

Programming Assignment #3

Application of Arrays: Sorting and Searching

Sorting is the process of taking a bunch of items, all of the same type, and putting them into some defined order. For example, you might sort your CD collection alphabetically by the name of the artist or group. Books in a library are sorted by topic, then author name, e.g. by using the Dewey decimal system, or the newer standard you see in for example our TAMUC library today. We can also talk of sorting items in a computer program. Often we have a collection of items, held in an array, that we want to put into some order. If the items are integers, we might sort them in increasing numerical order, from the smallest item in index 0 to the largest integer in the last index of the array. If the array contains names, we might sort them alphabetically.

A search is a process where we have a collection of items and we want to find out if 1) the item is in the collection, and if it is 2) the location of the item. For example, if we have a grocery list written by someone else for us to use to pick up some items at the grocery store, we might be passing the milk and we would want to search our grocery list to see if milk is on the list of items to buy. For a small grocery list, you might search the list by scanning from the top item on the page, down the page item by item until you find milk on the list (or not as the case may be). This is essentially a serial or sequential search, where we look at items one by one in order from the top of the page to the bottom.

However, imagine you are instead searching for a phone number in the telephone book. This is another type of search we all perform regularly. If you tried to perform you search for a persons name in the telephone book using a sequential search, you would be there all day, starting with the people who's last name begins with 'A', then the 'B's, etc., especially if you looking for Tom Zimmerman. For this type of a search we instead use a bit of knowledge about the phone book and the types of surnames people have. The important bit of knowledge is, that in this case, the collection of items (the people's names along with their phone numbers) is arranged in a special order. By ordering the names alphabetically by the person's last name, we know that Zimmerman, Tom will appear somewhere near the back of the phone book, while Applewhite, Susan will be near the front, and Smith, John more towards the middle.

The way that most people search for a name in a phone book is similar to what is known as a binary search. We open the phone book to some spot, and based on the names at that spot we will either search for the name further on towards the back of the phone book, if the name we are looking for appears alphabetically after the page we have opened the phone book too. Likewise, if the name we are

searching for appears in the alphabet before the names on the current page, we will look further towards the front of the phone book. A Binary search is basically the same as this method, but of course since we need to write it for a computer to perform, we have to be very methodical. So we start by opening the phone book to exactly the middle, and either looking at the front $\frac{1}{2}$ or back $\frac{1}{2}$ respectively based on the name we are searching for. And we repeat this process of eliminating $\frac{1}{2}$ of the remaining names by splitting the unsearched portion of the items in $\frac{1}{2}$ and making a decision.

Objectives

1. Practice more with using functions and arrays, and passing arrays into functions.
2. Become more familiar with sorting algorithms by redoing one of the books sorting algorithms to sort an array of strings alphabetically.
3. Likewise to better understand binary search by rewriting the textbooks binary search to search for a name in the array of strings you sort.

Instructions

Part 1: Download and Run assg3.cpp

Download the assg3.cpp file from the eCollege web site as well as the unsortednames.dat data file. This file contains code to read in a list of names from a file, and to store the names in an array of strings. It also contains a useful function for displaying the names in the array. The unsortednames.dat file contains almost 5000 names, all in random order.

In this first step you should simply confirm that the assg3.cpp will compile and run correctly for you in your IDE. You will want to create a new project and copy the code from the assg3.cpp file you download into a source file in your project. You will also need to copy the unsortednames.dat to an appropriate location. The assg3.cpp source code assumes that the file is in the current directory where the program is running from. This will vary depending on which IDE you use, but will normally be the same location that the source files are created at when you use your IDE to create new source code files.

Once you have the given assg3.cpp code downloaded and running correctly, you are ready to move on to part 2 and begin coding your sort and search.

Part 2: Implement a Sort for an Array of Strings

In this part of the assignment you are to modify one of the textbooks sorting examples to sort an array of strings instead of integers. You may implement either the bubble sort, selection sort, or insertion sort, whichever you prefer.

Using the code for the bubble sort (pg. 559) selection sort (pg. 564) or insertion sort (pg. 571) from the book as an example, implement a sort function that will sort an array of strings, rather than integers. Your function should take 2 parameters, an array of strings containing the names to be sorted, and an integer which indicates the number of names in the list. The function does not return an explicit result, so it is a void function, its results are returned implicitly by returning the array of strings back but now in a sorted order.

You should add the appropriate line to main() to invoke your sort function, and check that it is in fact correctly sorting the array of names by examining the displayed output before and after the sorting.

Part 3: Implement a Binary Search for an Array of Strings

Likewise reimplement the books binary search (pg. 576) to search for a string instead of an int. You should again invoke your binary search appropriately from the main() function. If you like, you could add some code to main to prompt for a name to search for, and even a loop so you can search for multiple names. You should display the index where the name was found, or an appropriate message if the name being searched for was not found in the array.

Assignment 3 Finished

You have now completed Assignment 3. If your program compiles and runs correctly and you have successfully uploaded your source file to the eCollege online submission site, then you are done.