Solucions a problemes 2: TADs i estructures de dades lineals

Estructura de Dades - Curs 2019/2020
Grau d'Enginyeria Informàtica
Facultat de Matemàtiques i Informàtica

Objectiu

Aquests problemes estan enfocats en l'ús i implementacions de TADs (Tipus Abstracte de Dades), en particular Piles, Cues i Llistes tant en la seva estructura estàtica com dinàmica.

1 Problemes d'ús de TADs

1. Especifica un TAD que representi les últimes 10 trucades rebudes per un mòbil i que suporti de forma eficient els següents mètodes:

   ```
   crear() -> construeix el TAD
   novatrucada(String telefon) -> modifica el TAD afegint una nova trucada
   escriuretrucades() -> escriu totes les trucades per ordre segons l'hora
   que s'ha realitzat la trucada
   ```

   Quina estructura de dades és la més convenient? Amb quina implementació? Justifica la resposta basant-te amb els costos temporals dels mètodes i el cost espacial de l'estructura proposada.

2. Especifica i dissenya el TAD polígon amb les següents operacions:

   ```
   crea() -> Construeix el polígon buit inserir(Punt2D vertex) ->
   insereix un nou vèrtex en el polígon pel final tancar() ->
   insereix com a últim punt el primer de tots consulta(int i) ->
   consulta el vèrtex i-èssim
   ```

   Els polígons definits per aquest TAD no tenen definit "apriori" un nombre màxim de punts. Justifica les teves decisions de disseny i d'implementació.

3. Dissenya el TAD que sigui capaç d'avaluar expressions aritmètiques en notació inversa. Per exemple, 4 8 + 9 8 - * es equivalent a (4+8) * (9-8). Quina és l'estructura de dades que et permet dissenyar de forma eficient aquest TAD? Quina de les seves implementacions escolliries?

4. Els polinomis d'una variable amb coeficients enters tenen les següents operacions:

```
zero() -> crea el polinomi zero
afegeix(k: enter, n: enter) -> afegeix al polinomi k*xˆn
avalua(c: enter) -> real que es el valor del polinomi en el punt c
coef(n:enter)-> dóna el coeficient del terme de grau n
```

   Com a molt hi hauran MAXCOEF coeficients diferents de zero. Com implementaries aquest TAD? Justifica els costs de cada operació i el seu cost espacial.

5. Considera una matriu de n x m que representa un laberint. Cadascuna de les seves posicions pot estar etiquetada com: '**E**' (entrada), '**S**' (sortida), '**P**' (paret) i '**B**' (buit, es pot passar). Es comença de la casella '**E**' i només es consideren moviments verticals i horitzontals. Realitzeu un algorisme que trobi el camí des de l'entrada '**E**' fins a una sortida '**S**', o un camí - buit en el cas que no existeixi. Dissenyeu un TAD per definir el camí amb els mètodes que necessiteu. Amb quina estructura de dades et basaries? Justifica la teva resposta.

6. Un usuari A vol enviar un missatge M a un usuari B per Internet. L'usuari A trenca el missatge M en n paquets de dades, els numera de forma consecutiva i els envia per la xarxa. Quan els paquets arriben a l'usuari B, poden estar desordenats i l'usuari B ha d'ordenar-los per assegurar que llegirà el missatge original. Descriu un algorisme eficient per fer-ho. Amb quina estructura et basaries? Quin és el seu cost computacional?

7. Implementa la llista de favorits utilitzant un vector i no una representació encadenada. Quins costos temporals obtens?

8. Implementa un Comparador d'enters no negatius que determini el seu ordre basant-se en el nombre de 1's de la seva representació binària, és a dir, i ¡ j si el nombre de 1's de la representació binària de i és menor que el nombre de 1's de la representació binària de j. *Nota*: per desplaçar un bit cap a la dreta en C++ es pot fer servir l'operador >> i per obtenir el bit més baix d'un enter es pot aconseguir fent $bit = tmp\&1$.

