Lab6

(Link) → cpp.sh/7kade

(Note, I had trouble trying to compile the VM version because of a few errors)

Day 1 progress

```
56          }
57
58
59          else if(option == "c"){
60              sortC();
61          }
62
63      }
64  }
65
66
67  //sort from 1-1000
68   int sortA(){
69      int i;
70     //   std::cout << " 1000 ";
71      for ( i=0; i<1000; i++){
72      a[i] = i;
73      std::cout << a[i] << std::endl;
74      }
75   }
76
77
78      // sort reverse 1000-1
79
80      int reverseSortA(){
81      }
82
83
84
```

Day 2 progress

```
133  std::cout << std::endl;
134  std::cout << std::endl;
135              //   randSortA(a, size);
136                 return 0;
137      }
138
139
140  int randSortA(int a[], int size){
141      std::cout << " First 10 and last 10 pre-sorted random order is... " << std::endl;
142      int n;
143      n = rand() % 1000+1;
144   for (int i=0; i<size-990; i++){
145         n = rand() % 1000+1;
146   a[i] = n;
147   std::cout << a[i] << "\t";
148   }
149   std::cout << std::endl;
150   for (int i=10; i<size-980; i++){
151       n = rand() % 1000 + 1;
152       a[i] = n;
153       std::cout << a[i] << "\t";
154   }
155    std::cout << std::endl;
156   // randAsort(a,size);
157   return 0;
158  }
159
160          //After Pre-sorting all functions, now sort all them in the correct ascending order.
161
```

Short URL: cpp.sh/3h3o

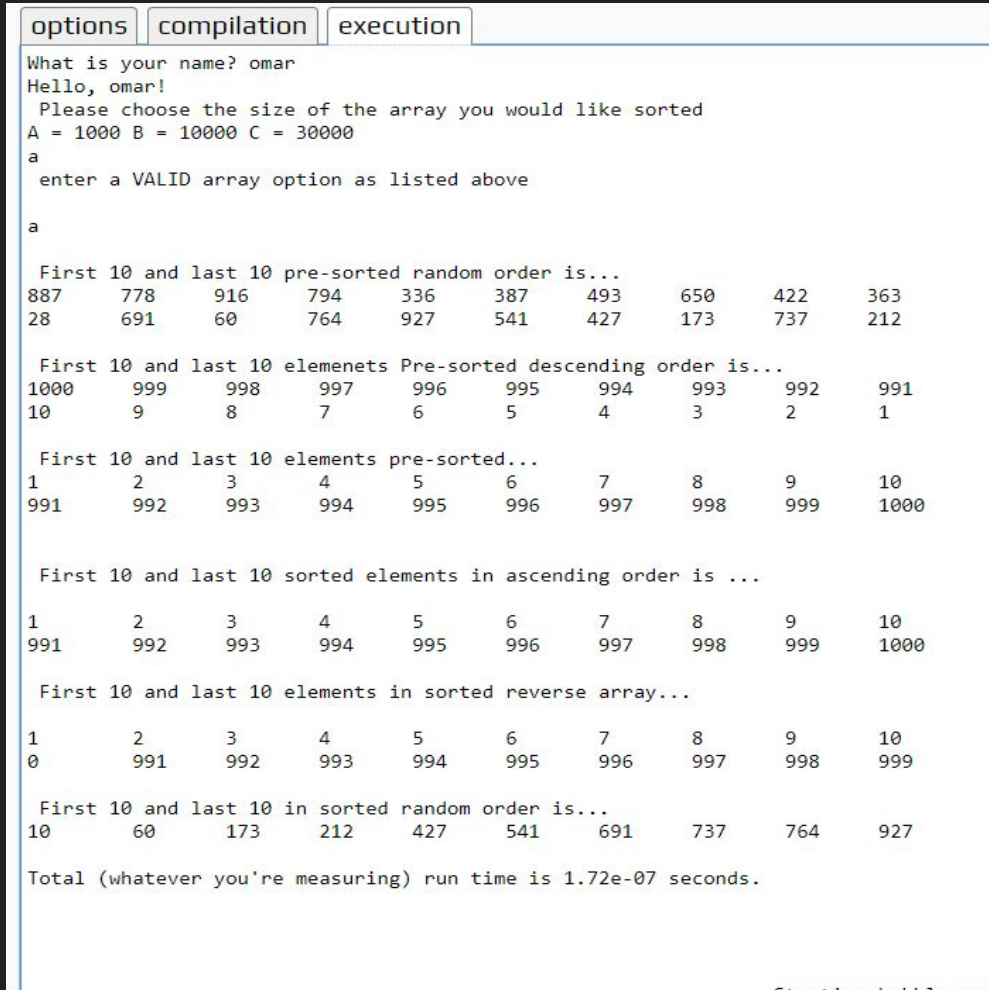
options | compilation | execution

```
What is your name? a
Hello, a!
 Please choose the size of the array you would like sorted
A = 1000 B = 10000 C = 30000
a
 enter a VALID array option as listed above
a

 First 10 and last 10 pre-sorted random order is...
887     778     916     794     336     387     493     650     422     363
28      691     60      764     927     541     427     173     737     212

 First 10 and last 10 elemenets Pre-sorted descending order is...
1000    999     998     997     996     995     994     993     992     991
10      9       8       7       6       5       4       3       2       1
```

