Universidad Simón Bolívar
Departamento de Computación
y Tecnología de la Información
Laboratorio de Computación II
Ene-Mar 2019

> 308 *El no saber comercial es distinguido.*—El vender su mérito lo más caro posible o negociarle con usura, a título de profesor, funcionario o artista, pone al talento o al genio a la altura de un tendero. No está bien querer ser demasiado *hábil* con la sabiduría.
>
> Aurora, Federico Nietzsche.

Lab. 5—Sem. 6—Listas Enlazadas de Cadenas de Caracteres.

En este laboratorio definiremos una **Lista de Cadenas de Caracteres** como un apuntador a un **nodo de tipo Cab** que contiene sólo un campo que es un apuntador a un nodo de tipo **Nodoc**. Nodoc es a su vez un registro con dos campos: *un apuntador a una cadena de caracteres y un apuntador a un Nodoc.*
A continuación se da un file que describe los primeros pasos:

```
/*
   Se implementa una lista de cadenas de caracteres.
   La Lista se define como un apuntador a un nodo
   Cab (por cabeza) que sólo tiene un apuntador
   de nombre alpha a un nodo de
   tipo Nodoc que contiene un apuntador a char y
   y un apuntador a un nodo de tipo Nodoc.
   La lista es vacía cuando alpha es NULL.

*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

///Modelo de Representación:

typedef struct nodoc {
   char* key;
   struct nodoc* sig;
} Nodoc;

typedef struct cab {
   //int size; podria ser conveniente.
   Nodoc* alpha;
} Cab;

typedef Cab* Lista;

///Invariante de representación:
///L es vacia ssi L->alpha == NULL


///Operaciones del tipo abstrato Lista

///Constructores:

///Recibe una cadena y a donde apunta
Nodoc* newNodoc(char* c, Nodoc* sig){
   Nodoc* p = malloc(sizeof(Nodoc));
   p->key = malloc(sizeof(c));
   strcpy(p->key, c);
   p->sig = sig;
```

```
   return p;
}

///Crea una lista vacía L con L->alpha == NULL
Lista newLista(){
   Lista L = malloc(sizeof(Cab));
   L->alpha = NULL;
   return L;
}

void addCab(char* k, Lista L){
   Nodoc* p = newNodoc(k, L->alpha);
   L->alpha = p;
   //L->alpha = newNodoc(k, L->alpha); ???
}

void addCol(char* k, Lista L){
   if(L->alpha == NULL){
      L->alpha = newNodoc(k, NULL);
   }
   else{
      Nodoc *p = L->alpha;
      while(p->sig != NULL) p = p->sig;
      p->sig = newNodoc(k, NULL);
   }
}

///Observadores:

int esVacia(Lista L){
   return L->alpha == NULL;
}

int estaEn(char* k, Lista L){
   Nodoc* p = L->alpha;
   while(p != NULL && strcmp(p->key, k) != 0) p = p->sig;
   return p != NULL;
}

int estaEnRAux(char* k, Nodoc* n){
   if(n == NULL) return 0;
   else if(strcmp(n->key, k) == 0) return 1;
        else return estaEnRAux(k, n->sig);
}

int estaEnR(char* k, Lista L){
   return estaEnRAux(k, L->alpha);
}

void printLista(Lista L){
   if (L->alpha == NULL) printf("\n[]");
   else {
      Nodoc* p = L->alpha;
      printf("\n[");
      while( p->sig != NULL){
```

```
         printf("%s, ", p->key);
         p = p->sig;
      }
      printf("%s]", p->key);
   }
}

///Destructores...

void delete(char* key, Lista L){
   if( L->alpha == NULL) return;
   Nodoc* p = L->alpha;
   if(strcmp(p->key, key) == 0){
      L->alpha = p->sig;
      free(p);
   }
   else {
      while(p->sig != NULL && strcmp(p->sig->key, key) != 0)
         p = p->sig;
      if(p->sig != NULL) {
         Nodoc* q = p->sig;
         p->sig = q->sig;
         free(q);
      }
   }
}

///programa de pruebas...
int main(void){
   Lista L = newLista();
   printLista(L);
   addCab("Zixia", L);
   addCab("Amor", L);
   addCab("Amigo", L);
   addCol("Amargura", L);
   delete("Amor", L);
   printLista(L);
   if(estaEn("casa", L)) printf("\nEsta");
   else printf("\nNO esta");
}
```

Tareas

1. *Clonar.* Escriba una función que permita clonar una Lista de cadenas de caracteres, esto es, producir otra lista que contenga los mismos elementos de la lista en el mismo orden. `Lista clonar(const Lista L);`
2. *Tamaño de una Lista.* Escriba una función que permita determinar el tamaño de una Lista. `int size(const Li`
3. *Mezcla de Listas.* Escriba un una función que reciba dos lista ordenadas de forma no decreciente y retorne una nueva lista ordena de forma no decreciente que contenga exactamente todos los elementos de las dos lista que recibe. `Lista merge(const Lista L, cons Lista M);`
4. *Inserción Ordenada.* Escriba una función que reciba una cadena de caracteres x y una lista ordenada L e inserte el elemento x en la liata L de tal forma que la lista final esté ordenada. `void insertOrd(int x, cons Lis`

5. Escriba una función que permita ordenar los elementos de una Lista de caracteres.

6. Halla el máximo de una lista no vacía. Debe retornar un apuntador al Nodo que contiene la primera ocurrencia del máximo.

7. Hallar la posición de un elemento x dentro de una lista. Debe retornar un apuntador NULO si no se encuentre, de lo contrario debe retornar la dirección de memoria del nodo donde se encuentra.