9. Un aeroport vol fer una simulació del control del tràfic aeri que gestioni els esdeveniments d'enlairaments i d'aterratges dels avions. Cada esdeveniment té la informació de l'hora i els minuts en els què es produeix. Per realitzar la simulació, el programa necessita realitzar les següents operacions:

   - inserir(esdeveniment, temps)
   - extreure() retorna l'esdeveniment que té el temps més petit

   Les insercions no es fan per ordre de temps. Quina és l'estructura de dades més adient per realitzar aquestes operacions?

10. Un grup de nens i nenes juguen a un joc anomenat *Unmonopoly*, on a cada torn el jugador que té més diners ha de donar la mitat dels seus diners al jugador que té menys diners. Quina estructura de dades usaràs per implementar de forma efectiva aquest joc? Per què?

2 Problemes d'implementació d'estructures de dades lineals

11. Partint d'una STACK (`LinkedStack`) amb una representació amb enllaços simples que té un punter al top de la pila anomenat `first`. Implementeu una funció que desplaça el node que hi ha al top n posicions a la pila (sense usar les operacions del TAD pila), el nou top serà el node següent a l'anterior top. Assumiu que els nodes (`SingleNode`) tenen un punter public `next`. S'aconsella que useu dibuixos per justificar la vostra codificació. Quines excepcions has controlat?

```
1  void LinkedStack::moveTopNPositions(int n)
2  {
3   SingleNode * aux, * temp; aux= first; temp = aux;
4   if (first != null)
5   {
6           for (int i = 0; i< n; i++)
7           {
8                   if (temp->next == null) throw new
                       out_of_range("n massa gran");
9                   else temp = temp->next;
10          }
11          first = aux->next;
12          aux->next = temp->next;
13          temp->next = aux;
14  } else throw new out_of_range("Empty Stack");
15 }
```

12. Partim d'una STACK anomenada LinkedStack amb una representació amb enllaços simples que té un punter al top de la pila anomenat `front`. Implementeu el destructor de la pila utilitzant les funcions pròpies de la pila: size, empty, push, pop, top. A més a més, es demana que implementeu les operacions que heu cridat de la pila en la implementació del destructor de la pila. Assumiu que els nodes (`SingleNode`) estan definits tal i com s'especifica a continuació.

```
class SingleNode {
private:
        char element;
        SingleNode *next;
public:
        SingleNode(char e);
       ~SingleNode();
        const char & getElement()const;
        SingleNode * getNext()const;
        void setNext(SingleNode * elem);
};
```

S'aconsella que useu dibuixos per justificar la vostra codificació. Quines excepcions heu llençat en la vostra codificació?

```
template <class Element>
LinkedStack<Element>::~LinkedStack()
{
    cout << "Destroy LinkedStack" << endl;
    while (!this->empty())
        pop();
}

// Com a exercici queda pendent que implementeu les funcions
   empty i pop
```

13. Considera un nou tipus de dades `CuaDoble`. El seu funcionament és similar a una Cua, però en aquest cas es poden fer insercions, esborrats i consultes pels dos extrems de la cua. Dissenya i implementa aquest nou Tipus de Dades amb les seves funcions bàsiques. Considera diferents implementacions. Quines són les més adients? Justifica la teva resposta.

```
template <class Element>
void CuaDoble<Element>::enqueueFront(const Element &elem)
   throw(logic_error) {
    Node<Element>  *node = new Node<Element>(elem);
    node->setNext(front);
    this->front = node;
    if (size==0) this->back = node;
    size++;
}

template <class Element>
void CuaDoble<Element>::enqueueBack(const Element &elem)
   throw(logic_error) {
    Node<Element>  *node = new Node<Element>(elem);
    if (size>0) this->back->setNext(node);
    this->back = node;
    size++;
}

template <class Element>
const Element& CuaDoble<Element>::dequeueFront() throw
   (logic_error) {
    Node<Element> *aux = this->front;
    size--;
    if (size>0) this->front = this->front->getNext();
    else{
        this->front = NULL;
        this->back = NULL;
    }
    return aux->getElement();
}

