Estructura de Dades: Pràctica 2

Pràctica 2. Estructura de dades DeQue

1. Introducció

Objectius: Familiaritzar-se amb les estructures lineals

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

Les pràctiques es faran amb l'IDE NetBeans 8.2 i la versió 11 de C++.

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 pels dos extrems. Es pot veure com un mecanisme que unifica en una única estructura les funcionalitats de les piles (estructura LIFO) y les cues (estructura FIFO). És a dir, aquestes estructures (piles i cues) es podrien implementar fàcilment amb un deque.

Les dues operacions més importants són la inserció (enqueue) d'un element i l'eliminació (dequeue) d'un element. Una cua pot tenir les següents operacions:

OPERACIÓ	DESCRIPCIÓ
void enqueueFront(int key);	Inserir l'element a l'inici la cua
void enqueueBack(int key);	Inserir l'element al final de la cua
void dequeueFront();	Treure el primer element de la cua
void dequeueBack();	Treure el darrer element de la cua
int size()	Retorna el nombre d'elements de la cua
bool isEmpty();	Retorna cert si l'estructura està buida
void print();	Imprimeix tots els elements de l'estructura
int getFront();	Retorna el primer element de la cua
int getBack();	Retorna el darrer element de la cua

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

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

```
#ifndef ARRAYDEQUE_H
#define ARRAYDEQUE_H

#include <vector>

class ArrayDeque {
public:
    ArrayDeque(const int);  // constructor on s'indica la mida
    void enqueueBack(const int key); // inserir al final de la cua
    void dequeueFront(); // eliminar el primer element de la cua
    bool isFull(); //  cert si està plena la cua, fals altrament
    bool isEmpty(); // cert si està buida la cua, fals altrament
    void print();  // imprimir tot el contingut de la cua
    const int getFront(); // obtenir el primer element de la cua
    void enqueueFront(const int); // inserir a l'inici de la cua
    void dequeueBack(); // eliminar el darrer element de la cua
    const int getBack(); // obtenir el darrer element de la cua
private:
    int _max_size; // mida màxima de la cua
    int _size; // nombre d'elements actuals a la cua
    std::vector<int> _data; // per guardar els elements de la cua
};

#endif /* ARRAYDEQUE_H */
```

Estructura de Dades: Pràctica 2

Realitzeu la implementació del TAD ArrayDeque. **Tingueu en compte que, tot i que sigui un vector, enlloc d'usar les operacions del TAD vector com push_back, etc..., s'espera que feu el control del contingut del TAD amb accessos com un array. És a dir, per exemple, per inserir feu _data[pos] = valor**.

Tot seguit, codifiqueu un *main.cpp* que tingui un *menú* amb les següents opcions:

1. Inserir element al davant de la cua
2. Inserir element al darrera de la cua
3. Treure element pel davant de la cua
4. Treure element pel darrera de la cua
5. Consultar el primer element
6. Consultar el darrer element
7. Imprimir tot el contingut de l'ArrayDeque
8. Sortir
9.

Quan tingueu el *menú* codificat, feu les següents proves:

Cas de prova 1:

Pas	**Entrada**	**Sortida**
1	Crear un ArrayDeque de mida 3	Estructura creada
2	Inserir element al davant 10	Element 10 agregat
3	Inserir element al darrere 20	Element 20 agregat
4	Inserir element al davant 30	Element 30 agregat
5	Inserir element al darrere 40	EXCEPTION: L'estructura està plena
6	Imprimir ArrayDeque	[30,10,20]
7	Treure element pel davant	Element 30 eliminat
8	Inserir element pel darrere 40	Element 40 agregat
9	Imprimir ArrayDeque	[10, 20, 40]

Cas de prova 2:

Pas	**Entrada**	**Sortida**
1	Crear un ArrayDeque de mida 3	Estructura creada
2	Inserir element pel davant 10	Element 10 agregat
3	Consultar el primer element del davant de la cua	10
4	Inserir element pel darrere 20	Element 20 agregat
5	Inserir element pel darrere 30	Element 30 agregat
6	Imprimir ArrayDeque	[10, 20, 30]
7	Treure element pel darrera	Element 30 eliminat
8	Consultar el darrer element de la cua	20
9	Treure el darrer element	Element 20 eliminat
10	Treure el primer element	Element 10 eliminat
11	Treure el primer element	EXCEPTION: L'estructura està buida
12	Imprimir ArrayDeque	[]

Estructura de Dades: Pràctica 2

Exercici 2. Implementeu el TAD Deque amb una estructura enllaçada

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

Aquesta LinkedQueue està formada per nodes de tipus TAD DoubleNode. Aquest TAD DoubleNode conté com a mínim tres atributs: *element,* on es guarda l'element a inserir a la cua, *next o següent* que és l'apuntador al següent node, i *previous o anterior* que és l'apuntador a l'anterior node.