Day 3 progress

side comment)

In the 3 days I worked on this project I felt like I was behind. I felt like I had lost everything I learned and everyone else was ahead of me. I spent countless hours on this program and ended up with 1000+ lines of code… I had to set this aside and got help to code it. I still wish I had a better understanding of how parameters work and how functions can be included as a parameter.



Specification

This program prompts the user for their name and asks whether they want to sort an array of 1000, 10,000, or 30,0000 integers. The program then continues and runs from selection sort function first and sorts the array to ascending order from smallest to largest. Once it completes the program will display the time it took for selection sort and continue on to sorting in bubble sort. Once bubble sort is complete, the program will display the time it took for buble sort.

Analysis

Inputs: User must input name, and the array size of choice they would like to be sorted.

a=1000, b=10,000 , c = 30,0000

Process:

1.) → Ask user for name
2.) → Ask user to choose the array size of their choice (a,b,c)
3.) → Begin time and selection sort
4.) → Display 3 pre-sorted arrays { random, ascending, descending)
5.) → Sort the arrays by placing them the smallest element in front and comparing them to the others to sort them correctly.

Analysis Cont.

6.) → Display sorting, end time and begin bubble sort.

7.) → Display 3 pre-sorted arrays again (random, ascending, descending)

8.) → Sort the arrays by finding the largest element and moving it to the end of the array.

9.) → Continue to move elements until the array is left with the smallest element in the front and largest in back.

Outputs: Program outputs the pre-sorted random array order, ascending order, and descending order. It then prints out the arrays in a sorted ascending order. Does it twice. Once with selection sort and once again with bubble sort.

Design

Lab.h → Header file that includes the libraries needed and lists the function declarations.

getName.cpp → Prompts the user for their name and prints the greeting message.

Lab.cpp → Main function, asks the user for their name and array size of their choice they would like sorted from ascending, descending, and random back to ascending order.

SelectSort.cpp → Sorts the selection sort and bubble sort functions by following the correct algorithm

Swap.cpp → Swap function that uses a temporary variable to take place of int k in order to swap the numbers in the indexes of the function.

Design Cont.

ArrayVal.cpp → Takes user input and checks what character they inputted. Returns the value associated with that letter as the amount being sorted in the array.