```

```
30 template <class Element>
31 const Element& CuaDoble<Element>::dequeueBack() throw
      (logic_error) {
32     Node<Element> *aux = this->front;
33     size--;
34     if (size>0){
35         while (aux->getNext() != this->back){
36             aux = aux->getNext();
37         }
38         this->back = aux;
39         aux = aux->getNext();
40         back->setNext(NULL); //no es necessari
41     }
42     else{
43         this->front = NULL;
44         this->back = NULL;
45     }
46     return aux->getElement();
47 }
```

14. Considera un nou tipus de dades `SpecialPila`. Que com afegit a les funcionalitats de Pila conté els mètodes `topSecond()` i `popSecond()` que retornen respectivament el segon i penúltim element de la Pila. Implementa aquestes dues funcions de dues maneres:

    (a) Considera que el TAD té una implementació en array. Especifica la classe SpecialPila i implementa les dues funcions.

    (b) Considera que el TAD està definit amb una implementació amb encadenaments simples. Especifica la classe LinkedStack, la classe Node i implementa les dues funcions.

    Solució (a):

(a)
```
1  #include "SpecialPilaArray.h"
2
3  template <typename DataType>
4  SpecialPilaArray::SpecialPilaArray(int maxNumber) {
5      maxSize = maxNumber;
6      top = -1;
7      size = 0;
8      dataItems = new DataType[maxNumber]();
9  }
10
11 template <typename DataType>
12 void SpecialPilaArray::push(const DataType &newDataItem) {
13     dataItems[++top] = newDataItem;
14     size++;
15 }
16
17 template <typename DataType>
```

```
18 DataType SpecialPilaArray::pop() {
19     if (!this->isEmpty()) {
20         return dataItems[top--];
21     } else {
22         throw logic_error("trying␣to␣pop␣an␣empty␣list!");
23     }
24 }
25
26 template <typename DataType>
27 DataType SpecialPilaArray::topSecond(){
28     if(!isEmpty() && this->size >= 2){
29         return dataItems[top-1];
30     }else{
31         throw logic_error("␣topping␣a␣second␣with␣an␣less␣
              than␣2␣size.");
32     }
33 }
34
35 template <typename DataType>
36 DataType SpecialPilaArray::popSecond(){
37     if(!isEmpty() && this->size >= 2){
38         DataType r = dataItems[top-1];
39         top = dataItems[top-1];
40         size--;
41         return r;
42     }else{
43         throw logic_error("␣popping␣a␣second␣with␣an␣less␣
              than␣2␣size."); }
44 }
45
46 bool SpecialPilaArray::isEmpty() const {
47     return top == -1;
48 }
```

Solució (b):

(b)
```
1 template <class Element>
2 const Element& LinkedStack<Element>::topSecond() const
3 {
4     if (this->empty() && this->size()<2 ) throw
          out_of_range("LinkedStack<>::topSecond():␣empty␣
          stack");
5     return this->front->getNext()->getElement();
6 }
7
8 template <class Element>
9 void LinkedStack<Element>::popSecond()
10 {
11     if (this->empty() & this->size() < 3) throw
```

```
            out_of_range("LinkedStack<>::popSecond(): empty
            stack");
12      else
13      {
14          Node<Element> * aux_node = front->getNext();
15          front->setNext(aux_node->getNext());
16          delete aux_node;
17      }
18      this->num_elements--;
19  }
```

