INSTITUTO TECNOLÓGICO DE COSTA RICA
Escuela de Computación
Bachillerato en Ingeniería en Computación
IC-2001 Estructuras de Datos
Prof. Mauricio Avilés

Tarea Corta 3 – Listas doblemente enlazadas

Esta tarea consiste en la implementación, utilización y modificación de las siguientes estructuras de datos:

1. Lista doblemente enlazada (DLinkedList)
2. Lista circular doblemente enlazada (DCircleList)

Parte 1

Implementar la clase **DLinkedList**. Esta clase debe derivar de la clase abstracta **List** vista en clase. A continuación, se detallan los métodos que deben implementarse.

1. **Constructor y destructor**
2. **void insert(E element)**
3. **void append(E element)**
4. **E remove()**
5. **void clear()**
6. **E getElement()**
7. **void goToStart()**
8. **void goToEnd()**
9. **void goToPos(int pos)**
10. **int getPos()**
11. **int getSize()**
12. **void sort()**
    Este método ordena los elementos de la lista de menor a mayor. El algoritmo de ordenamiento debe ser el Quicksort. Haga uso de listas temporales para implementar el algoritmo.

Escriba un programa que demuestre extensamente el funcionamiento de todos los métodos de la clase. Utilice cantidades grandes de datos (miles de elementos). Muestre claramente el funcionamiento del método **sort**.

Parte 2

Implementar la clase **DCircleList**. Esta es una lista circular implementada con nodos doblemente enlazados. El funcionamiento es exactamente el mismo que la clase **CircleList**, pero cada nodo está enlazado con el nodo siguiente y el anterior. Esta clase no debe heredar de la clase abstracta **List**. A continuación, se detallan los métodos que deben implementarse:

1. **Constructor y destructor**
2. **void insert(E element)**
3. **E remove()**
4. **void clear()**

5. **E getFront()**
6. **E getBack()**
7. **void next()**
8. **void previous()**
9. **int getSize()**
10. **DLinkedList<E>* getElements(bool reversed = false)**
    Este método debe crear en memoria dinámica una instancia de **DLinkedList**, insertar en ella los elementos que contiene la lista circular (en el mismo orden), y retornar el puntero a la **DLinkedList**. Recibe por parámetro un valor booleano que indica si la lista circular debe recorrerse en orden normal o en orden inverso. Si el parámetro *reversed* es falso, entonces los elementos se deben insertar en la lista resultado desde el que se encuentra en la posición *front*, continuando hasta el que se encuentra en la posición *back*. Si el parámetro es verdadero, entonces los elementos deben insertarse en la lista resultado desde *back* hasta *front*. La lista circular no debe ser modificada, al finalizar el método debe contener los mismos elementos y la posición actual debe encontrarse apuntando al mismo lugar que estaba antes de ejecutar el método.

Escriba un programa que demuestre extensamente el funcionamiento de todos los métodos de la clase. Utilice cantidades grandes de datos. Muestre claramente que el método **getElements** retorna la lista de elementos en los dos órdenes. Este programa es el responsable de liberar la memoria de las **DLinkedList** creadas por el método **getElements**.

Entrega: Sección de evaluaciones del TEC-Digital.
Formato: Archivo comprimido ZIP con dos proyectos de Code::Blocks, uno por cada parte de la tarea. Corrobore que se incluyen todos los archivos necesarios para que los proyectos se ejecuten correctamente.