(measured in seconds)

Ascending Order	.00083	0.000829681	0.000884078	0.000842145	0.0447297 (b)	0.328234 (a)	0.0457241 (b)	0.327576 (c)	0.0387292 (b)	.038563 (b)
Random Order	0.000778638	0.000788762	0.00081696	0.000775418	0.035639 (b)	0.315255 (c)	0.036022 (b)	0.32033	0.0371418 (b)	0.0370916 (c)
Descending	0.00065719	0.00655505	0.00068986	0.000653705	0.0345432 (b)	0.308812 (c)	0.035061 (b)	0.314278	0.0360433 (b)	0.035961 (b)

Test

```
Hey, can I get your name?
omar
Hello, omar
What would you like the array size?.
a. 1000 (one thousand)                  b. 10000(ten thousand)              c. 30000 (thirty thousand)              Valid option must be entered!! : a
Array size... 1000
 ------------------------------------------
Generating pre-sorted elements in arrayFirst 10 and last 10 elements of the pre-sorted ordered array generated...:
1       2       3       4       5       6       7       8       9       10
991     992     993     994     995     996     997     998     999     1000

First 10 and last 10 elements of the pre-sorted random array generated:
383     886     777     915     793     335     386     492     649     421
693     334     439     334     421     159     985     957     354     761

First 10 and last 10 elements of the pre-sorted reverse array generated are
1000    999     998     997     996     995     994     993     992     991
10      9       8       7       6       5       4       3       2       1

Continuing to initialize selection sort algorithm...

Array Sortation Completed.. 0.000478369 seconds for regular sort


 Array Sortation Completed..  0.000443097 seconds for random sort


Array Sortation Completed.. 0.000373543 seconds for descending sort



First 10 and last 10 elements of the sorted ordered array:
1       2       3       4       5       6       7       8       9       10
991     992     993     994     995     996     997     998     999     1000

First 10 and last 10 elements of the sorted random array:
0       2       2       4       6       8       9       10      11      11
993     994     994     996     996     996     996     996     999

First 10 and last 10 elements of the sorted reverse array generated :
1       2       3       4       5       6       7       8       9       10
991     992     993     994     995     996     997     998     999     1000

----------------------------
Pre-Sorted BUBBLE loading...
```

