**Programming Fundamentals I**

**COSC1436**

**Lab 8. Searching and Sorting[[1]](#footnote-1)**

**Objectives:**

1. to learn to search array based lists using the linear and binary search algorithms
2. to learn to sort array based lists using sorting algorithms (selection sort)

**Task 1: Working with Linear search**

1. Create a new C++ project in Visual Studio, add a new C++ file and copy the following code into it:

// 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;

system("pause");

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

}

1. Build and run the program for various character values.

1. Re-write this program so that it searches an array of integers rather than characters. Search the integer array numbers[16] = {34,19,19,18,17,13,12,12,12,11,9,5,3,2,2,0};
2. Test your program with various test values. Make sure you try integers that are in the array and others that are not. What happens when you search for 19 and for 12?
3. Add code to output the number of lookups (i.e. search comparisons) the program performs. Show the results for the following sample test data: 19, 13, 2, 55.

**Demo Exercise 5 and show results to your instructor**

**Task 2: Working with Binary Search**

1. Create a new C++ project in Visual Studio, add a new C++ file and copy the following code into it:

// 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;

}

system ("pause");

return 0;

}

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

// binarySearch

//

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

// data in: List of values in an orderd 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 half of the array

//and search the first

else

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

//and search the second

}

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

}

1. Test your program with various test values. Make sure you try integers that are in the array and others that are not. What happens when you search for 19 and for 12? Explain the difference.

1. Add code to output the number of lookups (i.e. search comparisons) the program performs. Show the results for the following sample test data: 19, 13, 2, 55. How do they compare to the Linear Search program of the previous task?

**Demo Exercise 3 and show results to your instructor.**

**Task 3: Working with Sorting Algorithms**

1. Create a new C++ project in Visual Studio, add a new C++ file and copy the following code into it:

// This program uses a selection sort to arrange an array of integers in

// ascending order

//PLACE YOUR NAME HERE

#include <iostream>

using namespace std;

// function prototypes

void selectionSortArray(int [], int);

void displayArray(int[], int);

const int SIZE = 5;

int main()

{

int values[SIZE] = {9,2,0,11,5};

cout << "The values before the selection sort is performed are:" << endl;

displayArray(values,SIZE);

selectionSortArray(values,SIZE);

cout << "The values after the selection sort is performed are:" << endl;

displayArray(values,SIZE);

system("pause");

return 0;

}

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

// displayArray

//

// task: to print the array

// data in: the array to be printed, the array size

// data out: none

//

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

void displayArray(int array[], int elems) // function heading

{ // Displays array

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

cout << array[count] << " ";

cout <<endl;

}

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

// selectionSortArray

//

// task: to sort values of an array in ascending order

// data in: the array, the array size

// data out: the sorted array

//

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

void selectionSortArray(int array[], int elems)

{

int seek; // array position currently being put in order

int minCount; // location of smallest value found

int minValue; // holds the smallest value found

for (seek = 0; seek < (elems-1);seek++) // outer loop performs the swap

// and then increments seek

{

minCount = seek;

minValue = array[seek];

for(int index = seek + 1; index < elems; index++)

// inner loop searches through array starting at array[seek] searching

// for the smallest value. When the value is found, the subscript is

// stored in minCount. The value is stored in minValue.

{

if(array[index] < minValue)

{

minValue = array[index];

minCount = index;

}

}

// the following two statements exchange the value of the

// element currently needing the smallest value found in the

// pass(indicated by seek) with the smallest value found

// (located in minValue)

array[minCount] = array[seek];

array[seek] = minValue;

}

}

1. Compile the program and run it.
2. Re-write the sort program so that it orders integers from largest to smallest rather than smallest to largest.
3. Rewrite the sort program so that it sorts an array of strings instead of the array of integers.

**Demo Exercise 3 and 4 to your instructor.**

1. Based on SOW C++ Lab Manual (by T. Gaddis, Pearson Publishing) [↑](#footnote-ref-1)