Pràctica 2. Estructura de dades DEQUE (Double Endend QUEue)

1. Introducció

***Objectius**:* Familiaritzar-se amb les estructures lineals

Temes de teoria relacionats amb la pràctica: Tema 3 Estructures de dades lineals bàsiques

2. Enunciat

Una cua doblement acabada o deque (de l'anglès *double ended queue*) és una estructura de dades lineal que permet inserir i eliminar elements per ambdós extrems de la cua. Noteu que en una única estructura es permet les funcionalitats de les piles (LIFO- Last Input First Output) i de les cues.

Les seves principals operacions són:

OPERACIÓ	DESCRIPCIÓ
void insertFront(int key);	Inserir l'element al davant de la cua
void insertRear(int key);	Inserir l'element al final de la cua
void deleteFront();	Eliminar l'element que està al davant
void deleteRear();	Eliminar l'element que està al final
bool isFull();	Retorna cert si l'estructura està plena
bool isEmpty();	Retorna cert si l'estructura està buida
void print();	Imprimeix tots els elements de l'estructura
int getRear()	Retorna l'element que està al final de la cua
int getFront()	Retorna l'element que està al davant de la cua

Exercici 1. Implementeu el TAD ArrayDeque amb una implementació amb array

A continuació teniu la definició de l'especificació de la classe ArrayDeque.

```
#ifndef ARRAYDEQUE_H
#define ARRAYDEQUE_H
#Aquí poseu els includes que necessiteu

class ArrayDeque{
    public:
        ArrayDeque( int maxSize) ;
        ~ArrayDeque();
        bool isEmpty() const;
        bool isFull()const ;
       void insertFront(int element);
       void insertRear(int element);
       void deleteFront();
       void deleteRear();
       void print() const;
       int getFront()const;
       int getRear()const;
    private:
        int size;
        std::vector<int> data ;
        int front;
        int rear;
        int count;

};
#endif /* ARRAYDEQUE_H */
```

Realitzeu la implementació del TAD ArrayDeque. Posteriorment, codifiqueu un *main.cpp* que tingui un *menu* amb les següents opcions:

1. Inserir element pel davant
2. Inserir element pel final
3. Eliminar element pel davant
4. Eliminar element pel final
5. Imprimir contingut de l'ArrayDEQUE
6. Sortir

Cas de prova 1:

Pas	**Entrada**	Sortida
1	Crear un ArrayDeque de mida 5	Estructura creada
2	Eliminar element pel davant	EXCEPTION: L'estructura està buida
3	Eliminar element pel final	EXCEPTION: L'estructura està buida
4	Inserir element 20 pel davant	Element 20 agregat
5	Inserir element 30 pel davant	Element 30 agregat
6	Inserir element 80 pel final	Element 80 agregat
7	Inserir element 45 pel final	Element 45 agregat
8	Eliminar pel davant	Element 30 eliminat
9	Imprimir ArrayDeque	[20, 80, 45]

Cas de prova 2:

Pas	Entrada	Sortida
1	Crear un ArrayDeque de mida 3	Estructura creada
2	Eliminar element pel davant	EXCEPTION: L'estructura està buida
3	Eliminar element pel final	EXCEPTION: L'estructura està buida
4	Inserir element 2 pel davant	Element 2 agregat
5	Inserir element 3 pel davant	Element 3 agregat
6	Inserir element 8 pel final	Element 8 agregat
7	Inserir element 4 pel final	EXCEPTION: Estructura plena
8	Eliminar pel davant	Element 3 eliminat
9	Imprimir ArrayDeque	[2, 8]

Exercici 2. Implementar TAD LinkedDeque amb una estructura enllaçada

En aquest exercici es demana dissenyar i implementar l'estructura de dades `LinkedDeque` com una estructura dinàmica amb encadenaments.

