**Project #7**

**Bubble Sorting**

Josue Ponce

12/14/2016

Author’s Note

This project was prepared for CMSC 140 CRN #22669, taught by professor Madhvi Shah

**Table of Contents**

Program #1 Test Plan………….………………………………………………………...2

Screenshot of Running Application………………………………………….…………3

Algorithm…………………….………..........................................................................6

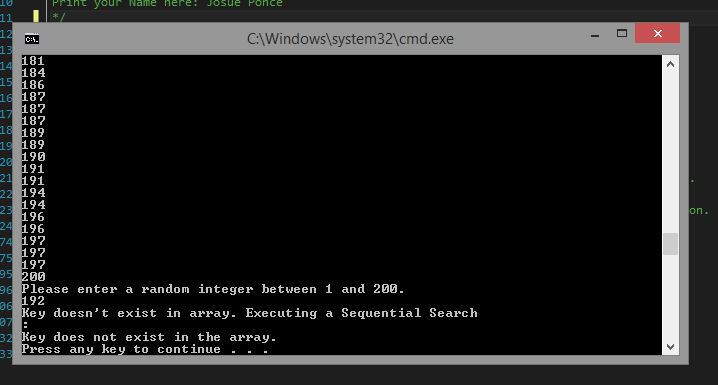
|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Expected Output** | **Actual Output** | **Did Test Pass?** |
| Enter a random integer between 1-200 (inclusive): 192 | Key doesn’t exist in array | Key doesn’t exist in array. | **Y** |
| Enter a random integer between 1-200 (inclusive): 201 | Key doesn’t exist in array. | Key doesn’t exist in array. | **Y** |
| Enter a random integer between 1-200 (inclusive): 192 | Key detected at position 193 by both binary and sequential search. | Key detected at position 193 by both binary and sequential search. | **Y** |
| Enter a random integer between 1-200 (inclusive): 199 | Key detected at position 199 by both binary and sequential search. | Key detected at position 199 by both binary and sequential search. | **Y** |
| Enter a random integer between 1-200 (inclusive): A | Key doesn’t exist in array. | Key doesn’t exist in array. | **Y** |

**Test Plan**

*Note.* Five tests were conducted to ensure that the program worked as intended. During the first test, the number 192 was put into the program which was a number that didn’t exist within the array. The program displayed “Key doesn’t exist in array” which terminated the program since the number could not be found even after a sequential search was executed. The second test was to ensure that the user could only input a number between 1 and 200. Test 3 and test 4 were conducted to ensure that the program displayed the desired results to the user. The input numbers were found within the array and the program displayed both binary search and sequential search results. Test 5 was conducted to ensure that only numbers could be put into the program. The letter A was put into the program which displayed “Key doesn’t exist in array”. Thus, all the tests conducted for this program passed because no flaws were found and the program worked as intended.

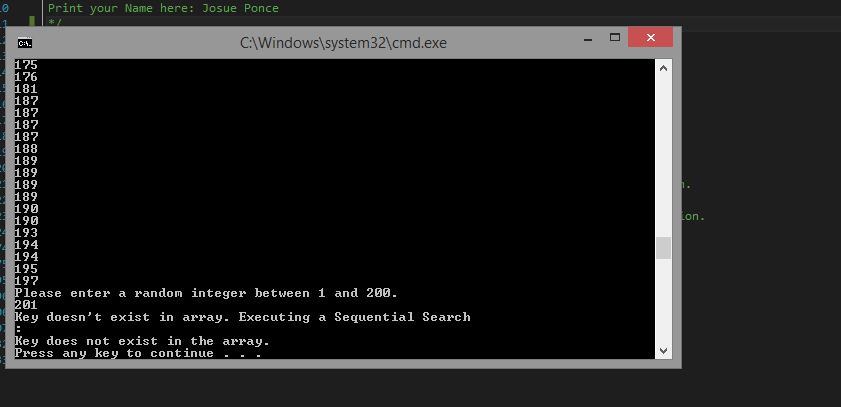
**Screenshots of Running Application**

*Screenshot 1.*



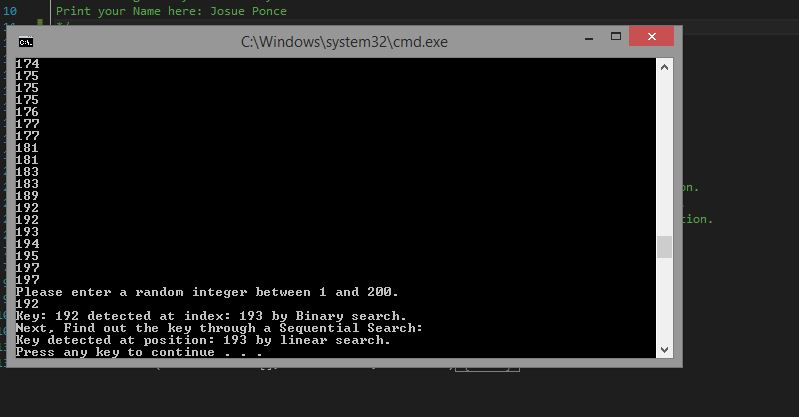
*Note.* Screenshot of running application after test one was conducted.

