Assg 07: Remove Duplicates from Array~

CSci 515 Spring 2015

2015-02-19

Dates:

Due: Tuesday March 24, by Midnight

Objectives

- Practice declaring and using arrays.
- More practice writing functions that take arrays as parameters and process them.
- Look at shifting values in arrays.

Description

In this assignment, we want to solve the following task. We need to design and implement a function that will take an array of integers as input, and will return a new array with only the unique values from the original input array (e.g. a new array without any duplicate items). For example, if we had the following values in an array of size 5:

```
1
3
2
3
2
```

The result would be a new array of size 3, with only the following unique values in the array

```
1
3
2
```

The general idea or algorithm that is easiest to implement to solve this task is as follows. Iterate through each value in the input array. For each value in the input, first see if it already exists in the output array, and if it doesn't then copy it to the output array. Of course, if it already exists in the output array, then you will do nothing.

In order to implement this algorithm, it would be nice to have a function that takes an array as input and searches the array to determine if a value is in the array or not. This function should return true if the value is in the array, and false otherwise. This function should be able to handle an array of size 0, and the answer when searching for a value in an array with nothing yet in it would be false, e.g. the value is not yet present in the array.

Given this search function that can handle searching in an empty array, the implementation of the copying from the input to the output array, checking for duplicates, becomes relatively straightforward. You simply need to start with the output array being empty (with a size of 0). For every value in the input array, you simply perform the search, and if it is not in the output array, you put it on the end of the array, and increase the size of the array by 1. The result, after iterating through the whole input array, should be to only copy the values the first time they are seen to the output array.

Perform the following tasks:

1. Write a function called `searchForValue`. This function takes an array of ints and the size of the array as its parameters. This function also takes a third parameter, an int, which is the value to search for. This function returns a `bool` result of true if the value being searched for is in the given array, and false if the value is not in the array. If you pass in an empty array (e.g. an array of size 0), the function should still work. If the array is empty then of course the answer is false, since there are no values in the array and so there is no possibility that you can find the value that is being asked to search for.

2. Write a function called `findUniqueValues`. This functions takes an input array of int values, and the size of the array as its first two parameters. You should also pass another array of int values as a third parameter. This array should be allocated to be the same size as the input array, since potentially in the worst case, if there are no duplicates, this array will end up being simply a copy of the original

array. This function should copy over the values from the input array into the output array, but as described above it should only copy over unique values (duplicates will not be copied to the output array). The resulting size of the array can be smaller than the input array. Thus this function should return the size of the output array as its result e.g. the size of the array after removing all duplicates.

3. In your main function, declare an array of integers called `values` of size 10. Initialize all of the values in this array to random integers in the range from 0 to 4. Declare another array called `uniques` of 10 integers. Use your `removeDuplicates` functions to remove any duplicate values, and remember to get the new size of the `uniques` array from the function, after duplicates have been removed.

4. Display the `values` array and the `uniques` arrays to standard output. An example of the output you should display is given below.

Your program output should look exactly like the following when I run your program.

```
----- Values array (with duplicates):
[000]     3
[001]     4
[002]     4
[003]     4
[004]     1
[005]     0
[006]     0
[007]     2
[008]     1
[009]     3

----- Uniques array (duplicates removed):
[000]     3
[001]     4
[002]     1
[003]     0
[004]     2
```

NOTE: Now that our programs have more functions than just the `main()` function, the use of the function headers becomes meaningful and required. Make sure that all of your functions have function headers preceding

them that document the purpose of the functions, and the input parameters and return value of the function.

Assignment Submission

An eCollege dropbox has been created for this assignment. You should upload your version of the assignment to the eCollege dropbox named `Assg 07 Remove Duplicates` created for this submission. Work submitted by the due date will be considered for evaluation.

Requirements and Grading Rubrics

Program Execution, Output and Functional Requirements

1. Your program must compile, run and produce some sort of output to be graded. 0 if not satisfied.

2. 35+ pts. Your implementation of the `searchForValue` function must be correct and must use the correct parameters and return the correct return type as specified above for the assignment.

3. 45+ pts. Your implementation of the `removeDuplicates` function must be correct. The function should take the stated parameters as input. The function must return an integer value, the size of the output array after duplicates were found and removed. The function must perform its task correctly.

4. 20+ pts. You should create the `values` and `uniques` arrays in your `main()` function as specified. Your `values` array should be initialized with random values. Your output for your program should look exactly as shown in the example output.

Program Style

Your programs must conform to the style and formatting guidelines given for this course. The following is a list of the guidelines that are required for the assignment to be submitted this week.

1. The file header for the file with your name and program information and the function header for your main function must be present, and filled out correctly.

2. A function header must be present for all functions you define. You must document the purpose, input parameters and return values of all functions. Your function headers must be formatted exactly as shown in the style guidelines for the class.

3. You must indent your code correctly and have no embedded tabs in your source code. (Don't forget about the Visual Studio Format Selection command).

4. You must not have any statements that are hacks in order to keep your terminal from closing when your program exits (e.g. no calls to system()).

5. You must have a single space before and after each binary operator.

6. You must have a single blank line after the end of your declaration of variables at the top of a function, before the first code statement.

7. You must have a single blank space after , and ; operators used as a separator in lists of variables, parameters or other control structures.

8. You must have opening `{` and closing `}` for control statement blocks on their own line, indented correctly for the level of the control statement block.

9. All control statement blocks (if, for, while, etc.) must have `{ }` enclosing them, even when they are not strictly necessary (when there is only 1 statement in the block).

10. You should attempt to use meaningful variable and function names in your program, for program clarity. Of course, when required, you must name functions, parameters and variables as specified in the assignments. Variable and function names must conform to correct `camelCaseNameingConvention` .

Failure to conform to any of these formatting and programming practice guidelines for this assignment will result in at least 1/3 of the points (33) for the assignment being removed for each guideline that is not followed (up to 3 before getting a 0 for the assignment). Failure to follow other class/textbook programming guidelines may result in a loss of points, especially for those programming practices given in our Deitel textbook that have been in our required reading so far.