Assg 08: Selection Sort

CSci 515 Spring 2015

2015-02-19

Dates:

Due: Tuesday March 31, by Midnight

Objectives

- Implement a sorting algorithm
- Practice using arrays.
- Implement and use a swap function
- More practice with passing arrays to functions.

Description

A selection sort searches an array looking for the smallest element. Then the smallest element is swapped with the first element of the array. The process is repeated for the subarray beginning with the second element of the array on the second pass. Each pass of the array results in one element being swapped into its proper location. This algorithm is also an $O(n^2)$ algorithm, for an array of n elements, $n-1$ passes must be made, and for each subarray $n-1$ comparisons must be made to find the smallest value.

In this assignment you are to implement the selection sort algorithm. To support the selection sort, you will also (re)implement two helper functions that work on arrays (and which we have seen examples of before in class). In particular, you need to create a function that, given an array and two indexes will swap the values between the two indexes. And you need to implement a function that will search an array, from some beginning start position, and return the location of the minimum value found in the array

after that position. With these two functions implemented, you should be able to write the selection sort function that calls these two helper functions, in order to find the minimum value and then swap it into place.

Perform the following tasks:

1. Write a function called `findMinimumInSubarray`. This function takes an array of ints as its first parameter and an integer parameter indicating the size of the array. The third parameter is an integer position which indicates the location to start searching within the array. This function should return a single integer. The returned value should be the index location of the minimum value in the subarray that was searched. You will use this location in your sorting function to swap the minimum value into its proper place.

2. Write a function called `swapArrayLocations`. This functions takes an array of ints. It should then take two integer parameters, which represent two (valid) indexes into the array. The function should swap the values in the two indicated locations as a result of calling the function. This function does not return an explicit result (it is a void function) but as a side effect, the indicated locations will be swapped after this function is called.

3. Write an implementation of the selection sorting algorithm. Name your function `selectionSort`. This function takes an array of integers and the size of the array as its first two parameters. This function is a void function. This function should implement the selection sort algorithm as described above. The result of calling this function is that the array of integers should be sorted in ascending order after calling the function.

4. In your `main` function, create an array of size 20. Initialize the array with value in the range from 1 to 100 (inclusive). You may (re)use previous functions from class we have created to initialize an array of random values for this task. Call your sort function to sort the values in the array, and then display the contents of your array on standard output.

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

```
$ ./assg08
```

```
Array, before being sorted:
000:   7
001:  10
002:   3
003:  10
004:  14
005:   1
006:   6
007:   8
008:  20
009:   8
010:   6
011:  10
012:  19
013:  16
014:  16
015:  17
016:  16
017:   1
018:  13
019:  19

Array, after being sorted sorted:
000:   1
001:   1
002:   3
003:   6
004:   6
005:   7
006:   8
007:   8
008:  10
009:  10
010:  10
011:  13
012:  14
013:  16
014:  16
015:  16
016:  17
```

```
017:  19
018:  19
019:  20
```

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 08 Selection Sort` 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. 50+ pts. Your implementation of the two helper functions must be correct, and the functions must work as described. This is half of your grade, so if you are having trouble getting the sort to work, make sure you at least have these simple helper functions written and working correctly.

3. 40+ pts. Your selection sort implementation must work. The selection sort algorithm must be implemented as described in our assignment description.

4. 10+ pts. Your main function should create and initialize the desired array of random integers as described. After calling your sort function, your program should display the contents of the array of integers to standard 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.