Aquesta LinkedDeque està formada per nodes de tipus TAD `Node`. Aquest TAD Node conté com a mínim tres atributs: `element` on es guarda l'element a inserir a la cua, `next` que és l'apuntador al següent node i `previous` que és l'apuntador al node anterior. L'especificació amb el mínim d'operacions necessàries del TAD Node és la següent:

- **`constructor`**: construeix el node amb l'element que rep com a paràmetre
- **`getElement`**: retorna l'element que hi ha guardat al node
- **`getNext`**: retorna l'adreça del següent node o *`nullptr`* en cas que no hi hagi següent
- **`setNext`**: modifica l'adreça de `next` per l'adreça rebuda com a paràmetre
- **`getPrevious`**: retorna l'adreça del node anterior o *`nullptr`* en cas que no hi hagi anterior
- **`setPrevious`**: modifica l'adreça de `previous` per l'adreça rebuda com a paràmetre

A continuació es presenta l'especificació del TAD LinkedDeque.

```
#ifndef LINKEDDEQUE_H
#define LINKEDDEQUE_H
// POSEU aqui els vostres includes

#include "Node.h"

template class <Element>
class LinkedDeque{

    public:
        LinkedDeque();
        ~LinkedDeque();
       LinkedDeque(const LinkedDeque<Element>& deque);
       bool isEmpty()const;
       void insertFront(const Element & element);
       void insertRear(const Element & element);
       void deleteFront();
       void deleteRear();
       void print();
       const Element& getFront()const;
       const Element& getRear()const;
```

```
        private:
         Node<Element> *_front;
         Node<Element> *_rear;
         int  num_elements;
};

#endif /* LINKEDDEQUE_H */
```

Fixeu-vos en alguns detalls d'aquesta LinkedDeque:

- La cua s'ha definit amb nodes de tipus Node, que són bidireccionals.
- En aquesta pràctica es demana la implementació del Deque amb Node bidireccionals es faci amb dos sentinelles. Això facilitarà la implementació de les diferents funcions de la LinkedDeque.
- Cal implementar el ***constructor de còpia***. Aquest ha de duplicar la cua encadenada, de forma que els espais de memòria dels nodes de la cua des d'on es copia i de la cua final siguin diferents. No n'hi ha prou amb copiar la direcció dels punter *_front* i *_rear*, sinó que cal fer-ne un de completament nou per cada un dels elements de la cua original.
- La funció ***size*** retorna el nombre d'elements de la LinkedDeque en aquest moment.
- La funció ***isEmpty*** retorna *true* si la LinkedDeque està buida, *false* en cas contrari.
- Es pot inserir tant al davant de la LinkedDeque amb ***insertFront*** com al final d'aquesta amb ***insertRear***. La inserció sempre es realitza entre els dos sentinelles. És a dir, insertFront inserta davant de tots els elements i darrera del primer sentinella i insertRear inserta al final de tots elements i davant del darrer sentinella.
- Es pot eliminar tant al davant de la LinkedDeque amb ***removeFront*** com al final d'aquesta amb ***removeRear***. L'eliminació sempre es realitza entre els dos sentinelles. És a dir, removeFront elimina l'element darrera del primer sentinella i removeRerar elimina l'element davant del darrer sentinella.
- El mètode ***print*** ha d'imprimir per consola tots els elements de la LinkedDeque.
- Es pot consultar tant al davant de la LinkedDeque amb ***getFront*** com al final d'aquesta amb ***getRear***. La consulta sempre es realitza entre els dos sentinelles. És a dir, getFront retorna l'element darrera del primer sentinella i getRerar consulta l'element davant del darrer sentinella.

Per provar el vostre TAD LinkedDeque codifiqueu un *main.cpp* que tingui un *menu* amb les mateixes opcions i casos de prova de l'exercici anterior.