Recorda que s'han de gestionar les excepcions que es puguin produir. Addicionalment, implementeu un programa main.cpp que utilitzi el TAD LinkedStack.

```
1  int main() {
2      try{
3          LinkedStack<int> pila;
4          pila.push(2);
5          pila.push(3);
6          pila.push(4);
7
8          pila.popSecond();
9          cout << " top " << pila.top() << endl;
10         cout << " topSecond " << pila.topSecond() << endl;
11         cout << " num_elements " << pila.size() << endl;
12
13         pila.push(5);
14         pila.push(6);
15
16         int a = pila.top();
17         cout << " top " << a << endl;
18         cout << " num_elements " << pila.size() << endl;
19
20         int b = pila.topSecond();
21         cout << " topSecond  " << b << endl;
22         cout << " num_elements " << pila.size() << endl;
23
24         pila.push(7);
25         cout << " top " << pila.top() << endl;
26         cout << " topSecond " << pila.topSecond() << endl;
27         cout << " num_elements " << pila.size() << endl;
28
29         pila.pop();
30         cout << " top " << pila.top() << endl;
31         cout << " topSecond " << pila.topSecond() << endl;
32         cout << " num_elements " << pila.size() << endl;
33
34         pila.popSecond();
```

```
35          cout << "␣top␣" << pila.top() << endl;
36          cout << "␣topSecond␣" << pila.topSecond() << endl;
37          cout << "␣num_elements␣" << pila.size() << endl;
38
39      } catch (exception const& ex){
40          cerr<< "Exception␣" << ex.what() << endl;
41      }
42      return 0;
43  }
```

15. Considera un nou tipus de dades `SpecialCua`. Que com a afegit a les funcionalitats de Cua conté els mètodes `eliminaRepetits()` i `duplicaNodes()` que permeten eliminar els repetits i duplicar els elements respectivament. Implementa aquest nou TAD considerant dos tipus diferents de representació de la Cua:

    (a) Una representació amb encadenaments simples. Especifica la classe SpecialCua, la classe Node i implementa les dues funcions.

    (b) Una representació amb encadenaments dobles. Especifica la classe SpecialCua, la classe Node i implementa les dues funcions.

    Addicionalment, implementeu un programa main.cpp que utilitzi el TAD SpecialCua.

```
template <class Element>
void SpecialCua::eliminaRepetits(){
    Node<Element> *aux = this->front;
    while(aux != this->back){
        if (aux == aux->getNext()){
            if (aux->getNext() == this->back) this->back= aux;
            aux->setNext(aux->getNext()->getNext());
        }
        else{
            aux = aux->getNext();
        }
    }
}

template <class Element>
void SpecialCua::duplicaNodes(){
    Node<Element> * aux = this->front;
    for (int i = 0; i< size; i++){
        Node<Element> * newNode = new Node(aux->getElement());
        newNode->setNext(aux->getNext());
        aux->setNext(newNode);
        aux= aux->getNext()->getNext();
    }
}
```

NOTA: Considerem que eliminaRepetits és inversa a la funció duplicaNodes, per tant podem considerar que els repetits són contigus. Tanmateix, també podeu implementar el mètode de tal manera que elimini tots els nodes repetits, encara que no siguin contigus.

16. Considera un nou tipus de dades `MixedList`. Aquest TAD afegeix a les funcionalitats de List (llista enllaçada simple) un mètode `void cutAndJoin( int position )` que servirà per partir la llista per una posició passada per paràmetre i tornar-la a unir però en ordre invers. A tall d'exemple, donada la llista (1,2,3,4,5) si es crida el mètode amb el paràmetre 3, la llista haurà de quedar com (4,5,1,2,3). Implementa aquest nou TAD amb les funcionalitats de List i la de `cutAndJoin`.

```
template<class Element>
void MixedList<Element>::cutAndJoin(int n)throw (logic_error){
    for (int i = 0; i<=n; i++){
        this->gotoBeginning();
        Element item = this->getCursor();
        this->remove();
        this->gotoEnd();
        this->insert(item);
    }
}
```

17. Considera un nou tipus de dades `OrderedList`. Aquest TAD afegeix a les funcionalitats de List (llista amb punt d'interès amb una implementació amb array dinàmic) que els ítems són ordenats en ordre ascendent basat en les seves claus. Per a cada ítem de la llista, el ítem que el precedeix té la clau més petita i l'ítem que li segueix té la clau més gran. El punt d'interès (pdi) en una llista no buida sempre marca un dels ítems de la llista. S'itera per la llista usant operacions que canvien la posició del pdi. Implementa una `OrderedList` amb:

    - El seu constructor `OrderedList(int maxNumber = MAX_LIST_SIZE)` que crea una llista buida. Reserva suficient memòria per a una llista que contingui `maxNumber` ítems.
    - Un destructor que llibera l'espai de memòria usat per a la llista.
    - Un mètode `void insert( const DataType& newDataItem)throw (logic_error)`: que insereix el `newDataItem` a la seva posició corresponent a la llista. Si ja existeix un ítem amb la mateixa clau l'actualitza i deixa el pdi apuntant al `newDataItem`.
    - Un mètode `bool retrieve(const KeyType& searchKey, DataType& searchDataItem)const` que busca a la llista un ítem amb la clau `searchKey`. Si el troba deixa el pdi apuntant a dit ítem i el copia a `searchDataItem` i retorna true. I si no retorna fals i no modifica el pdi.
    - Un mètode `void remove ()throw ( logic_error )` que elimina l'ítem marcat pel pdi. Si la llista resultant no és buida el pdi apunta a l'ítem que segueix l'ítem eliminat i si era l'últim ítem de la llista, apunta el primer.
    - `void replace( const DataType& newDataItem )throw ( logic_error)` que substitueix l'ítem marcat pel pdi amb el `newDataItem` i mou el pdi a dit element.
    - `void clear()` que elimina tots el ítems de la llista.
    - `bool isEmpty()const` que retorna true si la llista és buida o false si no ho és.
    - `bool isFull()const` que retorna true si la llista està plena o false si no ho està.

- `void goToBeginning()throw (logic_error)`que mou el pdi al primer ítem de la llista.
- `goToEnd()throw (logic_error)`que mou el pdi a l'últim ítem de la llista.
- `goToNext()throw (logic_error)` si el pdi no és al final de la llista, el mou al següent ítem i retorna true, si no retorna false.
- `bool goToPrior()throw (logic_error)` si el pdi no és al començament de la llista el mou a l'ítem que el precedeix i retorna true, si no retorna false.
- `DataType getPdi ()const throw (logic_error)`que retorna el valor de l'ítem marcat pel pdi.
- `void showStructure()const` que treu per pantalla totes les claus dels ítems a la llista.

Per testejar el teu TAD Implementa el pas de missatges de l'exercici 6 basant-te en la definició de la classe paquet de *packet.cpp*.

Solució:

A continuació teniu algunes de les funcions que es demanen al problema.

```
#include "OrderedList.h"

template<typename DataType, typename KeyType>
OrderedList<DataType, KeyType>::OrderedList(int maxNumber) {
    this->dataItems = new DataType[maxNumber]();
}

template<typename DataType, typename KeyType>
void OrderedList<DataType, KeyType>::insert(const DataType
   &newDataItem) throw(logic_error){
    int index; //
    if(binarySearch(newDataItem.getKey(), index)){ // if found
        this->dataItems[index] = newDataItem; // replace

    } else{ // insert
        DataType next = this->dataItems[index+1];
        this->dataItems[index+1] = newDataItem;
        for(int i = index++; i < size; i++){
            this->dataItems[i] = next;
            DataType next = this->dataItems[i+1];
        }
    }
    cursor = index;
}

template<typename DataType, typename KeyType>
bool OrderedList<DataType, KeyType>::retrieve(const KeyType
   &searchKey, DataType &searchDataItem) {
    int index;
    if(binarySearch(searchKey, index )) {
        cursor = index;
        searchDataItem = this->dataItems[cursor];
```

```
31          return true;
32      }
33      return false;
34  }
35
36  template<typename DataType, typename KeyType>
37  void OrderedList<DataType, KeyType>::showStructure() const {
38      for(int i = 0; i < size; i++){
39          cout << this->dataItems[i].getKey());
40      }
41  }
42
43  template<typename DataType, typename KeyType>
44  bool OrderedList<DataType, KeyType>::isEmpty() const {
45      return size == 0;
46  }
47
48  template<typename DataType, typename KeyType>
49  bool OrderedList<DataType, KeyType>::isFull() const {
50      return size == maxSize;
51  }
52
53  template<typename DataType, typename KeyType>
54  void OrderedList<DataType, KeyType>::gotoBeginning()
      throw(logic_error) {
55      cursor = 0;
56  }
57
58  template<typename DataType, typename KeyType>
59  void OrderedList<DataType, KeyType>::gotoEnd()
      throw(logic_error){
60      cursor = maxSize;
61  }
62
63  template<typename DataType, typename KeyType>
64  bool OrderedList<DataType, KeyType>::gotoNext()
      throw(logic_error) {
65
66      if(cursor != size) cursor++;
67      return cursor != size;
68  }
69
70  template<typename DataType, typename KeyType>
71  bool OrderedList<DataType, KeyType>::gotoPrior()
      throw(logic_error) {
72      if( cursor != 0) cursor--;
73      return cursor != 0;
74  }
```

```
75
76 template<typename DataType, typename KeyType>
77 DataType OrderedList<DataType, KeyType>::getCursor() const
      throw(logic_error){
78     return this->dataItems[cursor];
79 }
```