*Screenshot 2.*



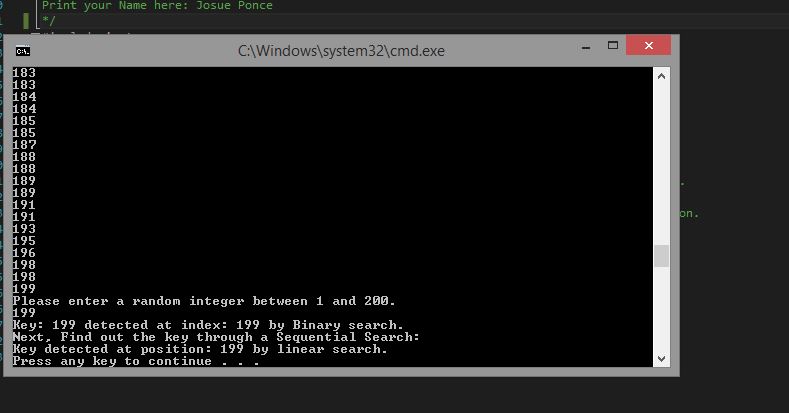
*Note.* Screenshot of running application after test two was conducted.

*Screenshot 3.*



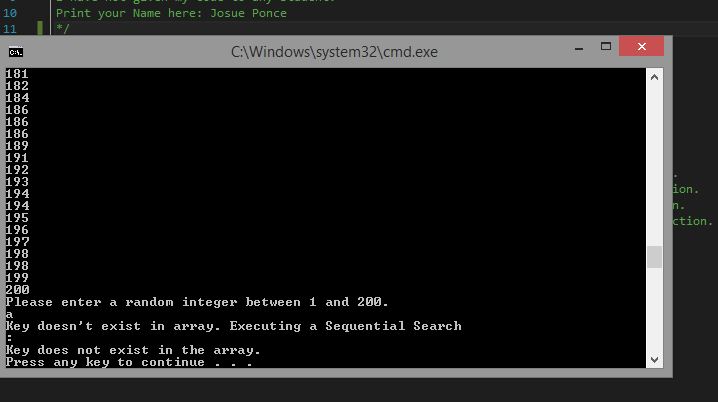
*Note.* Screenshot of running application displaying proper results after test three.

*Screenshot 4.*



*Note*. Screenshot of running application displaying proper results after test four.

*Screenshot 5.*



*Note.* Screenshot of running application after test five.

**Algorithm**

1. /\* Class: CMSC140 CRN #22669
2. \* Instructor: Madhvi Shah
3. \* Student Name: Josue Ponce
4. \* Project #7
5. \* Description: Program generates 200 random integers between 1 to 200. The program then sorts these integers for the user.
6. The user enters a number and the program will determine where the number is located within the sorted array by executing a binary and sequential search.
7. \* Due Date: 12/15/2016
8. \* I pledge that I have completed the programming assignment independently.
9. I have not copied the code from a student or any source.
10. I have not given my code to any student.
11. Print your Name here: Josue Ponce
12. \*/
13. #include<iostream>
14. #include<fstream> // open read and write
15. #include<cstdlib> // rand and srand
16. #include<iomanip>
17. #include<ctime>  // For time function.
18. #include<string>
19. **using** **namespace** std;
21. **void** sortArray(**int** list[], **int** size);                           // Sort numbers function.
22. **void** showArray(**const** **int** list[], **int** size);                     // Display numbers function.
23. **int** binarySearch(**const** **int** list[], **int** elements, **int** value);    // Binary Search Function.
24. **int** searchList(**const** **int** list[], **int** elements, **int** value);      // Sequential search function.
25. **int** main() {
27. // Declaring variables.
28. **const** **int**  MAX\_VALUE = 200;         // Constant value of 200 for arrays.
29. **int** value, hold = 0, index, meow;  // To hold numbers and functions.
30. **int** neko[MAX\_VALUE];
31. **int** num[MAX\_VALUE];
32. ofstream outFile;   // Create output Files.

