Carson Hanel
7/11/2017

1. Program Description; Purpose of the Assignment:
    a. The program is the implementation of the STL Vector class under the pseudonym “My_vec”, and not only is implemented to run with character type data, but any type as is shown by testing the Templated_My_vec with three different data types. The purpose of this assignment was to get more comfortable with the fundamentals of dynamic memory, I assume, and to have hands on experience implementing both search and sort algorithms that can handle any data type.
2. Data Structures Description:
    a. Theoretical Definition: A vector is an array of a defined class of object that acts as a wrapper for the array, allowing more fluid resizing and handling and throwing errors when addresses are incorrectly used.
    b. Real Implementation: To implement a STL vector, you must have the ingredients of the vector ADT. The three main ingredients for the class data are size, capacity and a pointer to whatever data type is being input into the vector. While the pointer exists within the stack, the array of defined type resides in the heap memory.
    c. Analysis of the best and worst case scenarios for vector:
        i. The best case:
            1. Insert_at_rank: Constant time if no values have to be moved and there is appropriate space within the capacity.
            2. Remove_at_rank: Constant time if removing the rightmost value of a vector.
            3. Find_max_index: n-1 comparisons used in best/worst case.
            4. Sort_max: n*log(n) comparisons made in the best case.
            5. The rest of the ADT functions are constant time in all cases.
        ii. The worst case:
            1. Insert_at_rank: The worst “case” for insertion is the case in which the capacity must be increased. When this happens, all vector items must be copied to a vector of twice the capacity, which takes linear n time.
            2. Remove_at_rank: The worst case for moving an index is removing at index 0, in which case you would have n-1 (n including the index 0) operations to remove a vector item.
            3. Find_max_index: n-1 comparisons used in best/worst case
            4. Sort_max: n*log(n) comparisons made in the worst case.
    d. Instructions:
        i. You can “make” and ./main run the non templated version. I couldn’t get the templated version compiled, so I’ve included the .exe that I compiled from DevKit C++, but I am fully confident the code works on UNIX

systems. I break the first 8 “tests” up in cout, and then the sorting/searching and final insertion are on their own.

ii. The templated version test ALL cases for three different values. I’m sorry if this looks like junk. I just thought it would be neat having all the variables in one place.

iii. I’ve made it to where the program sorts 12345 into 54321 since 12345 was asked for as the first 5 members of the arrays.

e. Logical Exceptions:
   i. Instead of using cerr and interrupting the process, I’ve used if/else protection on my error throws, and the message goes to cout rather than cerr. There are four exceptions that can be reached:
      1. Accessing a memory address out of range
      2. Attempting to insert at an invalid rank
      3. Replacing a rank that does not exist
      4. Attempting to access a vector rank that does not exist
   ii. All four exceptions are handled with cout, and I have made sure that no variables are changed if they’re thrown.
f. C++ OO or generic programming features:
   i. Classes are an object oriented feature.
   ii. Pointers are a major factor in object orientation.
   iii. Other things are great as well.
g. Testing results:
   i. There are a few insert out of range calls that happen, but other than that the testing was straightforward. Thank you for a fun assignment!