Description and Explanation of the Design Process with Testing Files

Name: Mike Kim

Date: September 29th, 2022

To ensure that all the functions work with multiple different data types, I have created an integer based double-ended queue and a string based double-ended queue. The main goal of this designing process was to ensure that the functions worked, no matter the data type and when it was called, thus enabling the deque class to pass all of the test cases.

Integer Based Double-Ended Queue

Firstly, checking the status of the deque and the array prior to any modification was demonstrated by using the display and ddisplay function. Next, enqueueing ten elements allowed people to ensure two things. First thing was whether if an element was getting enqueued into the queue properly. Second thing was to check if it would reserve more space since it was exceeding the default capacity, which was 8. Next, to test whether if jump function works in a chronological order, I jumped 35, 36, then 37. Unlike jumping the same number three times – this would surely demonstrate the accuracy of this function as properly working function would put 37 as the very first element on the display function. Afterwards, I went on to test out the eject function. I started it out by calling the function twice then using the display and ddisplay function to see if two elements have been removed properly and was still stored in the array system. Furthermore, to check if explicit calling of reserve function worked when newCapacity variable exceeded the current capacity (theCapacity), I called reserve with newCapacity being equivalent to 20. To check if the function worked properly, I have used the ddisplay function to provide me with the updated capacity. Lastly, I needed to check if clear function matches the description. To do so, after calling the function, I have used display function and ddisplay function to make sure that nothing was in the queue and the array still stored the data.

String Based Double-Ended Queue

Making a double ended queue based on char data type allowed others to confirm that these functions work with multiple distinct data types. To see if items were being enqueued in chronological order and being jumped properly, I have used the display function to confirm the status. For convenience, I've enqueued "three", "four", "five", and jumped "two" and "one" so when properly implemented, display function would show: < one, two, three, four, five, >, which was easy to recognize. I wanted to make sure that jump function worked whether if I have modified the queue beforehand or not. Once those functions were performed, dequeue function was iterated twice to see if it was implemented properly. By calling the dequeue function then using display that showed < three, four, five, > it showed that it has removed the front two elements out of the queue properly. To see eject function works after the queue has been modified, I have called it twice after the dequeue function. The expected outcome on display

function was < three, > since eject function removes the item from the very end of the queue. By using the display function, I was able to confirm that eject function was implemented properly. To check whether if reserve function increased the capacity when called explicitly under the condition where the new capacity was bigger than the current one (current capacity was 8), I called "strDeque.reserve(13);" to see whether if it actually expanded the capacity. To ensure that reserve function was implemented correctly, I've used the ddisplay function to notify me with the current capacity. Additionally, I have used the display function to ensure that nothing else has changed except the capacity. Lastly, to check if clear function matched the description of the desired implementation, I have called the function then checked with display function to make sure there were no contents, and the size was equivalent to zero. Additionally, I have called the ddisplay function to make sure no other traits such as capacity and the array were modified.

DQarraytest.cpp

To test out the array-like access to all elements in the deque for the function "int& operator[]", I have enqueued 0,2,4,6,8,10 to the queue. Afterwards, I have tried to access them by putting numerical values within the indices range and outputted them. This testing methodology ensured that this function works for multiple different types of data.