Vazquez, Erian

COP3530 Data Structures and Algorithm

Project 1b: Implement a Double-Ended Queue Documentation

Code organization-

I used an array to create this double ended queue, to be able to access indexes much quicker. I will store where the head and tail location, this will be helpful since accessing location with the a stored location take time O(1) with an array. Since there will be no need to shift items an array is ideal. Items will need to be copied if we run out of space and need to make the array larger but at worst case this will take O(n) time.

I created a method for each functions and then created a shrink method that would help the pop functions. I was not able to do the same for the push functions as heads and tails ended up different locations.

- push_front - Inserts an element at the front of the queue. This will take a template object and output void. This will cause the array to grow when it is full to double its current size.
- push_back - Inserts an element at the back of the queue. This will take a template object and output void. This will cause the array to grow when it is full to double its current size.
- pop_front - Deletes the element at the front of the queue. This has no input but outputs an object template. This may cause the array to shrink to ½ its original size if it is ¼ and the capacity is greater than 8.
- pop_back - Deletes the element at the back of the queue. This has no input but outputs an object template. This may cause the array to shrink to ½ its original size if it is ¼ and the capacity is greater than 8.
- empty - checks to see if the queue is empty. This takes in no object and output a bool.
- size - returns the size of the queue. This takes in no object and outputs an int with the capacity.
- toStr – concatenates the contents of the queue from front to back into a return string with each string item followed by a newline character in the return string. This has no input but outputs previously stated string.

How to use-

Provide main file named tDeque_main.cpp that will allow the user to select the object type that they want to create a double ended queue with. Then create a Deque and use the above functions to add, remove, and print objects in the Deque. To be able to use the toStr function your object must overload <<.

Exceptions will be thrown if user attempts to remove an item when nothing is there and if the user attempts to add a lot of items, the provided main file that uses Deque should catch these exceptions.

Special Diagnostics –

Only tools used to test the temple double ended queue were built in complier tools such as, breakpoints and debug mode. Along with this added cout statements along with other checks were used to ensure desired outputs.

Bugs-

No known bugs

Test Cases -

Test Case	Test Description	Expected Result	Observed Result	Pass/Fail?
01	Add 4 items with push back and then print	erian vazquez did this	erian vazquez did this	Pass
02	Add 4 items with push forward and then print	this did vazquez erian	this did vazquez erian	Pass
03	Add 3 items then remove 2 end up with 1 item and does not shrink print	Erian 8	Erian 8	Pass
04	Add 8 elements print size	erian vazquez did this erian vazquez did this 8	erian vazquez did this erian vazquez did this 8	Pass
05	Add 9 elements print size see if it grows	erian vazquez did this erian vazquez did this erian 16	erian vazquez did this erian vazquez did this erian 16	Pass
06	Add 17 elements print size then remove 13 then print size	32 8	32 8	Pass
07	Pop right away	Throw exception	Throw Exception	Pass
08	Add a ton of items	Throw bad alloc	Throw bad alloc	Pass