Estructura de Dades

Pràctica 2: Estructures enllaçades: Estructures encadenades: pila, cua i llista

UNIVERSITAT DE BARCELONA

Blai Ras Jimenez i Albert Morales, Parella 5

Professor: Pere Hierro, Grup D

1. Exercici 1

En aquesta primer entrega es demana implementar el primer exercici de la pràctica. Aquest primer exercici consisteix en implementar un TAD LinkedQueue d'enters. En la pràctica anterior ja vam fer un TAD Queue, però aquest cop, serà de tipus encadenada, és a dir, una cua amb nodes.

Per fer l'exercici més entretingut, es realitza aquesta cua perquè simuli el temps d'espera dels clients en un cinema, és a dir, mitjançant la codificació d'una LinkedQueue estàndard, es modifica el main amb un algorisme que simula aquesta cua del cinema.

Es demanen els següents mètodes, que tota LinkedQueue conté:

- ***Int Size():*** retorna un enter corresponent al número d'elements afegits a la cua
- ***Void enqueue(int x):*** mètode que afegeix un enter a la cua passat per paràmetre
- ***Void dequeue():*** mètode que treu el primer element afegit a la cua
- ***Void show():*** mètode que imprimeix per pantalla els elements continguts en la cua
- ***Bool full():*** booleà que serà cert si la cua està plena
- ***Bool empty():*** booleà que serà fals si la cua està buida
- ***Int first():*** mètode que retorna el primer element afegit a la cua

Per altra banda, al ser una cua encadenada, es demana la implementació dels nodes, amb els seus corresponent mètodes:

- ***Node* getNext():*** mètode que retorna el punter al següent element (next)
- ***Void setNext(Node* i):*** mètode que assignarà el node que apunta al següent element (next) la variable i de tipus punter a Node
- ***Int getElement():*** mètode que retornarà l'enter corresponent al element que hi ha guardat al node

Un cop hem realitzat tots els mètodes, i comprovat que funcionen tal i com ho haurien de fer (amb quatre línies del tipus cua.enqueue(4) i cua.show), ens disposem a realitzar l'algorisme de la cua del cinema.

Cua del Cinema i implementació de Clients i Temps d'espera

Aquest algorisme simula els clients d'un cinema que rebem cada x minuts. Per fer-ho més còmode, anomenarem cada client com el minut en què ha arribat. Assignem quiArriba[] com la taula d'enters de corresponent al número de clients que venen per minut.

Per gestionar-ho, ens ajudem amb la variable k i 3 definicions: si k és 1, és que ens entra un únic client. Si k és dos, és que ens entren dos clients, i si k és 3, ens que no ens entra ningú. Per tant, amb un for que apliqui l'algorisme segons quans minuts està oberta la taquilla del cinema, anirem encuan i descuan clients per minut.

Per altra banda, es demana modificar el codi del main donat perquè retorni també el nombre total de clients atesos, la mitjana de temps d'espera i el temps que ha trigat el client amb més temps a la cua. Ho organitzem així:

- ***Clients_atesos:*** número de clients que hem encuat perquè ja hem atès.
- ***Temps_espera:*** número corresponent al temps per un client en qualsevol minut. Es considerarà que és <temps de l'anterior minut> + <temps del client en qüestió que porta a la cua>.
- ***Max_temps***: número corresponent al màxim de temps que pot estar un client a la cua.
- ***Mitjana:*** número corresponent a dividir temps_espera entre número de clients, és a dir, la mitjana de temps d'espera.

Es demana calcular en paper i llapis els resultats amb el main donat, on quiArriba és { 1,1,3,2,2,1,1,3,3,3} :

Minut	K	Cua	Temps_espera	Max_temps	Mitjana	Clients_atesos
0	1	0	-	-	0	-
1	1	1	1	1	1=1	1
2	3	NULL	1+1=2	1	2/2=1	2
3	2	3,3	2+0=2	1	2/2=1	2
4	2	3,4,4	2+1=3	1	3/3=1	3
5	1	4,4,5	3+2=5	2	5/4	4
6	1	4,5,6	5+2=7	2	7/5	5
7	3	5,6	7+3=10	3	10/6	6
8	3	6	10+3=13	3	13/7	7
9	3	NULL	13+3=16	3	16/8=2	8

Ho comprovem fàcilment amb l'Output del nostre main:

```
-----------------
Ara la k es 3
El primer element de la cua és: 6
El desencuem

-----------------
He atès 8 clients
El temps d'espera es de 16
El temps màxim d'espera es 3
La mitjana de temps d'espera es 2

-----------------
```

Fins ara, érem nosaltres qui triàvem quants clients ens venien per minut, amb la taula quiArriba. Ara, es demana generar aquets números amb un pseudorandom, una funció de c++ que genera números aleatoris dins d'un rang. En aquest cas, doncs, el rang serà entre 1 i 3:

```
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

for (int minut=0; minut<100; ++minut){ // substituir per 100 minuts

    //k = guiarriba[minut];

    k=u(e); //k serà un nombre aleatori entre 1 i 3
```

Un cop he comprovat que tot funciona, ens disposem a ampliar el rang de minuts a 100, l'equivalent a gairebé 1 hora i mitja de temps en la que la taquilla es oberta.