3 Problemes d'ús d'estructures de dades lineals

18. Partint d'una `Queue<int> Q`, escriu un mètode en C++ que trobi el màxim element guardat a la cua. Només es poden usar les operacions de la cua: enqueue, dequeue, empty, etc. No es pot usar cap altra estructura de dades, només cues. La Queue d'entrada ha de quedar intacta després de buscar el màxim element de la cua.

```
1  int findMax(Queue<int> &Q)
2  {
3          int mida = Q.size();
4          int max = 0;
5          if (mida == 0) throw new out_of_range("Empty Queue");
6          else
7          {
8                  int element = 0;
9                  for (int i = 0; i< mida; i++)
10                 {
11                         element = Q.front();
12                         Q.dequeue();
13                         max = (element > max) ? element: max;
14                         Q.enqueue(element);
15                 }
16         }
17         return max;
18 }
```

19. Feu una mètode que donades dues piles (useu el TAD Stack) retorni una altra pila amb la fusió per dalt de les dues, i les originals buides. A tall d'exemple, la fusió per dalt de (1,2,3,4) i (5,6) (on 1 i 5 són els tops) és (1,5,2,6,3,4) i la de (1,2) i (3,4,5,6) (on 1 i 3 són els tops) és (1,3,2,4,5,6). Penseu en una solució iterativa. Heu de resoldre el problema usant les operacions del TAD.

```
1 Stack<int> fusioPerDalt(Stack<int> &pila1, Stack<int> &pila2){
2     Stack<int> aux;
3     Stack<int> resultat;
4     //fem la fusio. Aux sera la solucio al reves
5     while(not(pila1.isEmpty() && pila2.isEmpty())){
6         if(not pila1.isEmpty()){
7             aux.push(pila1.top());
8             pila1.pop();
```

```
        }
        if(not pila2.isEmpty()){
            aux.push(pila2.top());
            pila2.pop();
        }
    }
    //Invertim aux per tenir la solucio
    while(not aux.isEmpty()){
        resultat.push(aux.top());
        aux.pop();
    }
    return resultat;
}
```

20. Feu un mètode que solucioni el problema de les torres de Hanoi. És a dir, utilitzant tres piles (useu el TAD Stack), indiqui els passos que s'han de seguir per tal que donada dues piles buides i una tercera amb els números de l'1 a 'n ordenats de menor a major, s'acabi obtenint la mateixa seqüència en una altra de les dues piles seguint les normes següents: 1. Només es pot moure un nombre a la vegada d'una pila a l'altra. 2. En una mateixa pila, els nombres han d'estar ordenats de menor a major. A tall d'exemple, amb n = 2, hauria d'imprimir:
PAS 0: (1,2) () ()

