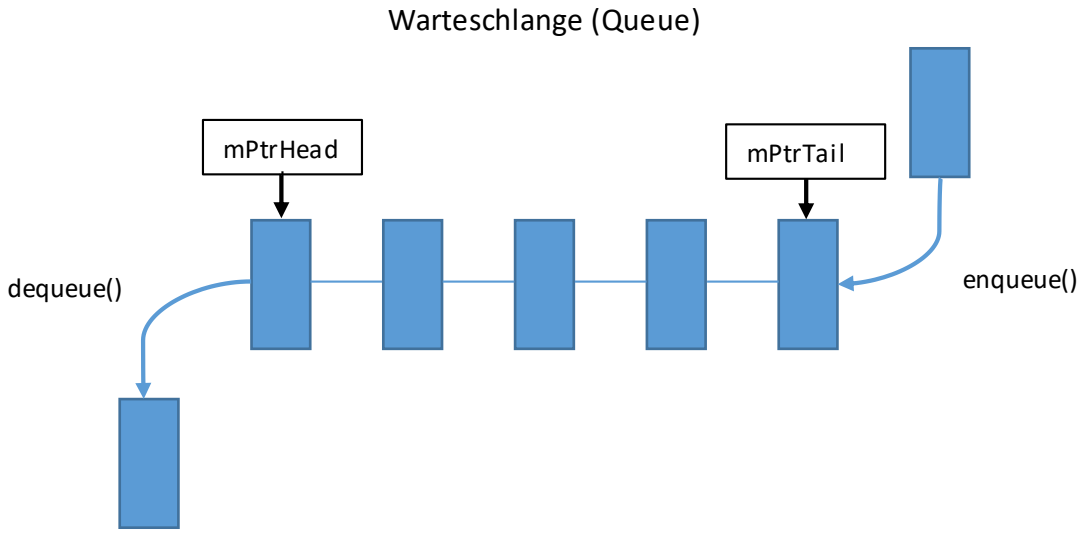
Aufgabe – Dynamische Warteschlange (queue)

Implementieren Sie in der Klasse „CQueueDyn“ (CQueueDyn.h, CQueueDyn.cpp) eine dynamische Warteschlange. Ebenso wie in der letzten Aufgabe (Dynamischer Stack) verwenden wir kein Root-Element. Die maximale Anzahl der Elemente in der Queue ist ebenfalls nicht von vornherein begrenzt. Die Klasse CElement können Sie 1:1 aus der vorhergehenden Aufgabe übernehmen.

Im Unterschied zu einem Stack, der nach dem LIFO-Prinzip (**L**ast **I**n **F**rist **O**ut) arbeitet, funktioniert eine Queue (bzw. Warteschlange) nach dem FIFO-Prinzip (**F**irst **I**n **F**irst **O**ut):

> *„Man kann sich eine Warteschlange wie eine Warteschlange von Kunden an einer Kasse vorstellen. Der Letzte, der sich in die Schlange stellt, wird auch als letzter bedient. Umgekehrt wird derjenige, der sich als erstes angestellt hat, als erster bedient.“* (Wikipedia: https://de.wikipedia.org/wiki/Warteschlange_(Datenstruktur) , 08.12.2020).



Implementieren Sie die Methoden … enqueue(…) und … dequeue(…) zum Hinzufügen und Herausnehmen von Elementen.

Hinweise:

1. Eine der beiden Methoden – dequeue() – funktioniert vergleichbar mit der Methode pop() für einen Stack: Das erste Element wird aus der Warteschlange bzw. vom Stack genommen.

2. Um die Implementierung der Methode enqueue() zu vereinfachen, verwalten Sie zusätzlich zum Zeiger auf das erste Element in der Warteschlange (mPtrHead) einen Zeiger mPtrTail. Dieser Zeiger zeigt immer auf das aktuell letzte Element in der Queue.

3. Lesen Sie sich den oben genannten Wikipedia-Artikel durch.

Eine Header-Datei (CQueueDyn.h) ist diesmal nicht vorgegeben und von Ihnen zu erstellen.

Die Main-Funktion zum Testen können Sie fast 1:1 aus der letzten Aufgabe übernehmen – statt push und pop gibt es jetzt enqueue und dequeue.

Wie gewohnt erwarte ich eine ausführlich kommentierte Lösung sowie die Abgabe von Screendumps mit geeigneten Testläufen.

Viel Spaß und Erfolg bei der Bearbeitung!