIMENSION(LowerLimit_2 : UpperLimit_2) :: Beta

Run-time or Allocatable arrays

- ALLOCATE attribute
  - REAL, DIMENSION(:), ALLOCATABLE :: list-of-array-names
  - REAL, DIMENSION(:), ALLOCATABLE :: A, B

- Later in program…..
  - ALLOCATE(list, STAT = status-variable)
  - List is a list of array specifications of the form array-name(l:u)
  - Status-variable is an integer variable that:
    - 0 is allocation of array is successful.
    - Non-zero if an error occurred.
- Even later in the program
  - DEALLOCATE(list, STAT=status-variable)
  - Status-variable is identical to ALLOCATE statement except for deallocating the array.

Array Constants

- Arrays may be filled by a list of values enclosed between (/ and /)
  - A = (/ 2, 4, 6, 8, 10, 12, 14, 16, 18, 20 /)
- This may be simplified with an implied do loop
  - A = (/ (2*N, N = 1, 10) /)
  - A = (/ 2, 4, (N, N = 6, 18, 2), 20 /)
- How do we use this?
  - INTEGER, DIMENSION(10) :: A
    A = (/ 2, 4, 6, 8, 10, 12, 14, 16, 18, 20 /)
    A = (/ (2*N, N = 1, 10) /)
    A = (/ 2, 4, (N, N = 6, 18, 2), 20 /)
- Each of these are the same as:
  - DO N = 1, 10
      A(N) = 2*N
    END DO

Quick Digression: Modulus

- MOD(A,P) = A – INT(A/P) * P
- Returns the remainder when A is divided by P.
- Examples
  - Z = MOD(9,3)
  - Z = MOD(10,3)
  - Z = MOD(Hours_into_model_run,24)
- Arguments can be integer or real.

Array Expressions

- Operators and functions normally applied to simple expressions may also be applied to arrays having the **same number of elements**.
  - Operations are carried out element by element.

- INTEGER, DIMENSION(4) :: A, B
  INTEGER, DIMENSION(0:3) :: C
  INTEGER, DIMENSION(6:9) :: D
  LOGICAL, DIMENSION(4) :: P
  A = (/ 1, 2, 3, 4 /)
  B = (/ 5, 6, 7, 8 /)
  C = (/ -1, 3, -5, 7 /)
  A = A+B
  D = 2 * ABS(C) + 1
  P = (C > 0) .AND. (MOD(B,3) == 0)

- What are A, D, and P?

- A = 6, 8, 10, 12
  D = 3, 7, 11, 15
  P = .false., .true., .false., .false.

Array Assignment

- Array-variable = expression
- The value of expression assigned to an array variable must be either:
  - An array of the same size as the array variable, or
  - A simple value.
- In the second case, the value is broadcast to all members of the array.

Array Sections and Subarrays

- Allows you to construct new arrays by selecting elements from another array
  - Array-name(subscript-triplet)
  - Array-name(vector-subscript)

- Subscript triplet: lower : upper : stride
  - If lower (upper) is omitted, the lower (upper) bound in the array declaration is used.
- INTEGER, DIMENSION(10) :: A
  INTEGER, DIMENSION(5) :: B, I
  A = (/ 11, 22, 33, 44, 55, 66, 77, 88, 99, 110 /)
  B = A(2 : 10 : 2)

- What is B?

Array sections cont.: Vector-subscripts

- A = (/ 11, 22, 33, 44, 55, 66, 77, 88, 99, 110 /)
  N = (/ 6, 5, 3, 9, 1 /)
  B = A(N)

- Assigns to B the element locations from array A listed by N.
  - Values in N become indices.
  - B = 66, 55, 33, 99, 11

- B = A((/5, 3 ,3 ,4 , 3 /))?

- A(1 : 10 : 2) = (/ N**2, N = 1, 5 /)?

Subarrays cont.: Input/Output

- DO N = 1, NumTemps
  READ(10,*) Temp(N)
  END DO

- READ(10,*) (Temp(N), N = 1, NumTemps)

- READ(10, *) Temp(1:NumTemps)

- PRINT *, Temp(1:NumTemps)

WHERE construct

- Used to assign values to arrays depending on the value of a logical array expression
- WHERE (logical-array-expr)
  array-var = array-expr
  ….
  ELSEWHERE
  array-var = array-expr
  END WHERE

- Logical expression is evaluated element by element.
- INTEGER, DIMENSION(5) :: A = (/ 0, 2, 5, 0, 10 /)
  REAL, DIMENSION(5) :: B

  WHERE (A > 0)
  B = 1.0/ REAL(A)
  ELSEWHERE
  B = -1.0
  END WHERE

- What is B?