Preguntes a contestar a la memòria:

1. Has implementat la LinkedDeque amb templates? Sigui quina sigui la teva resposta, justifica el motiu pel qual has pres aquesta decisió
2. Tenint en compte la teva implementació del TAD LinkedDeque, indica per a cadascuna de les operacions del TAD LinkedDeque quin és el seu cost computacional teòric. Justifica la resposta
3. Creieu que la classe Node hauria estat millor implementar-la amb encadenaments simples? Justifica la teva resposta

Exercici 3. Utilitzeu el TAD LinkedDeque per a resoldre el següent problema.

Es vol realitzar la gestió d'una cua d'impressió. A la cua arriben documents amb el nom de l'usuari, la prioritat del document i el nom del document. Per exemple:

maria 1 practica.pdf
jefe 1 memoria.pdf
assistent 2 gestio.doc
assistent 2 dades.txt

La màxima prioritat és la 1 i aquests documents s'han de posar a l'inici de la cua d'impressió. La mínima prioritat és 2 i aquests documents es poden esperar a la cua d'impressió, per tant van al final.

Utilitzeu el TAD LinkedDeque implementat en l'exercici 2 per crear un programa que tingui les següents opcions de menú.

1. Llegir un fitxer amb les entrades de la cua d'impressió
2. Eliminar una impressió pel davant
3. Eliminar una impressió pel final
4. Inserir **n** entrades d'impressió des de teclat (0 per finalizar)
5. Imprimir la cua d'impressió

Fixeu-vos que la primera opció del menú és per introduir les dades des d'un fitxer de text. El format del fitxer és una línia per cada impressió i els camps estan separats per un espai o tabulador. A continuació teniu l'exemple de fitxer que us trobareu al campus virtual per a descarregar-lo.

maria 1 practica.pdf
gestor 1 memoria.pdf
assistent 2 gestio.doc
adan 1 dibuix.png
silvia 2 dades.txt
pere 1 horaris.xls
pablo 1 codi.cpp
anna 2 practica2.cpp

3. Lliurament

A partir de la descripció del problema, es demana:

- Implementar els exercicis en C++, versió 11. Creareu un projecte NetBeans per a cada exercici. Lliurar el codi C++ corresponent als vostres exercicis en una carpeta anomenada codi, amb una subcarpeta per a cada exercici. Els comentaris de cada exercici (observacions, decisions preses i resposta a les preguntes plantejades, si n'hi ha) els fareu com a comentari al fitxer main.cpp de cada exercici.

Com a màxim el dia del lliurament es penjarà en el campus virtual un fitxer comprimit **en format ZIP** amb el nom del grup (A, B, C, D, E o F), el número de parella (ParX), el nom dels dos membres del grup i el número de la pràctica com a nom de fitxer. Per exemple, **GrupA_Par2_BartSimpsonLisaSimpson_P2.zip**, on Par2 indica que és la "parella 2" i P1 indica que és la "pràctica 2". El fitxer ZIP inclourà: la carpeta amb tot el codi.

Els criteris per acceptar la pràctica són:

- La pràctica ha de funcionar en la seva totalitat.
- La pràctica ha de ser orientada a objectes.

IMPORTANT: La còpia de la implementació de la pràctica implica un zero a la nota de pràctiques de l'assignatura per les parelles implicades (tant la que ha copiat com la que ha deixat copiar).

4. Planificació

Per aquesta pràctica els professors proposen la següent planificació:

- **Setmana 1 (5 de març al 9 de març) (Classe de laboratori)**
  - Implementació del TAD de l'exercici 1
- **Setmana 2 (12 de març al 16 de març) (No hi ha classe de laboratori)**
  - Implementació del programa principal i dels casos de proves
  - Lliurament Exercici 1 al campus virtual fins al 18 de març de 2018
- **Setmana 3 (19 de març al 23 de març) (Classe de laboratori)**
  - Implementació de l'exercici 2
- **Setmana 4 (9 d'abril al 13 d'abril) (No hi ha classe de laboratori)**
  - Implementació de l'exercici 3
  - Lliurament del codi de TOTS els exercicis

El lliurament final de la pràctica serà el **dia 15 d'abril de 2018** al campus virtual.