60-140 INTRODUCTION TO ALGORITHMS AND PROGRAMMING I

Dr. Ziad Kobti

Introduction to Arrays

Objectives:

1. Learn how to declare and set the size of an array.
2. Learn how to initialize an array, access array's elements and print them.
3. Simple manipulation of arrays.
4. Arrays and functions – how to pass an array to a function.

Instructions:

Recall in the previous lab you learned how to create a directory, change into it, and view its contents. (i.e. the UNIX commands you need to understand here are `mkdir` *`directory_name`*, `cd` *`directory_name`*, and `ls`)

In this lab you are asked to create one complete C program that includes several functions, but all inside the same source code file named **Lab7_q.c** in a directory called **lab7** which should be located inside the **cs140** directory you have already created in the first lab. The reason for this is to keep your files organized.

At the end of this lab class, you will have 1 source code files saved in a folder called lab7 under the cs140 folder. Recall that for the purpose of evaluating and grading your work by the GA's, you would need to create one script file which holds the source code, compilation result and the output of all programs. The process of creating such a file is described below:

```
>> script lab7.txt
>>  cat  lab7_q.c     // will show and add the source code for the question to the script file
>>cc lab7_q.c   // will compile lab7_q.c
>>./a.out        // will run the program and appends its output to lab7.txt file
>>exit          //closes the script file. Your work is ready to be graded!
```

Submission:

Your lab is graded either at the end of your current lab class (or at the very beginning of your next regular lab class without penalty). Late labs without a valid excuse (eg. illness) receive 0. You are to present your code on the computer in the lab to your lab instructor to receive a grade as follows:

0= not satisfactory or no documentation; 1=incomplete; 2=complete and well documented

Tips

An array is a sequential list. If you wanted to declare space for 10 variables say, instead of declaring 10 individual variables you simply declare one array of size 10, and it will hold all 10 values. e.g. int List[10]; Now how do you access the list? Use the subscript or index, starting from 0. e.g. the third value in the list will be: List[2] = 99; This will store 99 at the third position, index 2. (i.e. index 0 is first, 1 is second, 2 is third; so effectively an array size 10 will be accessible with indices from 0 to 9). The nice thing about automating arrays is to use them with loops (from 0 to 9) to access all the elements of the array!
for (int i=0; i <= 9; i++) List[i] = 99; // this will store 99 in each of the 10 elements of the list.

Lab question

You are to write a single C program that creates a one dimensional array of size MAX (to be declared as a declarative constant using directive `#define MAX 10`), initializes its elements with a given value using the function InitArray, then populate the array with some random numbers between 1 and MAX inclusive, and finally search the array for certain keys.
The structure for this C program is as follows:

```
int main()
{
      //array A to be locally declared here

      int size = MAX;  // note: actual size used can be less than
                       // MAX memory allocation

      // populate the array with a number from the user between 0-10
      InitArray(A, size, GetNumber(0, 10));
      PrintArray(A, size); // to verify the output
      …
      …
      …

      Return 0;
}
```

List of functions:

GetNumber
This function prompts the user to enter an integer value that will be used to fill up the array elements (array initialization). The inputted value will be validated to be within the range of min..max. The valid entry will be returned. (refer to previous labs – reuse the function!)

InitArray
This function uses the value returned from GetNumber to initialize all array elements with. The input parameters of this function are: name of the array, size of the array and the initialization value. It does not return a value.

PrintArray
This is a void function that accepts the name and size of array as two input parameters and prints the contents of the array.

PopulateArray
This function is void return and it fills array elements with random values in the range of 1 .. MAX/2. It has two input parameters which are the name and size of the array.

SearchArray
This function uses a linear search to look for a certain key (value) in the array elements and returns the number of occurrence(s) for that key among the array elements. It returns zero if no matching value was found.

Sample output of the program: MAX=10

```
      Please enter an integer between 0 and 10:  11
      You have entered an invalid entry!
      Please enter an integer between 0 and 10: 7 /*generated by
      GetNumber*/
      /* all elements of array A are initialized to 7 by InitArray */
      A=[7,7,7,7,7]                // generated by PrintArray
      //call PopulateArray
      A=[1,5,5,4,3]               // generated by PrintArray
      // call SearchArray
      Please enter a key between 1 and 5: 4
      No occurrence found for value 4 in the array! /* generated by a
 invoking SearchArray for key=4 */
      Do you want to search for more keys (Y/N)? Y
      Please enter a key between 1 and 5: 5
      2 occurrences found for value 5 in the array! /* generated by a
 invoking SearchArray for key=5 */
      Do you want to search for more keys (Y/N)? N
      Goodbye!
```