Lecture C14: Data structures - Arrays

Response to 'Muddiest Part of the Lecture Cards'

(45 respondents)

1) ***Why on the "Initializing Arrays" slide did we use coord1, coord2, and coord3 in the example?*** (1 student)
I introduced 3 new arrays all of type *small_arrays* defined on the previous slide.

2) ***How do you distinguish between column-major and row-major arrays (besides just knowing)?*** and similar questions (2 students)
That is just it, it is as always up to you as a programmer to interpret the data stored in memory. There is now way we automatically can know how data is stored, without taking a look at the code storing the data in the array.

3) ***How are arrays used as Parameters?*** and similar questions (6 students)
Take a look at 'sum_array.adb' distributed via email today. The example shows how a procedure can be written to add the elements of any array of integers. The RANGE attribute is used so that the loop control variable takes each index value in turn, no matter what the actual range of values is
an **unconstrained** array type is used for the formal parameter, so that any array of integers can be supplied as the parameter, no matter what its size.

4) ***How do you display an entire array?*** (1 student)
We have to print all the items one by one. For example, as shown in the lc_letter_freq.adb (shown in class and distributed via email) where all items of array 'Count' will be printed.

```
for T in Lc_Letters loop
Put(T);
Put(Count(T), Width=>8);
New_Line;
end loop;
```

5) ***Did not know what 16x8, 32x8, 7x8x8 were for?*** and similar questions (2 students)
Just pointing out that 3 different arrays with same amout of items/position (all three arrays large enough to hold 8 items) will need different amout of bytes to be stored in the computer memory.

Array 1: Typical Integer represented using 16bits, thus size of array = 16x8 bits

Array 2: A floating point number can be represented using 32 bits, thus size of array = 32x8 bits
Array 3: One character can be represented using 8 bits (ASCII value for character). Each string in the third array was larg enough to hold 7 characters, thus size of array = 7x8x8 bits

6) ***What are unconstarined arrays?*** and similar questions (3 students)
unconstrained array types:

- element type is specified in type declaration
- index type is specified in type declaration
- range of index values (ie size) is not specified in type declaration
- specify range of index values in variable declarations

Examples:

```
type int_u_array is array (INTEGER range <>) of INTEGER;
small_int_array : int_u_array (1 .. 3); -- just 3 items
big_int_array : int_u_array (1 .. 100); -- 100 items

type char_count is array (CHARACTER range <>) of INTEGER;
subtype digit_count is char_count('0' .. '9');
uc_counts : char_count ('A' .. 'Z');
dig_counts : digit_count;
```

STRING is an unconstrained array type:
```
type STRING is array (POSITIVE range <>) of CHARACTER;
```

7) ***Insertion sort, bubble sort? How does insertion sort know where in the array to put each number? For sorting, what is the meaning of J (for example : for J in I+1 .. Last loop)*** and similar questions (6 students)

The insertion sort algorithm is the sort unknowingly used by most card players when sorting the cards in their hands. When holding a hand of cards, players will often scan their cards from left to right, looking for the first card that is out of place. For example, if the first three cards of a player's hand are 4, 5, 2, he will often be satisfied that the 4 and the 5 are in order relative to each other, but, upon getting to the 2, desires to place it before the 4 and the 5. In that case, the player typically removes the 2 from the list, shifts the 4 and the 5 one spot to the right, and then places the 2 into the first slot on the left. This is insertion sort. Unlike other simple sorts like selection sort and bubble sort which rely primarily on comparing and swapping, the insertion sort achieves a sorted data set by identifying an element that is out of order relative to the elements around it, removing it from the list, shifting elements up one place and then placing the removed element in

its correct location. Follow the step by step process of sorting the following small list.

To understand bubble sort, think of an air bubble rising in water.
To sort N items, N passes are made through the data.

- The result of the first pass is that the smallest item is in the last location of the array.
- The result of the second pass is that the second smallest item is in the second last location of the array.
- etc.
- After N passes, the entire array is sorted.

