Lab 8

## LAB 8.1 Working with Linear Search

Retrieve program linear\_search.cpp from the Lab 9 folder. The code is as follows:

// This program performs a linear search on a character array

// Place Your Name Here

#include <iostream>

using namespace std;

int searchList(char[], int, char); // function prototype

const int SIZE = 8;

int main()

{

char word[SIZE] = "Harpoon";

int found;

char ch;

cout << "Enter a letter to search for:" << endl;

cin >> ch;

found = searchList(word, SIZE, ch);

if (found == -1)

cout << "The letter " << ch << " was not found in the list" << endl;

else

cout << "The letter " << ch << " is in the " << found + 1 << " position of the

list" << endl;

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// searchList

//

// task: This searches an array for a particular value

// data in: List of values in an array, the number of

// elements in the array, and the value searched for

// in the array

// data returned: Position in the array of the value or -1 if value

// not found

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int searchList(char List[], int numElems, char value)

{

for (int count = 0; count <= numElems; count++)

{

if (List[count] == value)

// each array entry is checked to see if it contains

// the desired value.

return count;

// if the desired value is found, the array subscript

// count is returned to indicate the location in the array

}

return -1; // if the value is not found, -1 is returned

}

Exercise 1

Re-write this program so that it searches an array of integers rather than characters. Search the integer array nums[8] =



for several different integers. Make sure you try integers that are in the array and others that are not. What happens if you search for 5?

Exercise 2

Re-write the program so that the user can continue to input values that will be searched for, until a sentinel value is entered to end the program. Should a pre- or post-test loop be used?

**LAB 8.2 Working with the Binary Search**

Retrieve program binary\_search.cpp from the Lab 9 folder. The code is as follows:

// This program demonstrates a Binary Search

// PLACE YOUR NAME HERE

#include <iostream>

using namespace std;

int binarySearch(int[], int, int); // function prototype

const int SIZE = 16;

int main()

{

int found, value;

int array[] = { 34, 19, 19, 18, 17, 13, 12, 12, 12, 11, 9, 5, 3, 2, 2, 0 };

// array to be searched

cout << "Enter an integer to search for:" << endl;

cin >> value;

found = binarySearch(array, SIZE, value);

// function call to perform the binary search

// on array looking for an occurrence of value

if (found == -1)

cout << "The value " << value << " is not in the list" << endl;

else

cout << "The value " << value << " is in position number "

<< found + 1 << " of the list" << endl;

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// binarySearch

//

// task: This searches an array for a particular value

// data in: List of values in an ordered array, the number of

// elements in the array, and the value searched for

// in the array

// data returned: Position in the array of the value or -1 if value

// not found

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int binarySearch(int array[], int numElems, int value) // function heading

{

int first = 0; // First element of list

int last = numElems - 1; // last element of the list

int middle; // variable containing the current

// middle value of the list

while (first <= last)

{

middle = first + (last - first) / 2;

if (array[middle] == value)

return middle; // if value is in the middle, we are done

else if (array[middle]<value)

last = middle - 1; // toss out the second remaining half of

else

first = middle + 1; // toss out the first remaining half of

// the array and search the second

}

return -1; // indicates that value is not in the array

}

Exercise 1

Search the array in the program above for 19 and then 12. Record what the output is in each case.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Note that both 19 and 12 are repeated in the array. Which occurrence of 19 did the search find?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Which occurrence of 12 did the search find?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Explain the difference.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Exercise 2

Modify the program to search an array that is in ascending order.

Make sure to alter the array initialization.

**LAB 8.3 Working with Sorts**

Retrieve program bubble\_sort.cpp or selection\_sort.cpp from the Lab 9 folder.

Exercise 1

Re-write the sort program you chose so that it orders integers from largest to smallest rather than smallest to largest.

Exercise 2

Modify your program from Exercise 1 so that it prints the array **at each step** of the algorithm. Try sorting the array



By hand using whichever algorithm you chose. Then have your program do the sorting.

Does the output match (the steps) that you did by hand?

**LAB 8.4 Searching Through Text**

Retrieve program searching\_words.cpp from the Lab 9 folder. The code is as follows:

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

**// 4a. ADD THE APPROPRIATE PROTOTYPE for a modified searchList function from Lab 8.1**

**// that searches for a string value within a string array**

int main() {

// Character arrays can take in an entire string of characters.

ifstream inputfile;

string story[27000]; // This should be enough...

int count = 0;

// open file

inputfile.open("alice\_in\_wonderland.txt");

**// 1. FILL IN CODE to read all words (without spaces and line breaks)**

**// into the string array story**

cout << "There are " << count << " number of words in this story." << endl << endl;

// close file

inputfile.close();

cout << "Printing an excerpt...." << endl;

**// 2. Show an excerpt of the story by PRINTING THE FIRST**

**// 100 WORDS OF THE STORY. Use a loop here to achieve that.**

cout << endl << endl;

string searchWord = "cats";

int wordloc;

**// 3. MAKE A FUNCTION CALL ON THE searchList FUNCTION**

**// to search the story for the given searchWord. RETURN THE**

**// FOUND LOCATION to variable wordloc**

cout << "The first occurrence of '" << searchWord << "' at word #" << wordloc << endl;

