

55 pts

Name1: CarlosName2: AguileraClass Day / Time: M/W 7:30pmDue Date: Feb. 28 22

Lab #5: Binary Search

In this lab, you will perform the tasks bellow. DO NOT USE GLOBAL CONSTANTS!

1. Create a header file that contains the following.
 - All necessary pre-processor directives
 - The prototype for a function that sorts an array using an insertion sort.
 - The prototype for a function that searches an array using a sequential search and returns the appropriate index in the array.
 - The prototype for a function that searches an array using a binary search and returns the appropriate index in the array.
 - The prototype for a function that outputs an array.
2. Create your source files as follows:
 - Create a source file that contains the code for the search functions.
 - Create a source file that contains the code for the sort function.
 - Create a source file that contains the code for the output function.
3. Create a file that contains the main function which should perform the following tasks in order.
 - Call the output function.
 - Allow the user to input a key
 - Call the function that performs a sequential search 4 times.
Output the index # that represents where the item was found.
 - Call the function that performs the insertion sort.
 - Call the output function.
 - Call the function that performs the binary search 4 times.
Output the index # that represents where the item was found.

Use the following Array:

```
int intArray[8] = {4, 1, 7, 12, 8, 13, 9, 21};
```

Turn in as a single PDF file (IN THIS ORDER)

- 1 – The first page of this lab (fill in the information on the top right)
- 2 – Program output (cut and pasted into a text file within eclipse)
- 3 – Header file
- 4 – Main.cpp
- 5 – Search functions source file, sort function source file and output source file

```
1 *****
2 *   PROGRAMMED BY : Carlos Aguilera
3 *   CLASS         : CS1B
4 *   SECTION       : MW: 7:30p - 9:50p
5 *   LAB #1        :
6 *****
7
8 Index #0: 4
9 Index #1: 1
10 Index #2: 7
11 Index #3: 12
12 Index #4: 8
13 Index #5: 13
14 Index #6: 9
15 Index #7: 21
16
17 Enter an integer to search for: 9
18 The integer 9 was found in index #6.
19
20 Enter an integer to search for: 6
21 6 was not found!
22
23 Enter an integer to search for: 21
24 The integer 21 was found in index #7.
25
26 Enter an integer to search for: 4
27 The integer 4 was found in index #0.
28
29
30 Performing insertion sort...
31
32 Index #0: 1
33 Index #1: 4
34 Index #2: 7
35 Index #3: 8
36 Index #4: 9
37 Index #5: 12
38 Index #6: 13
39 Index #7: 21
40
41 Enter an integer to search for: 12
42 The integer 12 was found in index #5.
43
44 Enter an integer to search for: 21
45 The integer 21 was found in index #7.
46
47 Enter an integer to search for: 2
48 2 was not found!
49
50 Enter an integer to search for: 1
51 The integer 1 was found in index #0.
```

```
1 #pragma once
2 #include <iostream>
3 #include <iomanip>
4
5 void displayHeader();//display header
6 void initArray(const size_t AR_SIZE, int intArray[]);//initialize the array by
  user input
7 int sequentialSearch(const int &searchKey, const size_t &AR_SIZE,int
  intArray[]);//search for key in array (linear)
8 void handleOutput(const size_t &AR_SIZE, int intArray[]);//display output
9 void insertionSort(const size_t &AR_SIZE, int intArray[]);//sort array using
  insertion
10 int binarySearch(const int &searchKey, const size_t AR_SIZE, int
  intArray[]);//search for key in array (logarithmic)
```

```

1  /*****
2  * AUTHOR      : Carlos Aguilera
3  * STUDENT ID   : 1152562
4  * LAB #5      : Binary Search
5  * CLASS       : CS1B
6  * SECTION     : M-W
7  * DUE DATE    : 02.28.22
8  *****/
9
10 #include "../include/main.h"
11
12 /*****
13 * Title: Binary Search
14 * -----
15 * This program will use sequential search and binary search to find a
16 * key that the user wants to find and output what index the key was found
17 * in.
18 * -----
19 * Data Table
20 * -----
21 * const size_t AR_SIZE {8} CALC - used to calc how big the array is
22 * int intArray [AR_SIZE] IN & OUT - array with values from user
23 * int searchKey {} IN & OUT & CALC - search key to find in the array
24 *****/
25 int main()
26 {
27     displayHeader();
28     const size_t AR_SIZE {8};
29     int intArray[AR_SIZE];
30     initArray(AR_SIZE, intArray);
31     int searchKey {};
32
33     for(size_t i {0}; i < 4; i++)
34     {
35         std::cout << "\nEnter an integer to search for: ";
36         std::cin >> searchKey;
37
38         int index {sequentialSearch(searchKey, AR_SIZE, intArray)};
39         if(index != -1)
40             std::cout << "The integer " << searchKey << " was found in index
41 #\" << index << ".\n";
42         else
43             std::cout << searchKey << " was not found!\n";
44     }
45
46     std::cout << "\n\nPerforming insertion sort...\n";
47     insertionSort(AR_SIZE, intArray);
48     handleOutput(AR_SIZE, intArray);
49
50     for(size_t i {0}; i < 4; i++)
51     {
52         std::cout << "\nEnter an integer to search for: ";
53         std::cin >> searchKey;
54
55         int index {binarySearch(searchKey, AR_SIZE, intArray)};
56         if(index != -1)
57             std::cout << "The integer " << searchKey << " was found in index
58 #\" << index << ".\n";
59         else

