Ace Buenavides

CIS-5

Prof. Conrad

9 MAY 2017

**Array Searching/Sorting Program [v.1.0]**

**Program Requirements**:

1. Write a C++ program that will create an array of 50 integers (**you are not allowed to use vectors**).

2. Using the rand function to generate random numbers between 1 and another number specified by the user (prompt the user for this value).

2. The program must output the array of random integers with a function named **displayList**.

3. Have the program sort the array of random integers from largest to smallest (current discussion on sorting has sorting algorithms sort from smallest to largest, how do we do the opposite? ). Use either bubble sort ( if you really feel up for the out-side-the-box-thinking challenge, try heap sort - this is normally taught in data structures - 1 point extra credit for using merge sort or heap sort [ **pseudocode located in the How To Folder** ] ).

4. The program must output the sorted array of random integers again with a function named **displayList.**

5. Prompt the user to enter a value to search for by using Binary Search (this value may or may not be in the list of random values generated ).

6. Output whether or not Binary Search found the search value and output the index of where the value was located, if it was found, otherwise output that it was not found.

Make sure to use the input validation techniques we have discussed and used (kitty cat proof input).

**Steps:**

1. Include the necessary libraries:
   1. Iostream
   2. Cstdlib
   3. Ctime
   4. Limits
   5. String
2. Declare function prototypes
   1. Kitty Cat Test Function
   2. displayList Function (Outputs array values)
   3. sort Function (Bubble Sort)
   4. binarySearch Function (Binary Search)
3. Main Function:
   1. Program Opening Statement:
      1. Welcome to this array searching/sorting program. This will ask you to input the maximum size for the array and will randomly generate numbers to fill the array. The program will output the array of random integer. It will then sort the array of random integers from largest to smallest using Bubble sort. It will then output the sorted array and prompt you to enter a value to be searched using Binary Search, and output if it was found and output the index of where the value was located.
   2. Declare necessary variables:
      1. cons tint SZ = 50 (Array has to hold 50 numbers)
      2. int maxrange
      3. int searchtarget
      4. int searchresult
   3. Declare array name:
      1. fill\_random[SZ]
   4. Prompt the user for the max range for the numbers to be generated. Ensure to use kitty cat proof input validation.
   5. Random number generation:
      1. Seed the rand function with time(0) + SZ
      2. fill the array with rand function with for loop:
         1. for (int i=0; i < SZ; i++)
            1. fill\_random[i]=(rand()% maxrange)+1;
   6. Output the results of the array with its randomly generated elements.
   7. Sort the array with Bubble Sort:
      1. Compare 1st two elements
         1. If out of order, exchange them to put in order
      2. Move down one element, compare 2nd and 3rd elements, exchange if necessary. Continue until the end of the array.
      3. Pass through the array again, exchanging as needed.
      4. Repeat until the pass made with no exchanges
   8. Output the results of the array, the elements should be sorted from largest to smallest at this point.
   9. Prompt the user for a value that will search the array using Binary Search:
      1. *Set first index to 0.*
      2. *Set last index to the last subscript in the array.*
      3. *Set found to false.*
      4. *Set position to -1.*
      5. *While found is not true and first is less than or equal to last*
      6. *Set middle to the subscript half-way between array[first] and array[last].*
      7. *If array[middle] equals the desired value*
      8. *Set found to true.*
      9. *Set position to middle.*
      10. *Else If array[middle] is greater than the desired value*
      11. *Set last to middle - 1.*
      12. *Else*
      13. *Set first to middle + 1.*
      14. *End If.*
      15. *End While.*
      16. *Return position.*
   10. Output the results of the Binary Search:
       1. If the target was found: Output the value of the element and the index it was found in.
       2. If the target was not found: Output that the search target was not found.
4. Define each necessary function prototypes:
   1. getInteger
   2. displayList
   3. sort
   4. binarySearch
5. End the program.