

Pràctica 2. Estructures encadenades: pila, cua i llista

1. Introducció

Objectius: Entendre les estructures encadenades i el cost computacional associat a cadascuna.

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

2. Enunciat

Exercici 1. Definir i implementar el TAD **LinkedQueue** d'enters (Cua d'enters amb una representació encadenada) i el TAD **Node**. A partir de la cua encadenada definida simula el temps d'espera dels clients en un cinema amb el següent algorisme:

```
LinkedQueue cua;

cout<< "La cua és buida? "<<cua.empty()<<endl;






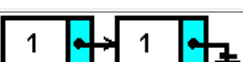
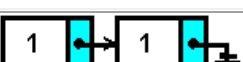

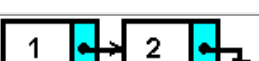


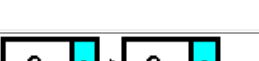




int item;
int minut;
int k;

int quiArriba[]={1,1,3,2,2,1,1,3,3,3}; // substituir quiArriba per un random
for (minut=0; minut<10; ++minut){      // substituir per 100 minuts
    k=quiArriba[minut];
    if (!cua.empty()) {
        item = cua.first();              // retorna el front de la cua
        cout << "El primer element de la cua és: "<<item << endl;
        cua.dequeue();
        cout << "El desencuem"<<endl;
    } //si la cua no és buida, treure un client
    if (k==1) {                          //Si k és 1 afegir un client a la cua
        cout<<"Encuem "<<minut<<endl;
        cua.enqueue(minut);
    }
    else if (k==2){                      //Si k és 2 afegir dos clients a la cua
        cout<<"Encuem "<<minut<<endl;
        cout<<"Encuem "<<minut<<endl;
        cua.enqueue(minut);
        cua.enqueue(minut);
    }
    else if (k==3) {                    // Si k és 3 no afegir ningú a la cua
    }

cua.show(); // Imprimeix la cua per pantalla
}
```

Estructura de Dades: Pràctica 2

Un exemple del funcionament de l'algorisme anterior seria el que es mostra al següent esquema per als minuts 0, 1, 2, 3 i 4 amb l'array quiArriba inicialitzat amb `quiArriba={1,2,2,1,3}`.

minut	empty()	k	operacions	Diagrama corresponent
0	true	1	no dequeue ()	
			enqueue(0)	
1	false	2		
			dequeue()	
			enqueue(1)	
			enqueue(1)	
2	false	2		
			dequeue()	
			enqueue(2)	
			enqueue(2)	
3	false	1		
			dequeue()	
			enqueue(3)	
4	false	3		
			dequeue()	
				

Estructura de Dades: Pràctica 2

Es recomana que el TAD **Node** tingui encapsulada la seva informació. Aquest TAD conté com a mínim dos atributs: `element` on es guarda l'element a inserir a la cua i `next` que és l'apuntador al següent node. 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
- **getNext**: retorna l'adreça del següent node
- **setNext**: modifica l'adreça de next per l'adreça rebuda com a paràmetre
- **getElement**: retorna l'element que hi ha guardat al node

Programa el codi necessari perquè el main calculi també el nombre total de clients atesos, la mitjana de temps d'espera de tots els clients i el temps que ha trigat el client amb més temps a la cua.

Es recomana que abans de programar calculis tu els valors per a cada minut i cada k donats i quan comencis a programar posis missatges per pantalla en el programa per anar veient els diferents valors.

Minut	K	Cua	Temps_espera	Max_temps	Mitjana_temps	Clients_atesos
0	1	[0				
1	1	[1	1	1	1	1
...
10

Un cop hagi comprovat que tot funciona, genera els valors de k aleatòriament amb les següents funcions de la llibreria random (hauràs d'afegir `#include <random>`):

```
uniform_int_distribution<> u(1,3); //u és un enter que pot prendre els valors 1, 2 i 3
default_random_engine e;          //e és un generador de nombres random
unsigned k;                        //k és l'enter que necessitem
k=u(e);                           //k serà un nombre aleatori entre 1 i 3
```

I quan hagi comprovat que tot funciona, augmenta el nombre de minuts a 100. Extreu els resultats i comenta'ls a la memòria de la pràctica. A més a més, indica a la memòria quin és el cost computacional de cadascuna de les operacions que has implementat en el TAD LinkedList.

Exercici 2. Definir i implementar el TAD **NodeList** com a llista d'enters amb encadenaments dobles i una representació lògica amb punt d'interès i el TAD **DoubleNode**. L'especificació del TAD NodeList és la següent:

- **Constructor**: construeix la llista sense rebre cap paràmetre. Inicialitza la llista a buida
- **Destructor**: destrueix tots els elements de la llista
- **empty**: retorna un booleà indicant si la llista està buida, cert si està buida, fals en cas contrari
- **size**: retorna el nombre d'elements que hi ha actualment a la llista
- **get**: retorna l'element del node on està situat el punt d'interès

Estructura de Dades: Pràctica 2

- **insert**: afegeix un nou node davant del punt d'interès amb les dades de l'element que rep com a paràmetre. El punt d'interès es queda al mateix node
- **remove**: elimina el node on està situat el punt d'interès. El punt d'interès quedarà situat en el següent node. No retorna l'element, només elimina el node
- **modify**: modifica el valor del node on està situat el punt d'interès amb l'element que rep com a paràmetre
- **begin**: situa el punt d'interès al primer node
- **end**: retorna cert si el punt d'interès ha arribat al darrer node. En cas contrari, retorna fals
- **next**: situa el punt d'interès al següent node
- **show**: mostra el contingut de la llista per pantalla. El punt d'interès no s'ha de moure del node on estigui situat

