Prácticas de Estructuras de Datos
Grado en Ingeniería en Informática
Curso 2014/15

Práctica 2. Listas enlazadas

Sesiones de prácticas: 2

Objetivos

Implementar y utilizar un lista enlazada para una aplicación de radio de Internet

Estructura de datos a implementar: lista enlazadas

En esta segunda práctica implementaremos una lista enlazada (simple) con la siguiente interfaz:

```
template<typename T>
class ListaEnlazada {
  Nodo<T> *cabecera, *cola;
public:
  ListaEnlazada() : cabecera(0), cola(0) {}
  ListaEnlazada(const ListaEnlazada &l);
  ~ListaEnlazada();
  ListaEnlazada &operator=(ListaEnlazada &l);
  Iterador<T> iterador() { return Iterador<T>(cabecera); }
  void insertarInicio(T &dato);
  void insertarFinal(T &dato);
  void insertar(Iterador<T> &i, T &dato);
  void borrarInicio();
  void borrarFinal();
  void borrar(Iterador<T> &i);
};
```

Aplicación: Radionauta (v2)

Ahora sustituiremos la cola de peticiones que teníamos implementada mediante un vector dinámico por una lista enlazada. Además mejoraremos la búsqueda de canciones (opción 1) pudiendo localizar por código o subcadena en autor o título. Ejemplo:

```
C 212 <- Añadir a la lista la canción con código 212
A Beatles <- Buscar canciones con ‘‘Beatles’’ en autor
T Fire <- Buscar canciones con ‘‘Fire’’ en título
```

Si se usan las opciones de búsqueda por título o autor aparecerá una lista de resultados similar a esta:

```
A Charles
10 What'd I Say - Ray Charles
44 Georgia on My Mind - Ray Charles
161 I Can't Stop Loving You - Ray Charles
235 I Got a Woman - Ray Charles
377 Hit the Road Jack - Ray Charles
```

Y a continuación el usuario podrá escoger una seleccionando el código correspondiente:

```
C 235
Añadida: ''I Got a Woman - Ray Charles'' a lista de peticiones
```

Otra mejora será la implementación de un histórico de canciones pinchadas usando un vector dinámico. Cada vez que una canción sea pinchada se retirará de la lista de peticiones y se añadirá a este vector. Para mejorar la variedad musical de la emisora, no se admitirá una petición de una canción que esté en la lista de peticiones o haya sido una de las 100 últimas canciones pinchadas.

Estilo y requerimientos del código:

1. El código debe ser claro, tener un estilo definido y estar perfectamente indentado, para ello se pueden seguir algunos de los estilos preestablecidos para el lenguaje C++ (`http://geosoft.no/development/cppstyle.html`).
2. El código debe manejar excepciones siempre que éstas tengan sentido. En concreto en esta práctica se pueden utilizar para el manejo del fichero (fichero inexistente, etc), posibles errores al asignar memoria al vector o acceder a una posición del vector inexistente.

Exercise 2. Linked lists

Sesions: 2

Goals

Implement and use a linked list for implementing a request queue in an Internet radio.

Data structure to implement: singly linked lists

In this second exercise you must implement a singly linked list with the following interface:

```
template<typename T>
class LinkedList {
  Node<T> *head, *tail;
public:
  LinkedList() : head(0), tail(0) {}
  LinkedList(const LinkedList &l);
  ~LinkedList();
  LinkedList &operator=(LinkedList &l);
```

```
  Iterator<T> iterator() { return Iterator<T>(head); }
  void insertFirst(T &data);
  void insertLast(T &data);
  void insert(Iterator<T> &i, T &data);
  void eraseFirst();
  void eraseLast();
  void erase(Iterator<T> &i);
};
```

Test application: Radionauta (v2)

Now you must change your test program from exercise 1 to use a linked list for implementing the request queue, as deleting the first element in a dynamic vector is very inefficient. A second improvement is allowing searching for songs using different criteria. For instance:

```
C 212 <- Add to queue the song with code 212
A Beatles <- Search for songs with ‘‘Beatles’’ in the author field
T Fire <- Search for songs with ‘‘Fire’’ in the title field
```

After searching for author or title you should get a list of matching songs as this:

```
A Charles
10 What’d I Say - Ray Charles
44 Georgia on My Mind - Ray Charles
161 I Can’t Stop Loving You - Ray Charles
235 I Got a Woman - Ray Charles
377 Hit the Road Jack - Ray Charles
```

And then the user can add one of the songs to the request queue:

```
C 235
Added: ‘‘I Got a Woman - Ray Charles’’ to the request list
```

The last improvement is implementing an historic of played songs using a dynamic vector. Each time a song is played, it is deleted from the request queue and added to the historic vector. In order to improve the quality of the musical selection, a song request will is only accepted if it is not int the request queue and has not been one of the last 100 played songs.

Style and code requirements:

1. The source code has to be clear, with a coherent notation and properly indented. If possible follow the notation and suggestions given in the following document: `http://geosoft.no/development/cppstyle.html`.
2. Use exceptions in the template for notifying errors when required. For instance, accesses to invalid array positions. The template must have a robust implementation.