```

```
58         std::cout << searchKey << " was not found!\n";
59     }
60 }
61
```

```

1 #include "../include/main.h"
2
3 void displayHeader()
4 {
5     /*****
6      * CONSTANTS
7      * -----
8      * OUTPUT - USED FOR CLASS HEADING
9      * -----
10     * PROGRAMMER : Programmer's Name
11     * CLASS      : Student's Course
12     * SECTION    : Class Days and Times
13     * LAB_NUM    : Lab Number (specific to this lab)
14     * LAB_NAME   : Title of the Lab
15     *****/
16     const char PROGRAMMER[] = "Carlos Aguilera";
17     const char CLASS[]      = "CS1B";
18     const char SECTION[]    = "MW: 7:30p - 9:50p";
19     const int LAB_NUM       = 1;
20     const char LAB_NAME[]   = "";
21
22     // (variable declarations go here)
23
24
25     /*****
26      * OUTPUT - Class Heading
27      *****/
28     std::cout << std::left;
29     std::cout << "*****\n";
30     std::cout << "*   PROGRAMMED BY : " << PROGRAMMER << std::endl;
31     std::cout << "*   " << std::setw(14) << "CLASS" << ": " << CLASS <<
std::endl;
32     std::cout << "*   " << std::setw(14) << "SECTION" << ": " << SECTION <<
std::endl;
33     std::cout << "*   LAB #" << std::setw(9) << LAB_NUM << ": " << LAB_NAME <<
std::endl;
34     std::cout << "*****\n\n";
35     std::cout << std::right;
36 }

```

```

1 #include "../include/main.h"
2 /*****
3  * Title: initArray
4  * -----
5  * FUNCTION:
6  *   Initializes array with user input
7  * -----
8  * No Data Table
9  * -----
10 *****/
11
12 void initArray(const size_t AR_SIZE, int intArray[])
13 {
14     for(size_t i {0}; i < AR_SIZE; i++)
15     {
16         std::cout << "Index #" << i << ": ";
17         std::cin >> intArray[i];
18     }
19 }

