PROGRAMACION II

MATERIAL para TEORIA

LISTAS

Código Asignatura 6A4

Año 2017

Segundo Cuatrimestre

LISTAS SIMPLEMENTE ENLAZADAS

A partir de “nodos” dinámicos, implementados como variables de tipo registro, donde un campo de tipo puntero a registro contiene la dirección del próximo nodo, es posible implementar Pilas, Colas y Listas tomando como enlaces los vínculos en memoria dinámica.

Cada inserción implica crear y vincular nodos y cada eliminación desvincularlos devolviendo al heap los bytes que ocupa. La forma en que se modifican los enlaces o vínculos depende de la forma de operar de la estructura. No se considera la posibilidad de estructura llena, ya que ésta crece y decrece de acuerdo a los requerimientos.

TIPO LISTA SIMPLEMENTE ENLAZADA

```
typedef struct {
  char dato[15];
  struct nodo * sig;} nodo;
typedef struct nodo * TLista;
```

EJERCICIOS

1) Dada una lista simplemente enlazada de cadenas, retornar la cantidad que tienen longitud par
2) Dada una lista simplemente enlazada de cadenas, verificar si X está
3) Dada una lista ordenada simplemente enlazada de enteros, verificar si X está
4) Dada una lista de enteros modificar cada aparición de X por X+1
5) Insertar un dato en una lista ordenada simplemente enlazada de enteros
6) Dada una lista ordenada simplemente enlazada de enteros, eliminar X

TAREAS

7) Destruir una lista simplemente enlazada
8) Eliminar todas las apariciones de X de una lista simplemente enlazada de enteros
9) Eliminar todas las apariciones de X de una lista simplemente enlazada ordenada de enteros

LISTAS CON SUBLISTAS

1) Se tiene una lista L con la siguiente información:

- Codigo de Contenedor (no se repite)
- Destino
- Peso (en toneladas)

Además, una lista de barcos en puerto esperando embarcar los contenedores que están en la lista:

- Codigo de Barco (no se repite)
- Destino (puede repetirse)
- Capacidad
- Sublista de Contenedores
  - Codigo de Contenedor
  - Peso

Se pide: procesar la información de L y distribuir los contenedores en los barcos. Si alguno no pudiera ser embarcado por falta de capacidad en el destino o por falta de barcos a ese destino, quedará en L, de forma tal que al terminar el proceso queden en la misma los contenedores que no pudieron ser despachados.

TIPO DE DATOS

```
typedef struct {
  char codC[15], dest [15];
  float peso;
  struct nodo * sig;} nodo;
typedef struct nodo * TListaC;
```

```
typedef struct {
  char codC[15];
  float peso;
  struct nodito * sig;} nodito;
typedef struct nodito * SubLista;

typedef struct {
  char codB[15], dest[15];
  float capac;
  struct nodoB * sig;
  SubLista sub;} nodoB;
typedef struct nodoB * TListaB;
```

2) Se tiene una lista de clientes que registran pagos de un crédito con el siguiente diseño:

- Numero de Cliente (no se repite, ordenado ascendente)
- Total Credito, Total Adeudado (en $)
- Sublista de Pagos
  - Fecha (Ordenada descendente, no se repite)
  - Importe

Se pide:

a.- Dado un número de cliente *correcto*, una fecha y un importe, insertar el pago actualizando el valor adeudado.

b.- Dado un número de cliente y una fecha, eliminar el pago (si existe) actualizando el valor adeudado.

c.- Dado un número de cliente, eliminarlo de la lista

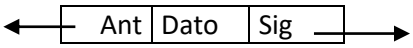
d.- Eliminar de la lista los clientes que ya no tienen deuda.