// prints part of the phrase containing the searchWord, for some context

cout << "...";

for (int i=wordloc; i<wordloc+10; i++)

cout << story[i] << " ";

cout << "..." << endl;

}

**// 4b. WRITE THE DEFINITION for the modified searchList function used earlier in Lab 8.1**

**// to accommodate a string array instead of a character array**

**//**

**//**

Exercise 1

Follow through the steps indicated by the numbers in comments (1, 2, 3, 4a, 4b) to complete the program so that it produces the expected output below:

Sample Run:

**There are 26448 number of words in this story.**

**Printing an excerpt....**

**Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, `and what is the use of a book,' thought Alice `without pictures or conversation?' So she was considering in her own mind (as well as she could, for the hot day made her feel very sleepy and stupid), whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies,**

**The first occurrence of 'cats' at word #833**

**... cats eat bats, I wonder?' And here Alice began to ...**

Whe

Exercise 2

If the word cannot be found (for example try, the symbol “#” or “refrigerator”), display a different message:

**The word ‘refrigerator’ does not exist in the story.**

Exercise 3

Where are the first occurrences for the following words?

* simple : \_\_\_\_\_\_\_\_
* green : \_\_\_\_\_\_\_\_
* ready : \_\_\_\_\_\_\_\_

To test for so many sample words, make your program friendlier by allowing the user to continue entering search words until a sentinel string/symbol (e.g. # is a good candidate) is entered to terminate the program.

**LAB 8.5 Sorting with std::sort() from C++ STL**

Retrieve program sorting\_authors.cpp from the Lab 9 folder. The code is as follows:

#include <iostream>

#include <fstream>

#include <vector>

#include <string>

using namespace std;

void printVec(const vector<string>);

int main() {

ifstream inputfile;

string wisewords;

int dashloc;

vector<string> authors;

// open file

inputfile.open("quotes.dat");

// loop through each row of quotes, and proceed to get each line as a string

while (getline(inputfile, wisewords)) {

// 1. CALL string.find() function to search for a specific string from

// each line of quote, and store the location in variable dashloc. Look up

// the C++ reference for strings to see in detail how to use this function.

// Notice that the author of the quote always appears after a double dash.

// Extract the substring containing the author name using the found location

// of the double dash.

string author;

author = wisewords.substr(dashloc+2, string::npos);

// 2. ADD the author name to the vector authors

}

// close file

inputfile.close();

cout << "Author List: " << endl;

printVec(authors);

// 4. CALL sort function (you need to include <algorithm>) to sort the

// vector of authors. Look up the C++ reference on how to use this function

cout << "Sorted List:" << endl;

printVec(authors);

}

// 3. COMPLETE the function printVec that takes in a vector of strings as input

// and prints out all its contents, one line each. Use a range for-loop to

// achieve this.

void printVec(const vector<string> line) {

}

Exercise 1

Follow through the steps indicated by the numbers in comments (1, 2, 3, 4) to complete the program so that it produces the expected output below. The program processes a data file (quotes.dat) containing some quotes from famous people. The program performs:

* Searching (look in the entire line of quote for double dashes), and
* Sorting (order the vector of author names)

The double dashes indicate the location where you can extract the author name, which is a sub-string of the original quote.

Sample Run:

**Author List:**

**Pythagoras**

**Nelson Mandela**

**John Lennon**

**Benjamin Franklin**

**Mother Teresa**

**Confucius**

**Thomas A. Edison**

**Steve Jobs**

**Alexander Graham Bell**

**Bjarne Stroustrup**

**Sorted List:**

**Alexander Graham Bell**

**Benjamin Franklin**

**Bjarne Stroustrup**

**Confucius**

**John Lennon**

**Mother Teresa**

**Nelson Mandela**

**Pythagoras**

**Steve Jobs**

**Thomas A. Edison**

Exercise 2

Find out (from C++ STL reference) how to use the sort() function to sort the vector of author names in descending order.

**LAB 8.6 Student Generated Code Assignment**

Write a program that prompts the user to enter the number of elements and the numbers themselves to be placed in an integer array that holds a maximum of 50 elements. The program should then prompt the user for an integer which will be searched for in the array using a binary search. Make sure to include the following steps along the way:

1. A sort routine must be called before the binary search. You may use either the selection sort or the bubble sort. However, the sort must be implemented in its own function and not in main.
2. Next include a function called by main to implement the binary search. The ordered array produced by the sort should be passed to the search routine which returns the location in the sorted array of the sought value, or -1 if the value is not in the array.
3. Add a value returning function that computes the **mean** of your data set. Recall that the mean is the sum of the data values divided by the number of pieces of data. Your program should output the size of the array entered, the array as entered by the user, the sorted array, the integer being searched for, the location of that integer in the sorted array (or an appropriate message if it is not in the array), and the mean of the data set.
4. (Optional) Modify your program so that the data is entered from a file rather than from the keyboard. The first line of the file should be the size of the integer array. The second line should contain the integer searched for in the data set. Finally, the array elements are to start on the third line. Make sure you separate each array element with a space. The output, as described in *iii*), should be sent to a file.