En el nostre cas, atenem a 90 clients, obtenim un temps d'espera final de 252, un temps màxim de 3 i per tant una mitjana de 2.8. Això significa que he atès a 90 clients en una mitjana de temps de 2.8 en la que el client que mes s'ha esperat ho ha fet durant un temps de 3, gens malament. Per desgràcia, no em pogut atendre a 4 clients, tal i com podem veure aquí:

```
He atès 90 clients
El temps d'espera es de 252
El temps màxim d'espera es 3
La mitjana de temps d'espera es 2.8

-----------------

96
98
99
99
RUN SUCCESSFUL (total time: 1s)
```

Cost computacional

Per últim, analitzo el cost computacional de cada mètode de LinkedQueue():

- Enqueue: O(1), és a dir, ordre constant, perquè no recorro la cua en cap moment. Faig un simple if (primer == NULL) i una assignació.

```
void LinkedQueue::enqueue(int x) {
    //New Node
    Node *node = new Node(x);

    node->info = x;

    node->next = NULL;

    if(primer == NULL){

        primer = node;
        mida++;

    }else{

        ultim->next = node;
        mida++;

    }

    ultim = node;

}
```

- Dequeue: O(1), és a dir, ordre constant, perquè tampoc recorro la cua i també faig ús d'un únic if amb dos assignacions en aquest cas, però el temps continua sent estimat a 1.

```
void LinkedQueue::dequeue() {
    Node *temp = new Node(primer->getElement());
    if(primer == NULL){

        cout<<"\nQueue is Emtpty\n";

    }else{

        temp = primer;

        primer = primer->next;

        mida--;
        //cout<<"The data Dequeued is "<<temp->info;

    delete temp;

    }
}
```

- Show():O(n), és a dir, ordre d'n, perquè faig us del iterador while, que té cost n, per recorre tots els nodes mentre aquest no sigui null, de manera que així mostro tota la informació de la cua per pantalla

```
void LinkedQueue::show() {

    Node *p ;

    p = primer;
    if(primer == NULL){

        cout<<"\nLa cua es buida!\n";

    }else{

        while(p!=NULL){

            cout<<endl<<p->info;

            p = p->next;

        }

    }
}
```

- Els mètodes full(), empty(), front() i size() tenen els quatre també ordre constant, ja que només retorno una variable o condició.

```
int LinkedQueue::size() {
    return mida;
}

bool LinkedQueue::empty() {
    return (mida == 0);
}

bool LinkedQueue::full() {
    return (mida = 100);
}

int LinkedQueue::first() {
    return primer->info;
}
```

SEGONA ENTREGA

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.

És a dir, en aquesta segona entrega implementarem una llista doblement encadenada amb punt d'interès. Aquesta representa es basa en un punter a una posició, anomenada Punt d'Interès, en la que es gestiona tots els mètodes de la Llista. En conseqüència, afegirem, esborrarem pel punt d'interès. Aquest punt també té els seus propis mètodes:

- Begin(): situa el PI al inici
- Next(): avança la posició una unitat
- End(): retorna booleà si el PI arriba al final

Per altra banda, la llista té els següent mètodes...:

- Genèrics:
  - Size(): mida del vector
  - Empty: cert si la llista està buida, fals si no.
- Consultors:
  - Get(): retorna l'element on es troba el PI

Anem a veure els mètodes que ens demanen. Al inserir, hem de gestionar totes les possibles combinacions que es puguin donar:

```
void NodeList::insert(int e) {

    DoubleNode* node = new DoubleNode(e); //Nou node

    if(pi == NULL) {
        pi = node;
        first = node;
        last = node;
    }else if (pi == last) {
        node->prev = last;
        node->next = NULL;
        last->next = node;
        last = node;
    }else if (pi == first) {
        node->prev = first;
        node->next = first->next;
        pi->next = node;
    }else{
        DoubleNode* n1 = pi;
        DoubleNode* n2 = pi->next;

        node->next = n2;
        node->prev = n1;
        n1->next = node;
        n2->prev = node;
    }

    listSize++;
}
```

En conseqüència, remove és molt semblant, ja que no és el mateix esborrar quan el punt de interès està al principi, al final, entre mig...

```
void NodeList::remove() {

    if (!empty()){
        if (pi == first) {
            DoubleNode* n1 = pi->next;
            pi = n1;
            first = n1;
            delete n1->prev;
            n1->prev = NULL;
        } else if (pi == last) {
            pi = pi->prev;
            last = pi;
            delete pi->next;
            pi->next = NULL;
        } else if(size() == 1) {
            first = NULL;
            last = NULL;
            delete pi;
            pi = NULL;
        } else {
            DoubleNode* p1 = pi->prev;
            DoubleNode* p2 = pi->next;
            p2->prev = p1;
            p1->next = p2;
            delete pi;
            pi = p1;
        }
        listSize--;
    }else {
        cout << "La llista ja esta buida LAMMER " << endl;
    }
}
```

Modify és bastant senzill, simplement, canvio l'element vell pel nou:

```
void NodeList::modify(int e) {
    this->pi->setElement(e);
}
```

Per fer el show, recorro la tota la Llista, amb la condició "While (node != NULL)" i després d'imprimir cada element assignaré el node al següent.

```
void NodeList::show() {

    DoubleNode *node = first;

    if(first == NULL){
        cout<<"[]";
    }else{
        while(node != NULL){
            cout<<node->getElement()<<endl;
            node = node->next;
        }
    }
}
```

Preguntes

1. 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?

   La implementació la he explicat a dalt. Sí, he fet una operació addicional que no em demanen en l'enciat: setElement(int x). Aquest mètode del Node, canvia l'element inserit en ell per un passat per paràmetre, és a dir, l'element de la Classe Node ara passa a ser l'enter passat per paràmetre. Ho utilitzo en el mètode de la LinkedList Modify(int x):

2. Quin és el cost computacional de cadascuna de les operacions que has implementat en el TAD NodeList?

   En notació Gran O, diem que l'espai usat per totes les operacions de la llista és d'O(1), excepte el mètode show(); el qual recorre la Llista per imprimir per pantalla la seva informació. Aquest mètode, per tant, diem que te cost O(n).

- No ens ha donat temps a fer els exercicis 3 i 4 degut a falta de temps per parcials, ens sap greu.