Taller preparatorio 3er Parcial

Prof. Luis E. Garreta U.

November 12, 2017

Para los siguientes puntos, tome los ejemplos de listas, tablashash y archivos visto en clase y el código y las diapositivas disponibles en el github del curso:

Dadas las siguientes implementaciones de Listas y TablaHash:

Listas Simplemente Enlazadas	Tabla Hash con vectores y listas La tabla hash está implementada con vectores y listas enlazadas para tratar colisiones (encadenamiento).

`struct Nodo;`
`  int info;`
`  Nodo *siguiente;`
`};`
 	
`class Lista;`
`private:`
`  Nodo *head, *tail;`
`public:`
`  Lista();`
`  void adicionar(int valor);`
`  void insertar(int pos, int valor);`
`  void mostrarse();`
`  void eliminar(int pos);`
`  int getElemento (int x);`
`  void setElemento (int valor, int x);`
`  int longitud ();`
`  bool vacia ();`
`  int buscarElemento (int valor);`
`  bool existeElemento (int valor);`
`};`
	`#include <list>`
	`#include <vector>`
	
	`class TablaHash {`
	`  private:`
	`  vector <list <int> > buffer;`
	`  int longitudActual;`
	`  int funcionHash (int & x);`
	`  void rehash ();`
	
	`  public:`
	`  TablaHash (int size);`
	`  bool adicionar (int x) ;`
	`  bool eliminar (int & x) ;`
	`  bool existe (int x);`
	`  void limpiar ();`
	`  void mostrarse ();`
	`};`

1 Listas Simplemente Enlazadas y Archivos

Dada la implementación de listas mostrada anteriormente:

1. Adicione a la clase *Lista* (implemente en C++) un método que guarde todo el contenido del valor de los nodos en un archivo. El método ingresa como parámetro el nombre del archivo, así: ***void guardarse (string nombreArchivo);***

2. Adicione a la clase *Lista* (implemente en C++) un método que cargue los elementos que están en un archivo tipo texto a los nodos de la lista. El método toma como parámetro el nombre del archivo desde donde se cargan los datos, así: ***void cargarse (string nombreArchivo)***;

2 Tablas Hash y Archivos

Dada la implementación de tablas hash mostrada anteriormente:

1. Adicione a la clase TablaHash un método que guarde los datos de la tabla hash a un archivo tipo texto. El método solo recibe como parámetro el nombre del archivo, así: ***void guardarse (string nombreArchivo);***

2. Adicione a la clase TablaHash un método que cargue los datos desde un archivo tipo texto a la tabla hash. El método solo recibe como parámetro el nombre del archivo, así: ***void cargarse (string nombreArchivo);***

3 Archivos Sequenciales Indexados

Realizar una función que dado como entrada tres parámetros tipo string: una palabra, un nombre de archivo texto de indices, y un nombre de archivo binario de registros, busque dentro del archivo texto de índices el índice que corresponde a la palabra y después use este índice para calcular la posición del registro de esa palabra dentro del archivo binario de registros y retorne el significado. El prototipo de la función y la descripción detallada de los archivos se presentan a continuación:

```
string buscarSignificado (string palabra, string nombreArchivoIndices, string nombreArchivoRegistros) {
  ...
  ...
  ...
  return significado;
|
```

La estructura en memoria secundaría de un diccionario inglés-español está compuesta sde dos archivos, uno de tipo texto (llamado "indices.txt"), y el otro de tipo binario (llamado "datos.dat"). El primer archivo, de tipo texto, contiene los índices con dos elementos: indice y palabra, cuyo índice indica la posición del registro dentro del archivo binario de la palabra, como se muestra a continuación:

```
0 apple
1 back
2 car
...
...
18423 running
18424 while
18425 zoo
```

El segundo archivo, de tipo binario, contiene los registros completos de la palabra y sus posibles significados. La estructura del registro es la siguiente:

```
typedef struct Estudiante {
    char palabra [10];
    char significado [50];
};
```

y el archivo contiene los registro de las 18426 palabras (incluyendo el zero) que están indexadas en el archivo de índices descrito anteriormente. Cada registro contiene en total 60 bytes, es decir existen alrededor de 18426 bloques de datos binarios que contienen la información de una palabra y su significado, parecido a lo que se muestra a continuación:

Posición	**Bloque de 60 bytes**
0	Bloque con palabra="apple" y significado="manzana"
60	Bloque con palabra="back" y significado="parte trasera, espalda"
120	Bloque con palabra="car" y significado="carro"
....	
...	
	Bloque con palabra="back" y significado="parte trasera"
1105380	Bloque con palabra="running" y significado="corrida, carrera, marcha"
1105440	Bloque con palabra="while" y significado="mientras, corto tiempo"
1105500	Bloque con palabra="zoo" y significado="zoológico"

4 Batalla Naval y Archivos

1. Adicione un método a la clase *Jugador* que guarde línea por línea en un archivo texto la información de cada una de las naves. Por ejemplo, para el caso, en que al jugador se le asignen 3 barcos, 2 submarinos, y 1 portaaviones, así podría quedar el archivo:

```
barco horizontal 2  b
barco horizontal 2  b
barco vertical   2  b
submarino vertical 3 s
submarino horizontal 3 s
submarino horizontal 3 s
portaaviones vertical 4 s
```