- B = -1.0, 0.5, 0.2, -1.0, 0.1

Arrays as Arguments

- Several intrinsic functions exist for arrays
- ALLOCATED(A)
  - Returns true if memory has been allocated to the allocatable array A and false otherwise
- MAXVAL(A)
  - Returns the maximum value of A
- MINVAL(A)
  - Returns the minimum value of A
- MAXLOC(A)
  - Returns a one-dimensional array containing on element whose value is the position of the first occurrence of the maximum value in A.
  - MINLOC(A)
- DOT_PRODUCT(A,B)
  - Returns the dot product of arrays A and B.
- SUM(A)
  - Returns the sum of the elements in A.
- PRODUCT(A)
  - Returns the product of the elements in A.
- SIZE(A)
  - Returns the number of elements in A.
- Others can be found in appendix A.

Multi-Dimensional Arrays

- Many applications
  - Data naturally fits into a table.
    - Times series of temperature measurements from multiple stations.
  - Data is on a grid.
- Example: Hourly data from Flory MicroNet.
  - 50 stations reporting hourly output.
  - Construct table/form array to hold data.
- REAL, DIMENSION(24,50) :: Temperature
  - Indices are arranged as (hour, station)

  REAL, DIMENSION(1:24,1:50) :: Temperature
- Question: What is Temperature(12,12)?

Multi-Dimensional Arrays cont.

- This array only holds data for one day, what if I want multiple days?
- REAL, DIMENSION(365,24,50) :: Temperature
  - Indices are arranged as (day, hour, station).
- What is Temperature(180,12,20)?
- Temperature from the 20^{th} station on the 180^{th} day of year at the 12^{th} hour.

Declarations – Compile time

- type, DIMENSION(L_1:U_1,L_2:U_2,….,L_k:U_k) :: list
- type, list-of-array-specifiers
  - Number of dimensions, k, is called the “rank” of the array.

- REAL, DIMENSION(1:2,-1:3) :: Alpha
  REAL, DIMENSION(0:2,0:3,1:2) :: Beta

- List all of the valid references to each array?

Declarations - Allocatable

- type, DIMENSION(:, :, :, ..., :), ALLOCATABLE :: list
- REAL, DIMENSION (:, :, :), ALLOCATABLE :: Beta
  REAL, DIMENSION (:, :), ALLOCATABLE :: Alpha
- ALLOCATE(Beta(0:2,0:3,1:2), Alpha(1:2,-1:3), &
  STAT = AllocateStatus)
- Example: Table of temperatures.

Sorting and Searching

- Section 8.4

- Sorting: Arranging items in a list so that they are in ascending or descending order.
  - Selection sort, quick sort, bubble sort

- Searching: Finding a specified item and retrieving information associated with that item.
  - Linear search, binary search

Selection Sort

- Traverse the list, or part of the list, several times, each time selecting one item to be correctly positioned.

- Method (ascending order):

    1. Find smallest item and move it to the first position by exchanging it with that number.
    2. Scan the rest of the list starting from position #2 and repeat.
    3. Continue in this manner until the list is sorted.
        1. 3rd element in the list to the end of the list, etc.

- Let's practice.

Quick Sort (Recursive sorting)

- More efficient than a selection sort.
- One of the fastest sorting methods.
- Method:
  - Select pivot element (typically first element).
  - Perform exchanges so that all elements on left of pivot are less than the pivot and all elements on the right are greater than the pivot.
    - Correctly positions the pivot element.
    - Divides the list into two sub-lists.
  - Sort sub-lists independently in the same way.
    - Divide and conquer approach.
- 50, 30, 20, 80, 90, 70, 95, 85, 10, 15, 75, 25

Searching

- Linear Search
  - Begin with first item in list and search sequentially until desired item is found or end of list is reached.

- Binary Search
  - If a list has been sorted, a binary search can be used more efficiently.
  - Method
    - Examine middle element in the list. Is it the desired element?
    - If not, determine if desired item in the first or second half of list.
    - Search that half of the list using the same approach.