DynamicArray Template, Lab Assignment 3

Re-submit Assignment

Due Sep 20 by 11:59pm **Points** 100 **Submitting** a file upload **File Types** cpp and h

This is a two-part assignment. You may wait to submit the files for both parts when you are done with both, or submit part 1 first and part 2 later. It should work just like your **StaticArray** from Lab Assignment 2, but with auto-adjusting capacity.

PART 1

Write And Test An Array Class [DynamicArray.h *and* DynamicArray.TestDriver.cpp]
Write and test a data structures template. The resulting template can be used in any program in place of a C++ array.

Requirements. Develop **DynamicArray.h** as you write **DynamicArray.TestDriver.cpp** with **class DynamicArray**, defined and *fully tested*. Write the public interface exactly as specified below -- do not add to, or change the public interface as specified.

1. Write the template for an array of 2 values (initially) of unspecified type.
2. Include a public square bracket getter and setter pair, both with index range-checking, returning whatever value you wish if out of range. But apply capacity auto-adjusting for the *setter* if out of range high.
3. Include a public getter named **DynamicArray::capacity( )** to return the data structure's now-variable capacity
4. Include a public setter named **DynamicArray::capacity(int)** to change the capacity.
5. Do tests with `int`, `double`, or `char`. Also do tests with an object, like `string`.

Note that there is no good reason to copy the "dummy" value in the dynamic memory management functions, so don't include it in your testing of const object copy or object assignment.

PART 2

Write An Array Application [MyDynamicArray.cpp]
Write **MyDynamicArray.cpp** using your**DynamicArray template**. Use your already-tested and verified H file from part 1.

Exactly as in Lab Assignment 2a's **MyStaticArray.cpp**, *this* app lets its user enter as many values as they like, and when that process is completed, lets the user look up values by matching index.

In a loop, the app should prompting the user to enter a *pair of numbers* on the same line: a whole number *index* and its corresponding ***floating point*** *value*. *Do* validate index input in the app. Quit the loop when an *uppercase* or *lowercase* Q is entered for *either* the index or the value. Indexes can be entered in any order -- they don't have to start with zero and go up by one thereafter. It's whatever the user enters.

Your app should keep track of which indexes got entered. Use a *bool* ***DynamicArray*** for that.

After all data entry is complete, the app should:

1. output how many (unique) indexes got entered,
2. output the list of all used indexes and their values, per the example below, and
3. implement an event-controlled loop that prompts for an index value and outputs whether the index is in use or not, and if in use, what is the value stored for that index. Loop until the user elects to stop by entering uppercase or lowercase Q.

Here's a sample of how this should work (user input in blue):

```
Input an index and a value [Q to quit]: 33 1.2
Input an index and a value [Q to quit]: 4 100
Input an index and a value [Q to quit]: 5 300
Input an index and a value [Q to quit]: x 1.7
Input an index and a value [Q to quit]: 33 120
Input an index and a value [Q to quit]: -1 23.4
Input an index and a value [Q to quit]: 2000 -999.9
Input an index and a value [Q to quit]: q
```

```
You stored this many values: 5
The index-value pairs are:
  0 => 1.7
  4 => 100
  5 => 300
  33 => 120
  2000 => -999.9

Input an index for me to look up [Q to quit]: 33
  Found it -- the value stored at index 33 is 120
Input an index for me to look up [Q to quit]: 0
  Found it -- the value stored at index 0 is 1.7
Input an index for me to look up [Q to quit]: -10
  Sorry, but there is no value stored at index -10
Input an index for me to look up [Q to quit]: 38
  Sorry, but there is no value stored at index 38
Input an index for me to look up [Q to quit]: 10000
  Sorry, but there is no value stored at index 10000
Input an index for me to look up [Q to quit]: 2000
  Found it -- the value stored at index 2000 is -999.9
Input an index for me to look up [Q to quit]: q
```

Design the prompts and the output formatting as you like.

Lab Assignment Rubric

Criteria	Ratings							Pts
Fully accurate results, following all specifications **view longer description**	Works the first time. 70.0 pts	Works on the 2nd try 65.0 pts	Works on the 3rd try 60.0 pts	Works after 4 or more tries. 50.0 pts	Doesn't work after 2 weeks. Partial credit. 20.0 pts	Not submitted within two weeks of the due date. 0.0 pts	Work is not original -- appears to be a marked-up copy of the work of another or previous student. 0.0 pts	70.0 pts
Submits all work on time, fully complete if not fully correct. **view longer description**	Submitted on time 20.0 pts	Submitted on time, but one or more files are missing or not correctly named. 16.0 pts	Submitted on time, but with missing identification in one or more submitted CPP or H files. 15.0 pts	Submitted on time but not fully complete. 10.0 pts	Late or wholly incomplete! 0.0 pts			20.0 pts
Well-organized and professional quality code. **view longer description**	Fully meets expectations 10.0 pts	Mostly meets expectations, just needs to be a bit more careful. 8.0 pts	Many areas are well done, but there are a lot of areas that need work. 6.0 pts	Getting there, but needs to be a lot better. 3.0 pts	Needs a lot of work. See the instructor for guidance. 0.0 pts			10.0 pts
							Total Points: 100.0	