PAS 1: (2) (1) ()

PAS 2: () (1) (2)

PAS 3: () () (1,2)

```
void towerOfHanoi_recursive(int &pas, int n, stack<int>*
   origen, stack<int>* desti, stack<int>* aux){
    if (n == 1){
            cout<<"PAS␣" << pas <<":␣";
            origen->showStructure();
            aux->showStructure();
            desti->showStructure();
            pas +=1;
            desti->push(origen->top());
            origen->pop();
            return;
    }
    towerOfHanoi_recursive(pas, n-1, origen, aux, desti);
    desti->push(origen->top());
    origen->pop();
    cout<<"PAS␣" << pas <<":␣";
    origen->showStructure();
    aux->showStructure();
    desti->showStructure();
    pas += 1;
    towerOfHanoi_recursive(pas, n-1, aux, desti, origen);
}
```

```
22 void towerOfHanoi(int n){
23     stack<int> *origen = new stack<int>();
24     stack<int> *aux = new stack<int>();
25     stack<int> *desti = new stack<int>();
26     for (int i=0; i<n; i++)
27             origen->push(n-i);
28     cout<<"PAS␣0:␣";
29     origen->showStructure();
30     aux->showStructure();
31     desti->showStructure();
32     int pas = 1;
33     towerOfHanoi_recursive(pas, n, origen, desti, aux);
34 }
```

21. Partint d'una `ArrayStack<char>` pila que guarda els caràcters d'una paraula. Escriu un mètode en C++ que indiqui si aquesta paraula és un palíndrom. L'especificació de la pila és la següent:

```
template <class E>
class ArrayStack {
public:
      ArrayStack();
      int size()const;
      bool empty()const;
      const E& top()const;
      void push(const E& e);
      void pop();
};
```

Noteu que la pila no disposa de constructor còpia. No es pot usar cap altra estructura de dades, només piles. La pila d'entrada ha de quedar intacta després de comprovar si el seu contingut és un palíndrom.

```
1  // NOTA: LA SOLUCIO NO ES UNICA. AQUI TENIU UNA POSSIBILITAT
2
3  bool esPalindrom(ArrayStack<char> &pila)
4  {
5      ArrayStack<char> pila2 = ArrayStack<char>();
6      ArrayStack<char> pila3 = ArrayStack<char>();
7      ArrayStack<char> pila_aux = ArrayStack<char>();
8
9      bool iguals = true;
10         while (not pila.empty()){
11             pila2.push(pila.top());
12             pila3.push(pila.top());
13             pila.pop();
14         }
```

```
15
16          while (!pila2.empty())
17          {
18              pila.push(pila2.top());
19              pila_aux.push(pila2.top());
20              pila2.pop();
21          }
22
23          while (!pila_aux.empty() and !pila3.empty() and iguals)
24          {   if (pila_aux.top() != pila3.top() ) iguals = false;
25              else
26              {   pila_aux.pop();
27                  pila3.pop();
28              }
29          }
30      return iguals;
31  }
```

22. La informació genètica codificada en una cadena d'àcid deoxyribonucleic (DNA) es guarda a les bases de purina i pirimidina (andina, guanina, citosina i timina) que formen la cadena. Els biòlegs estan molt interessats en les bases de la seqüència de DNA perquè determinen la funcionalitat de la seqüència.
    Per convenció, les seqüències de DNA són representades usant Llistes que contenen les lletres "A", "G", 'C' i "T". Implementa la funció següent que calcula una propietat d'un seqüència de DNA - el nombre de vegades que cada base apareix a la seqüencia.

    ```
    void countBases(List& dnaSequence, int& aCount, int& cCount, int& tCount, int&
    gCount )
    ```

    - on *dnaSequence*: conté les bases d'una seqüència DNA codificada amb els caràcters A, C, T i G.
    - *aCount, cCount, tCount, gCount*: el nombre de vegades que dita base apareix en la seqüència.