Punts a considerar en la implementació del TAD NodeList:

- Les excepcions més importants a controlar són:
 - No eliminar nodes si la llista està buida
 - No consultar elements si la llista està buida
- Es poden usar capçaleres, si ho considereu oportú
- Podeu usar templates per a que la llista i els nodes siguin genèrics
- Es recomana que el TAD **DoubleNode** tingui encapsulada la seva informació. Aquest TAD conté com a mínim tres atributs: `element` on es guarda l'element a inserir a la llista, `next` que és l'apuntador al següent node i `prev` que és l'apuntador al node anterior. L'especificació amb el mínim d'operacions necessàries del TAD DoubleNode és la següent:
 - **constructor**: construeix el node amb l'element que rep com a paràmetre
 - **getNext**: retorna l'adreça del següent node
 - **getPrev**: retorna l'adreça de node anterior
 - **setNext**: modifica l'adreça de next per l'adreça rebuda com a paràmetre
 - **setPrev**: modifica l'adreça de prev per l'adreça rebuda com a paràmetre
 - **getElement**: retorna l'element que hi ha guardat al node

A més a més, contesta a les següents preguntes: Com has implementat el TAD NodeList? Has hagut de realitzar alguna operació addicional a l'especificació en el TAD NodeList o en el TAD DoubleNode? Quin és el cost computacional de cadascuna de les operacions que has implementat en el TAD NodeList?

Exercici 3. Definir i implementar el TAD **OrderedList**, que hereta del TAD NodeList i que modifica el mètode insert a l'especificació:

- **insert**: rep per paràmetre un element i l'inserta ordenadament a la posició que li correspon dins de la llista. S'ordenarà la llista de menor a major. El punt d'interès no es mourà.

A més a més, contesta a les següents preguntes: Com has implementat el TAD OrderedList? Has hagut de realitzar alguna modificació al TAD NodeList? Quin és el cost computacional de la teva implementació per afegir i eliminar elements en el TAD OrderedList?

Estructura de Dades: Pràctica 2

Exercici 4. Fer un **main.cpp** que treballi amb els TADs **NodeList** i **OrderedList** i que faci les següents operacions:

- A partir d'una entrada per consola d'una seqüència de números, on l'últim element serà el número 99, creï una llista i una llista ordenada en les que insereixi aquests nombres per ordre d'arribada (el 99 no s'afegeix a cap de les llistes).
- Un cop creades les llistes, mostri els continguts de les dues llistes creades
- Demani un nombre a l'usuari, el busqui a les dues llistes, i mostri els nodes posteriors al nombre donat a cadascuna de les llistes
- Esborri les llistes i alliberi tots els recursos assignats

Un exemple d'execució del programa podria ser:

Introdueix una seqüència de nombres acabada per 99:
10 5 2 7 15 20 12 99
Llista: 5 2 7 15 20 12 10
Llista ordenada: 2 5 7 10 12 15 20
Introdueix un nombre a cercar:
15
Nombres posteriors a 15 a la llista: 20 12 10
Nombres posteriors a 15 a la llista ordenada: 20
Llista esborrada i recursos alliberats
Llista ordenada esborrada i recursos alliberats
Llista:
Llista ordenada:

Modificar el **main.cpp** anterior per a què permeti l'entrada a través d'un fitxer

Un exemple d'execució del programa podria ser:

Vols introduir les dades per consola o per fitxer (C/F):
C
10 5 2 7 15 20 12 99
Llista: 10 5 2 7 15 20 12
...

O

Estructura de Dades: Pràctica 2

Vols introduir les dades per consola o per fitxer (C/F):

F

Es llegirà el fitxer dades.txt

Llista: 10 5 2 7 15 20 12

...

En el cas de llegir les dades des de un fitxer no hi haurà el delimitador de final de seqüència (el 99). Si existeix el 99 a la llista, aquest s'inclou a la llista.

3. Lliurament

A partir de la descripció del problema, es demana:

- Escriure una memòria explicativa que inclogui els següents punts:
 - Portada amb: número de pràctica i títol complet, nom, cognom, NIUB de cada membre del grup, professor/a de pràctiques, número de parella assignat i grup de pràctiques al que assistiu (A, B, C, F)
 - Comentaris de cada exercici: observacions, decisions preses i resposta a les preguntes plantejades, si n'hi ha.
- La memòria s'entregarà en un document memoriaPX.pdf on X és el número de la pràctica.
- Implementar els exercicis en C++ . Lliurar el codi C++ corresponent als vostres exercicis en una carpeta anomenada codi, amb una subcarpeta per a 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 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 P2 indica que és la "pràctica 2". El fitxer ZIP inclourà: la memòria i 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

Lab6 i Lab7. Abril 2016

Per aquesta pràctica els professors proposen la següent planificació:

- **Setmana 1**
 - Implementació Exercici 1 i comentar el codi
 - Realització de la part de la memòria corresponent a l'exercici 1
- **Setmana 2**
 - Implementació Exercici 2 i comentar el codi
 - Implementació Exercici 3 i comentar el codi

Estructura de Dades: Pràctica 2

- Implementació Exercici 4 i comentar el codi
- Realització de la memòria completa de la pràctica

Cada setmana s'obrirà una tasca al campus virtual per dipositar la feina feta fins aquell moment.

Lliurament:

- Grups de dijous (B,C,D i F): dia 13 d'abril de 2016
- Grup de dimecres (E): dia 26 d'abril de 2016