```

```

1 #include "../include/main.h"
2 /*****
3  * Title: sequentialSearch
4  * -----
5  * FUNCTION:
6  * I did not use a for loop in this function because using return would
7  * break logic in the for loop, so I used do while loop to triverse through
8  * the array until key was found
9  * -----
10 * Data Table
11 * -----
12 * bool found {false} CALC - if found false keep searching
13 * int index CALC - counter for index
14 * const int NOT_FOUND {-1} CALC - return value if not found
15 *****/
16
17 int sequentialSearch(const int &searchKey, const size_t &AR_SIZE,int
intArray[])
18 {
19     bool found {false};
20     int index {0};
21     const int NOT_FOUND {-1};
22     do
23     {
24         if(searchKey == intArray[index]) {
25             found = true;
26             return index;
27         }
28         ++index;
29     } while (!found && index < AR_SIZE);
30     return NOT_FOUND;
31 }

```



```

1 #include "../include/main.h"
2 /*****
3  * Title: handleOutput
4  * -----
5  * FUNCTION:
6  *   Handles the output of the array in a certain format
7  * -----
8  * No Data Table
9  * -----
10 *****/
11
12 void handleOutput(const size_t &AR_SIZE, int intArray[])
13 {
14     std::cout << std::endl;
15     for(size_t i {0}; i < AR_SIZE; i++)
16         std::cout << "Index #" << i << ": " << intArray[i] << "\n";
17 }

```

```

1 #include "../include/main.h"
2 /*****
3  * Title: insertionSort
4  * -----
5  * FUNCTION:
6  * insertion sort how this sort works is it goes through the array comparing
7  * last key to the elements and pushing the elements forward if needed
8  * -----
9  * Data Table
10 * -----
11 * int key {intArray[i]} CALC - holds value that for loop is on
12 *****/
13
14 void insertionSort(const size_t &AR_SIZE, int intArray[])
15 {
16     // 4 18 1 3
17     // 1 3 4 18
18
19     for(int i {1}; i < AR_SIZE; i++)//i is 3
20     {
21         int key {intArray[i]};//key is 3
22         int j {i - 1};//j is 0
23
24         while(j >= 0 && intArray[j] > key)// key stays the same through while
        loop, you could also use a decrementing for loop here but it would be
        inefficient
25         {
26             intArray[j + 1] = intArray[j];
27             --j;
28         }
29         intArray[j + 1] = key;
30     }
31 }

```

```

1 #include "../include/main.h"
2 /*****
3  * Title: binarySearch
4  * -----
5  * FUNCTION:
6  * this is my implementation of binary search, after my way I searched up
7  * binary search and saw the recursive version of it and a simpler version
8  * of iterative implementation.
9  * In short words you have a max, mid, and min
10 * First the data set has to be in order
11 * Second check to see if key == max or min
12 * if not then we find out whether key is greater than mid or less than mid
13 * if not found return -1
14 * -----
15 * Data Table
16 * -----
17 * int max {AR_SIZE} CALC - used to find max size for the array
18 * int min {0} CALC - used to find min size of array
19 * int mid {AR_SIZE/2} CALC - used to find the mid of max and min
20 * bool notFound {false} CALC - if not found then set to true else
21 * int invalid {-1} CALC - return -1 if not found
22 *****/
23
24 int binarySearch(const int &searchKey, const size_t AR_SIZE, int intArray[])
25 {
26     int max {static_cast <int> (AR_SIZE)};
27     int min {0};
28     int mid {static_cast <int> (AR_SIZE)/2};
29     bool notFound {false};
30     int invalid {-1};
31
32     if(searchKey == intArray[max])
33         return max;
34     else if(searchKey == intArray[min])
35         return min;
36
37     do
38     {
39         if(searchKey >= intArray[mid])
40         {
41             if(searchKey == intArray[mid])
42                 return mid;
43             else {
44                 min = mid;
45                 mid = (max + min)/2;
46             }
47         }else if(searchKey <= intArray[mid]) {
48             if(searchKey == intArray[mid])
49                 return mid;
50             else {
51                 max = mid;
52                 mid = (max + min)/2;
53             }
54         }
55         if((max - min) == 1)
56             notFound = true;
57     } while (!notFound);
58     return invalid;
59 }

```