35. srand(time(0)); // Generates random numbers.
37. **for** (**int** i = 0; i < MAX\_VALUE; i++) {
38. value = rand() % MAX\_VALUE + 1;     // Generates random numbers ranging from 1- 200.
39. num[i] = value;                     // variable will hold 200 random numbers for function.
40. }
41. cout << "The sorted values are:\n";       // Displays random 200 numbers.
42. sortArray(num, MAX\_VALUE);              // Calls in function to bubble sort results.
43. showArray(num, MAX\_VALUE);              // Displays sorted results to the user.
44. outFile.open("BinarySearchResults.txt"); // Creates an output in order to save binary search results.
45. cout << "Please enter a random integer between 1 and 200." << endl; // Displays message to user to input a number ranging from 1-200.
46. cin >> hold;                                                      // obtain user input.
47. index = binarySearch(num, MAX\_VALUE, hold);             // variable holds the binary search results.
49. **if** (index == -1) {                                      // if results are equal to -1 the following sets of code will execute.
50. cout << "Key doesn't exist in array. Executing a Sequential Search\n: " << endl;    // tells user key doesn't exist in the array.
51. meow = searchList(num, hold, MAX\_VALUE);                                // Variable with hold sequencial search results.
52. **if** (index == -1) {                                                  // If results are equal to -1 the following lines of code will execute.
53. cout << "Key does not exist in the array." << endl;             // Tells user the key doesn't exist in the array.
54. **return** 0;                                                       // Terminates program.
55. }
56. }
57. **else**                                        // if the desired results are obtained the following lines will execute.
58. cout << "Key: " << hold << " detected at index: " << index << " by Binary search." << endl; // Displays binary search results to the user.
59. outFile << "Key: " << hold << " detected at index: " << index << " by Binary search." << endl; // Saves binary search results to the text file.
60. outFile.close();                                                    // Binary search text file will close.
62. meow = searchList(num, MAX\_VALUE, hold);    // Variable with hold sequencial search results.
63. outFile.open("SequentialSearchResults.txt");    // Opens a new text file to hold sequential search results.
64. cout << "Next, Find out the key through a Sequential Search: " << endl;     // Display message to user telling them
65. cout << "Key detected at position: " << meow << " by linear search." << endl;   // Displays sequential search results to user.
66. outFile << "Key detected at position: " << meow << " by linear search." << endl; // Saves sequential search results to a text file.
67. outFile.close();                                                                // Closes sequential search file.





74. **return** 0;
75. }
77. **void** sortArray(**int** list[], **int** size) // Function will perform an ascending order bubble sort.
78. {
79. **bool** swap;
80. **int** temp;           //Holding variable.
82. **do**
83. {
84. swap = **false**;                       // Indicates that a swap has occured.
85. **for** (**int** count = 0; count < (size - 1); count++)
86. {
87. **if** (list[count] > list[count + 1])
88. {
89. temp = list[count];           // Swapping elements.
90. list[count] = list[count + 1];
91. list[count + 1] = temp;
92. swap = **true**;                // Indicates that a swap has occured.
93. }
94. }
95. } **while** (swap);
96. }
98. **void** showArray(**const** **int** list[], **int** size)
99. {
100. ofstream outputFile;                // Creating an output file.
101. outputFile.open("sorted.txt");      // Opens output File.
102. **for** (**int** count = 0; count < size; count++) {
103. outputFile << list[count] << " " << endl; // Sends sorted results to the output file.
104. cout << list[count] << " " << endl;           // Displays sorted results to the user.
105. }
106. outputFile.close();                             // Closes outputFile.
107. }
109. **int** binarySearch(**const** **int** list[], **int** elements, **int** value)
110. {
111. **int** first = 0,              // First element.
112. last = elements - 1,    // Last number in element.
113. middle,                 // Mid point search.
114. position = -1;          // Position of the value.
115. **bool** found = **false**;         // flag.

118. **while** (!found && first <= last)
119. {
120. middle = (first + last) / 2;
121. **if** (list[middle] == value)
122. {
123. found = **true**;               // Value found at midpoint.
124. position = middle;          // Calculate mid point.
125. }
126. **else** **if** (list[middle] > value) // if value is lower half.
127. last = middle - 1;
129. **else**
130. first = middle + 1;     // if value is upper half.
131. }
132. **return** position;
133. }
135. **int** searchList(**const** **int** list[], **int** elements, **int** value)
136. {
137. **int** index = 0;       // Used for subcript in order to search the array.
138. **int** position = -1;   // To record the position of the value.
139. **bool** found = **false**;  // Flag will determine if the number was found.
141. **while** (index < elements && !found)
142. {
143. **if** (list[index] == value)  // If the number is found.
144. {
145. found = **true**;           // Flag
146. position = index;       //Record the number's position or -1;
148. }
149. index++;      // Goes to the next element.
151. }
152. **return** position;              // Returns the value's position or -1.
153. }