Test Cont.

```
Array Sortation Completed.. 0.000478369 seconds for regular sort


 Array Sortation Completed..  0.000443097 seconds for random sort


Array Sortation Completed.. 0.000373543 seconds for descending sort



First 10 and last 10 elements of the sorted ordered array:
1       2       3       4       5       6       7       8       9       10
991     992     993     994     995     996     997     998     999     1000

First 10 and last 10 elements of the sorted random array:
0       2       2       4       6       8       9       10      11      11
993     994     994     996     996     996     996     996     999

First 10 and last 10 elements of the sorted reverse array generated :
1       2       3       4       5       6       7       8       9       10
991     992     993     994     995     996     997     998     999     1000

----------------------------
Pre-Sorted BUBBLE loading...
First 10 and last 10 elements of the pre-sorted ordered array generated (BUBBLE):
1       2       3       4       5       6       7       8       9       10
991     992     993     994     995     996     997     998     999     1000

First 10 and last 10 elements of the pre-sorted random array generated (BUBBLE):
383     886     777     915     793     335     386     492     649     421
762     972     541     716     852     850     662     482     399     217

First 10 and last 10 elements of the pre-sorted reverse array generated (BUBBLE):
1000    999     998     997     996     995     994     993     992     991
10      9       8       7       6       5       4       3       2       1

 ----------------Begin Bubble Sort --------------------

First 10 and last 10 elements of the sorted ascending (BUBBLE) is
1       2       3       4       5       6       7       8       9       10
991     992     993     994     995     996     997     998     999     1000


10      9       8       7       6       5       4       3       2       1

--> Total run time: 0.00141828 seconds.
```

```
Hey, can I get your name?
Jafar
Hello, Jafar
What would you like the array size?.
a. 1000 (one thousand)                  b. 10000(ten thousand)              c. 30000 (thirty t
Array size... 30000
 ------------------------------------------
Generating pre-sorted elements in arrayFirst 10 and last 10 elements of the pre-sorted ordered a
1       2       3       4       5       6       7       8       9       10
29991   29992   29993   29994   29995   29996   29997   29998   29999   30000

First 10 and last 10 elements of the pre-sorted random array generated:
29383   886     12777   16915   7793    8335    5386    492     26649   21421
16804   16831   29898   17419   10367   28762   2132    16430   4331    17969

First 10 and last 10 elements of the pre-sorted reverse array generated are
30000   29999   29998   29997   29996   29995   29994   29993   29992   29991
10      9       8       7       6       5       4       3       2       1

Continuing to initialize selection sort algorithm...

Array Sortation Completed.. 0.330879 seconds for regular sort


 Array Sortation Completed..  0.330979 seconds for random sort


Array Sortation Completed.. 0.321128 seconds for descending sort



First 10 and last 10 elements of the sorted ordered array:
1       2       3       4       5       6       7       8       9       10
29991   29992   29993   29994   29995   29996   29997   29998   29999   30000

First 10 and last 10 elements of the sorted random array:
0       0       1       3       3       3       8       8       12      13
29990   29993   29994   29994   29994   29995   29995   29995   29998

First 10 and last 10 elements of the sorted reverse array generated :
1       2       3       4       5       6       7       8       9       10
29991   29992   29993   29994   29995   29996   29997   29998   29999   30000
```

```
Array Sortation Completed.. 0.330879 seconds for regular sort


 Array Sortation Completed..  0.330979 seconds for random sort


Array Sortation Completed.. 0.321128 seconds for descending sort



First 10 and last 10 elements of the sorted ordered array:
1       2       3       4       5       6       7       8       9       10
29991   29992   29993   29994   29995   29996   29997   29998   29999   30000

First 10 and last 10 elements of the sorted random array:
0       0       1       3       3       3       8       8       12      13
29990   29993   29994   29994   29994   29995   29995   29995   29998

First 10 and last 10 elements of the sorted reverse array generated :
1       2       3       4       5       6       7       8       9       10
29991   29992   29993   29994   29995   29996   29997   29998   29999   30000

----------------------------
Pre-Sorted BUBBLE loading...
First 10 and last 10 elements of the pre-sorted ordered array generated (BUBBLE):
1       2       3       4       5       6       7       8       9       10
29991   29992   29993   29994   29995   29996   29997   29998   29999   30000

First 10 and last 10 elements of the pre-sorted random array generated (BUBBLE):
29383   886     12777   16915   7793    8335    5386    492     26649   21421
29160   26489   6369    4413    14468   16127   21002   14978   9366    5618

First 10 and last 10 elements of the pre-sorted reverse array generated (BUBBLE):
30000   29999   29998   29997   29996   29995   29994   29993   29992   29991
10      9       8       7       6       5       4       3       2       1

 ----------------Begin Bubble Sort --------------------

First 10 and last 10 elements of the sorted ascending (BUBBLE) is
1       2       3       4       5       6       7       8       9       10
29991   29992   29993   29994   29995   29996   29997   29998   29999   30000


10      9       8       7       6       5       4       3       2       1

--> Total run time: 0.983896 seconds.
```

Authors Code

I used the authors code in my program effectively I only adapted swap into it's own function and I set the variable i to start from 0 instead of 1.

```
    for (n = size;   n >= 2;   n--) {

        // Find the index "iMax" of the largest element
        //   among a[0], ..., a[n-1]:

        for (iMax = 0, i = 1;   i < n;   i++)
            if (a[i] > a[iMax]) iMax = i;

        // Swap a[iMax] with a[n-1]:

        aTemp = a[iMax];     // Save a[iMax] in a temporary location.
        a[iMax] = a[n-1];    // Copy a[n-1] to a[iMax].
        a[n-1] = aTemp;      // Copy saved value to a[n-1].

        // Decrement n (accomplished by n-- in the "for" loop).
    }
}
```

```
    {
        for (iMax = 0, i = 0; i < n; i++)
        {
            if(Rand_A[i] > Rand_A[iMax])
            {
                iMax = i;
            }
        }

        Swap(Rand_A[iMax], Rand_A[n-1]);
        n--;
    }
```