

Lab 3.1 Coding with Parallel Arrays

The selection sort algorithm rearranges a list by selecting an element in the list and moving it to its proper position. For example, first the smallest item in the list is moved to the first location. Starting with the second item in the list, the next smallest item is moved to the second location. You can also sort by placing the largest value in the first position, the second largest value in the second position, and so on.

Once a list has been sorted—also called ordered—the list can be searched. In a sequential search on an ordered list, once a value in the list is greater than the value being searched for, the search is complete and the item is not found. In a binary search, an item is chosen mid way through the list and compared with the item being searched for. Based on the result of this comparison half of the list can be eliminated and the process repeated until the item is found or the search is unsuccessful.

Objectives

In this lab, you become acquainted with the selection sort. You will also search an ordered list of a particular size for a value using a binary search. Indicate whether the item is found; if it is, indicate the location where the item was found. If the number is not in the search list, return a failure indication (-1).

After completing this lab, you will be able to:

- ▶ Sort an unordered list using the selection sort.
- ▶ Search an ordered list of a given size for a particular value.
- ▶ Report whether an item is found in the list.
- ▶ If the item was found, report the location in the list where it was found.
- ▶ If the item was not found, stop the search and return -1.

Sorting an Array Using Selection Sort

1a. You are given a program, `Prize.cpp`, that will read in a file, called `PrizeList.txt`, into 2 parallel arrays. The first array is an integer, which is a randomly selected prize number. The second array is an array of strings which is an indication of the name of the prize for that number. For your first step, implement a selection sort (you can use the selection sort from the book as an example) that will sort the arrays. The parallel arrays should be sorted by the prize number. Displays the results of your ordered array after you have called the sorting function. For example, if the `PrizeList.txt` contains:

```
381  A New Boat
103  A New Car
768  $10,000
033  A Bahamas Vacation
691  A Caribbean Cruise
```

After sorting the 2 arrays, you should end up displaying:

033 A Bahamas Vacation
103 A New Car
381 A New Boat
691 A Caribbean Cruise
768 \$10,000

- 1b. Enter, compile, link and execute your modified Prize.cpp. Make sure that the program sorts the parallel arrays by the prize number, and that the prizes are still associated with the correct prize number after you sort.

Searching an Array using Binary Search

- 2a. Design a program that simulates a contest for a radio station that awards a prize to the first caller who correctly guesses a number in the list of randomly generated prize numbers. A caller can make one guess per call. The contest is held until a number has been guessed or the user enters a value of -1. The valid numbers range from 1 to 10000.
- 2b. Enter, compile, link and execute your new version of Prize.cpp with the radio contest simulation added. Once you are satisfied that you have parts 1 and 2 working, you should upload your Prize.cpp lab program into your Educator student account in an appropriately named folder for Lab 3.

The following is a copy of the screen results that might appear after running your program, depending on the data entered. The input entered by the user is in bold.

This program simulates a radio station that asks the caller to guess a number.

The number is compared against a list of 20 numbers between 1 and 500 inclusive.

The contest is held until a number has been matched or a value of -1 is entered.

A message is displayed containing the winning number, the location in the list of numbers, the number of calls made, and the amount of the prize.

Hello Caller. What number between 1 and 10000 are you guessing? **250**
It took 9 comparisons to check the list.
250 is not in the list. Call again.

Hello Caller. What number between 1 and 10000 are you guessing? **13305**
It took 0 comparisons to check the list.
Your guess must be between 1 and 10000 inclusively.

Hello Caller. What number between 1 and 10000 are you guessing? **9238**
It took 8 comparisons to check the list.
9238 is not in the list. Call again.

Hello Caller. What number between 1 and 10000 are you guessing? **4831**
It took 5 comparisons to check the list.
4831 is not in the list. Call again.

Hello Caller. What number between 1 and 10000 are you guessing? **103**
It took 9 comparisons to check the list.
Caller. Your number 103 was found at location 1 of the list
Counting you, there were 4 callers. Your win A New Car.

Lab 3 Finished

You have now completed Lab 3. Turn in the previous pages to the instructor and make sure you have uploaded your **Prize.cpp** and programs to your Educator account in your Lab-3 subfolder. Ask the instructor to examine your lab, which will be given either an acceptable or unacceptable evaluation. If unacceptable, you will be told of the problem areas and you may fix them and resubmit.

Attached at the end of this lab is a description of your third programming assignment, which will be due next Friday Oct 21. After completing this lab, now is a good time to begin thinking about how you will implement the second programming assignment.