UNIVERSIDAD NEBRIJA

Escuela Politécnica Superior

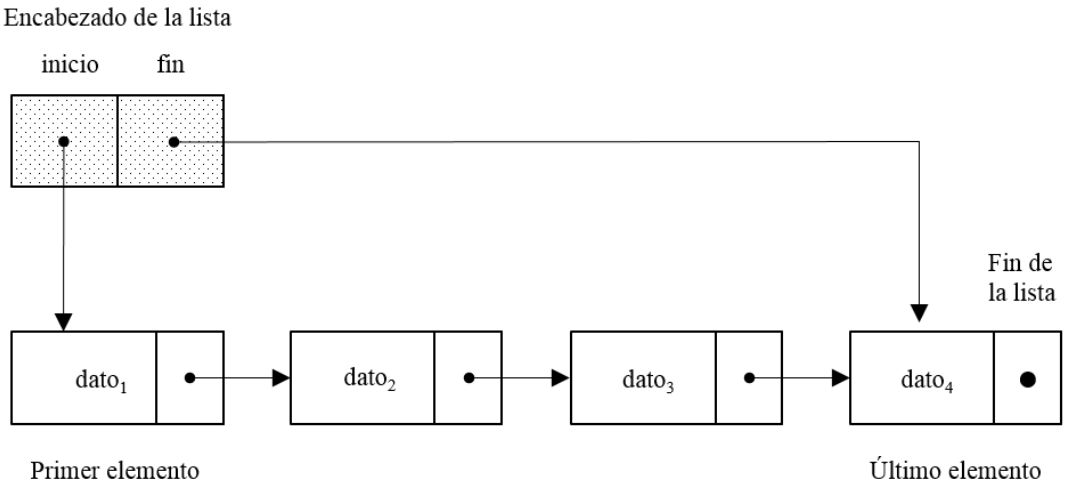
Estructuras de datos y algoritmos

Práctica 2

Una lista es un tipo abstracto de datos lineal. El TAD lista es una estructura de datos básica con múltiples aplicaciones. Una lista se puede utilizar para implementar estructuras de datos más complejas.

Las listas son estructuras de almacenamiento dinámicas, no tienen las limitaciones de los arrays. Una lista es una colección de nodos que almacenan datos y un puntero al nodo siguiente. Si el puntero al nodo siguiente es nulo, entonces se ha llegado al final de la lista.

El encabezado de la lista almacena un puntero al primer nodo y un puntero al último nodo de la lista.



La interfaz del TAD lista.

```
inicializa(L) Inicializa L a una lista vacía

  Precondición:  Ninguna
  Postcondición: Devuelve la lista L vacía

vacia(L) Devuelve verdadero si la lista L está vacía y falso en
         cualquier otro caso

  Precondición:  Ninguna
  Postcondición: true si la lista L está vacía y false e.o.c.

elementos(L) Devuelve el número de elementos de la lista

  Precondición:  Ninguna
  Postcondición: número de elementos de la lista

existe(L, x) Devuelve verdadero si el elemento x está en la lista y
             falso en cualquier otro caso

  Precondición:  Ninguna
  Postcondición: true si el elemento x está en la lista L y
                 false e.o.c.

insertaInicio(L, x) Inserta el elemento x en la primera posición de
                    la lista

  Precondición:  Ninguna
  Postcondición: El puntero inicio de la lista apunta al
                 nuevo nodo que almacena el elemento x

insertaFin(L, x) Inserta el elemento x en la última posición de la
                 lista

  Precondición:  Ninguna
  Postcondición: El puntero fin de la lista apunta al
                 nuevo nodo que almacena el elemento x

eliminaInicio(L, x) Elimina el elemento almacenado en la primera
                    posición de la lista y devuelve su valor

  Precondición:  Ninguna
  Postcondición: El puntero inicio de la lista apunta al nodo
                 siguiente del puntero inicio
```

```
eliminaFin(L, x) Elimina el elemento almacenado en la última
                 posición de la lista y devuelve su valor

  Precondición:  Ninguna
  Postcondición: El puntero fin de la lista apunta al nodo anterior
                 del puntero fin

elimina(L, x) Elimina el elemento x de la lista

  Precondición:  Ninguna
  Postcondición: La lista no contiene la primera ocurrencia
                 del nodo con el elemento x

toString(L) Devuelve una cadena con los elementos de la lista L

  Precondición:  Ninguna
  Postcondición: Ninguna

busca(L, x) Devuelve el puntero al nodo que almacena el valor x en
            la lista y NULL en cualquier otro caso

  Precondición:  Ninguna
  Postcondición: Devuelve el puntero al nodo que almacena el
                 valor x en la lista L, si no lo encuentra
                 devuelve NULL

elimina(L, p) Elimina el nodo p de la lista

  Precondición:  Ninguna
  Postcondición: La lista L no contiene el nodo p
```

Implemente el TAD lista utilizando las clases `NodoLista` y `Lista`.

```
class NodoLista {
public:

   int dato;
   NodoLista *siguiente;
};
```

La clase `Lista`.

```
#include <string>
#include "NodoLista.hpp"

class Lista {

public:

   Lista();
   ~Lista();
   bool vacia();
   int elementos();
   bool existe(int v);
   void insertaInicio(int v);
   void insertaFin(int v);
   int eliminaInicio();
   int eliminaFin();
   void elimina(int v);
   std::string toString(std::string s);

private:

   NodoLista *inicio, *fin;

   NodoLista* busca(int v);
   void elimina(NodoLista *p);

};
```

Modifique el TAD lista y defina las siguientes funciones:

```
Lista concatena(Lista c) {

   // concatena los nodos de la lista c a los nodos de la instancia

}

Lista invierte() {

   // devuelve una lista con los nodos de la instancia en orden
   // inverso

}
```