L'especificació amb el mínim d'operacions necessàries del **TAD DoubleNode** és el 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 del següent per l'adreça rebuda com a paràmetre
- **getPrevious**: retorna l'adreça de l'anterior node o *nullptr* en cas que no hi hagi anterior
- **setPrevious**: modifica l'adreça de l'anterior per l'adreça rebuda com a paràmetre

A continuació es presenta l'especificació del TAD LinkedDeque.

```
#ifndef LINKEDDEQUE_H
#define LINKEDDEQUE_H

#include "DoubleNode.h"

class LinkedDeque {
public:
public:
    LinkedDeque();
    virtual ~LinkedDeque();
    LinkedDeque(const LinkedDeque& q);
    void enqueueFront(Patient& f);
    void enqueueBack(Patient& f);
    void dequeueFront();
    void dequeueBack();

    bool isEmpty();
    void print();
    const Patient& getFront();
    const Patient& getBack();
    int size();

private:
    // definiu aquí els vostres atributs per gestionar la cua doble

};

#endif /* LINKEDDEQUE_H */
```

Fixeu-vos en alguns detalls d'aquesta LinkedDeque:

- La cua s'ha definit amb nodes bidireccionals de tipus DoubleNode.
- 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 del final siguin diferents. No n'hi ha prou amb copiar la direcció dels punters, sinó que cal fer-ne un de completament nou per cada un dels elements de la cua original.
- La funció ***isEmpty*** retorna *true* si la LinkedDeque està buida, *false* en cas contrari.

- El mètode ***print*** ha d'imprimir per consola tots els elements de la LinkedDeque.
- Es pot consultar el primer element de la cua amb ***getFront***.
- Es pot consultar el darrer element de la cua amb ***getBack***.
- El destructor elimina tots els elements de la cua i la deixa buida.

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

Implementeu inicialment el TAD LinkedDeque amb element de tipus int, a l'especificació teniu Patient per tal de mostrar-vos el que heu d'incorporar com a element a l'exercici 3.

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

Es vol realitzar la gestió d'una cua de pacients que arriben a les urgències d'un hospital. Els pacients tenen 4 dades, un identificador, un nom, un cognom i un estat. Per exemple:

CACH010321001,Carlos,Charles,OK

JOJO120315002,Jose,Josua,NOT_OK

PEPE150690003,Pepe,Pez,OK

MIMI240598004,Mireia,Mir,NOT_OK

Utilitzarem el nostre TAD LinkedDeque implementat a l'exercici 2. Per a aquest exercici us caldrà implementar una classe **Patient** que contindrà la informació del pacient descrita anteriorment. A més a més, caldrà modificar el DoubleNode per a que guardi elements de tipus **Patient**. La lògica d'encuar i treure elements de la LinkedDeque hauria de romandre igual.

Utilitzeu el TAD LinkedDeque implementat a l'exercici 2 i adaptat per tenir pacients, per crear un programa que contingui les següents opcions de menú.

1. Llegir un fitxer amb les entrades de pacients
2. Imprimir la cua de pacients
3. Eliminar el primer pacient de la cua
4. Eliminar el darrer pacient de la cua
5. Inserir **n** entrades de pacients des de teclat (0 per finalizar)
6. Consultar el primer pacient de la cua
7. Consultar el darrer pacient de la cua
8. Sortir

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 pacient i els camps estan separats per una coma. Teniu un exemple a l'inici de l'explicació de l'exercici. Els pacients que tenen un estat OK, es posen al final de la cua. Els pacients que el seu estat és NOT_OK, es posen a l'inici de la cua.

3. Lliurament

Implementar els exercicis en C++ i usar el **NetBeans 8.2 amb 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 nom i cognoms de la persona que ha fet la pràctica i el número de la pràctica com a nom de fitxer. Per exemple, **GrupA_BartSimpson_P2.zip**, on P2 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 persones implicades (tant la que ha copiat com la que ha deixat copiar).

4. Planificació

Per aquesta pràctica disposeu de tres setmanes.

Els professors us proposem la següent planificació:

- **Setmana 1 (del 8 al 12 de març)**
  - Implementació de l'exercici 1
- **Setmana 2 (del 15 al 19 de març)**
  - Implementació de l'exercici 2
  - Tasca al campus virtual per lliurar exercici 1 i 2 (tot el que tingueu fet, si no heu acabat també feu el lliurament)
- **Setmana 3 (del 22 al 28 de març)**
  - Implementació de l'exercici 3
  - Tasca al campus virtual per lliurar exercici 1, 2 i 3

El lliurament final de la pràctica serà el **dia 28 de març de 2021** al campus virtual. Recordeu que s'han de lliurar tots els exercicis en el fitxer .zip al campus virtual.