The operation in each pass is as follows:

- First, the values in the first two locations are compared. If necessary the values are exchanged, so that the smaller one is last.
- Then, the values in the second and third locations are compared. If necessary the values are exchanged, so that again the smaller one is last.
- This process repeats to the end of the array.
- In effect, the smaller item bubbles its way towards the top. It keeps on going until either it reaches the top (and the pass ends), or it strikes a smaller item. In that case it stops itself, and gives that smaller item a nudge to start it on its way.

If a complete pass is made without any exchanges being made, the data must already be sorted. This is easily checked. Thus it might be possible to halt the algorithm without going through all N passes.

8) ***Do we need this for this PSET of the one due next week?*** (1 student)
That would be problem C14 in next weeks pset.

9) ***How does arrays differ from lists?*** and similar questions (2 students)
A list is a kind of *data structure*. By definition, a list is a finite set of entries, all with a certain order. The entries in a list does not have to be of the same type. An array on the other hand is also a data structure that stores its entries sequentially. However, the items in an array has to be of the same type. Arrays hold a fixed number of equally sized data elements, individual elements are accessed by index.

Arrays permit efficient , constant time, random access to its items, but an array is not efficient when it comes to insertion and deletion of elements. This is in contrast to dynamic data structures as for example the linked lists (will be covered in spring term), which have the opposite trade-off. Consequently, arrays are appropriate for storing data which will be accessed in an unpredictable

fashion, and linked lists are best for data accessed sequentially.

10) ***Is there any way to call a certain index other than 'firt & 'last? What is 'last & 'first?*** and similar questions (6 students)

Lets say we have an array declared and instantiated as:

Big_Int_Array : Int_U_Array (1 .. 6) := (1, 4, 9, 16, 25, 36);

If you want to print the second value of that array, try the following:

Ada.Integer_Text_Io.put (big_int_array (2)); -- prints a '4' on the screen

'last returns the value of the smallest index
'first returns the value of the largest index

11) ***How can a, b , and c be Float values?*** (1 student)
On slide 'Example [3/3]' the following can be found: "**type** small_arrays **is array** ('a' .. 'c') of FLOAT; "

That defines a type for an array large enough to hold three (given by the range 'a'..'c') items, all items in the array are of type float. The range 'a'..'b' only shows that the index type for the items in the array is of type charcter. Remember that *element type* is **not** related to *index type*.

12) ***When comparing arrays, do they have to be of the same length?*** (1 student)
For them to be equal yes, they have to be of same length, and all items must be of same type and identical. Two arrays of different size/length can be compared, but will always result as "not equal".

13) ***In arrays, how is the range different than the length?*** (1 student)
Length says how many items the array can contain. Range tells us the values of the indexes, e.g., -5 .. 2

14) ***Are arrays most vital to an Ada code using a for..loop or a while..loop?*** (1 student)
Not really sure on how to interprete this question. An array is just a data type, that is, a way data data is stored in memory. We can use both FOR and WHILE operations interchangeably to operate on the items in an array.

15) ***Why wouldnt we always create dynamic arrays and fix their size at variable declaration? That way ?*** (1 student)

The goal behind using unconstrained arrays is to ensure that the arrays all have the same type and can be manipulated together even if they don't have the same size. The arrays are not dynamic. They are created at compile time. We will discuss dynamic array creation next semester.

16) ***When are we going to learn about classes and pointers?*** (1 student)
Pointers and linked lists will be spoken about during spring term.

What in C++ is called classes is called "tagged types" in Ada. I do not intend to cover that in this course.

17) ***Can you add arrays together, how do you pick an index?*** (1 student)
yes you can: Line := Line1 & Line2;

Depends on what you ment with add?

18) ***So, can an array be seen as a matrix? Does that mean we can use it to solve systems?*** (1 students)

A matrix is represented in Ada as a two-dimensional array. The elements of the array can be manipulated using the matrix package that has been created by Drexel University. Yes, you can use it to solve systems but you will have to write the code yourself. The matrix package only provides the matrix manipulation functions.

19) ***Is there any way to have an array of strings of varying legth?*** (1 student)

Arrays can only be used to store homogeneous elements. The array can still hold strings of varying length but the size of each element in the array will be the size of the largest string in the array.

20) ***"No mud"*** (12 student)
Good :-)