23. Els algoritmes genètics són uns dels algoritmes més utilitzats a l'hora de buscar solucions aproximades a problemes d'optimització i recerca. Un dels més comuns consisteix en traslladar els problemes a cadenes de 0 i 1, generar cadenes aleatòries, avaluar-les, seleccionar les millors i fer-les "reproduir". Per fer reproduir aquestes cadenes, s'utilitza un mètode que, donades dues de les llistes, les parteix per la meitat i les torna a ajuntar intercanviades, és a dir, donades les llistes (123456) i (abcdef) et retorna dues llistes (123def) i (abc456). Escriu una implementació d'aquest algoritme. Observa que has de generar les dues llistes "filles", per tant, en el teu programa també hauràs de seguir mantenint les llistes "mares".

24. Un problema clàssic de programació que pot ser resolt amb una pila és l'anomenat problema de les 8 reines. La pregunta és si és possible col · locar de manera segura (que no s'ataquin l'una a l'altra) 8 reines a un tauler d'escacs. La resposta és sí, així que la nova pregunta és com distribuir aquestes reines en el tauler. L'algoritme més senzill és col · locar una reina en un lloc potencialment segur i assegurar-se sí l'és. Si és un lloc segur deixar-la i passar a col · locar la següent reina; sinó l'és, treure-la i intentar col · locar-la a una altra banda. Si no

es troba cap lloc segur al tauler s'ha d'eliminar una de les reines anteriorment col · locades i continuar fins que totes les reines s'han col · locat a un lloc segur del tauler. Escriu una implementació de l'algoritme de les 8 reines. Has de mantenir la pila i guardar un històric d'on has col · locat cada reina.

25. Utilitza una cua per simular el flux de clients a una línia de caixa d'una botiga. Per tal de crear aquesta simulació heu de modelar el pas del temps i el flux de clients a la cua. Podeu modelar el temps utilitzant un bucle en el qual cada pas correspon a un conjunt de temps (1 min p.e.). Podeu modelar el flux de clients usant una cua en la qual cada ítem correspon a un client a la cua. Per tal de completar la simulació heu de conèixer el rati de clients que es posen a la cua i el ràtio amb el qual són servits a caixa. Imagina que la línia de caixa té les següents propietats:

    - Un client és atès i abandona la cua cada minut (assumint que hi ha com a mínim un client a la cua esperant a ser atès durant aquest minut).
    - Entre zero i un client s'afegeixen a la línia de caixa cada minut, on hi ha un 50 percent de probabilitat que no arribi cap client, un 25 percent que arribi un sol client, i un 25 percent de probabilitat que arribin 2 clients.

    Pots simular el flux de clients a la línia de caixa en un període d'n minuts usant l'algoritme següent:

    ```
    Inicialitza la cua a buida.
    Mentre la simulació no acabi
    Incrementa el temps de simulació en un minut.
    Si la cua no és buida, llavors agafa el client al front de la cua.
    Calcula un número aleatori k entre 0 i 3.
    Si k és 1, afegeix un client a la cua.
    Si k és 2, afegeix dos clients a la cua.
    Sinó ( si k és 0 o 3 ), no afegeixis res.
    ```

    Nota: Utilitzar el mètode rand és una bona manera de generar nombres pseudo-aleatoris. El podeu trobar a la llibreria ¡cstdlib¿.

26. Donat l'string d'entrada "*abc*", quines permutacions d'aquest string pot ser produït mitjançant un fragment de codi que consisteixi en aquest parell de línies?

    ```
    1 cin >> ch;
    2 permuteStack.push(ch);
    3
    4 ch = permuteStack.pop();
    5 cout << ch;
    ```

    on **ch** és un caràcter i **permuteStack** és una pila de caràcters? Podent repetir el parell de sentències tants cops com sigui necessari i en qualsevol ordre. Donat l'string "*abcd*", quines permutacions començant pel caràcter "*d*"poden ser generades a partir de les permutacions dels fragments de codi anteriors. Proveu-les.