TIPO DE DATOS

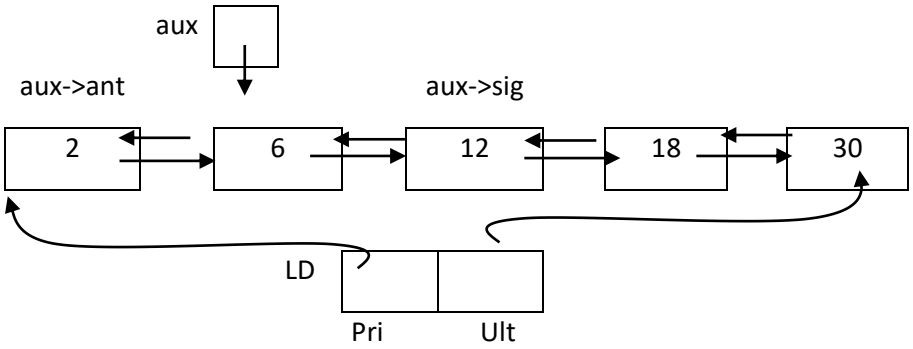
```
typedef struct nodito {
   char fecha[9];
   float imp;
   struct nodito * sig;} nodito;
 typedef struct nodito * SubLista;
typedef struct nodoC {
   int numC;
   float cred, deuda;
   struct nodoC * sig;
   SubLista sub;} nodoC;
typedef struct nodoC * TListaC;
```

LISTAS DOBLEMENTE ENLAZADAS

En esta estructura cada nodo almacena la dirección del nodo siguiente y del nodo anterior. Esto permite recorrer la lista en ambos sentidos.



Se debe mantener la dirección del primer nodo y del último para acceder a la lista, por el principio y por el final.



Ejercicio Se tiene una lista simplemente enlazada que almacena números enteros sin repeticiones. Armar una lista doblemente enlazada ordenada de forma ascendente.
A partir de la lista generada, eliminar nodos de tal modo que en la misma no aparezcan dos valores pares o impares consecutivos.

Ejemplo: 10 7 13 8 12 5 → 5 7 8 10 12 13 → 5 8 13

TIPO DE DATOS

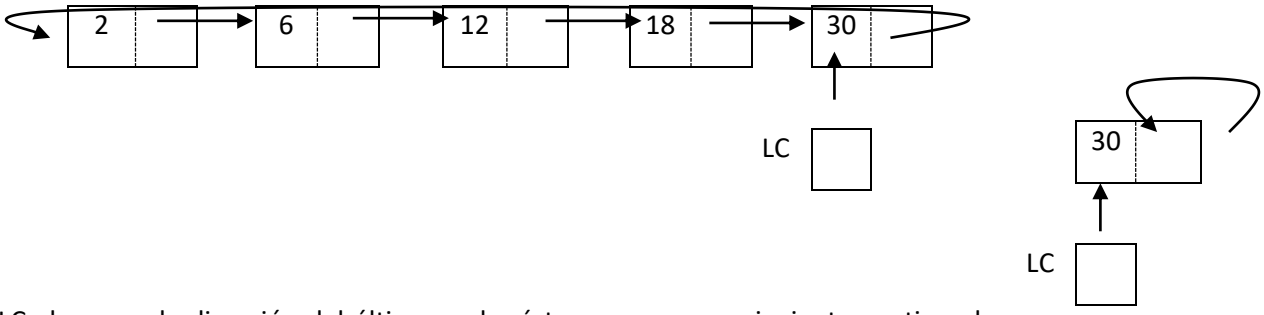
```
typedef struct {
int num;
struct nodo * sig;} nodo;
typedef struct nodo * TLista;

typedef struct {
  int num;
  struct nodoD *ant, * sig;} nodoD;
typedef struct nodoD * PnodoD;

typedef struct {
    PnodoD pri, ult;} TListaD;
```

LISTAS CIRCULARES

En esta estructura, el ultimo nodo está vinculado con el primero, permitiendo acceder a cualquier nodo independientemente de dónde se parta.
Ejemplo de dos listas con 5 y 1 nodos respectivamente.



LC almacena la dirección del último nodo, éste en su campo siguiente contiene la dirección del primer nodo de la lista ordenada.

Mantener la dirección del último, en lugar del primero, permite insertar al final de la lista sin tener que recorrerla. Para insertar al comienzo se recurre al último nodo y se modifica su campo siguiente con la dirección del nodo nuevo.

1) Mostrar el contenido de una lista circular

TIPO DE DATOS

```
typedef struct{
     int dato;
     struct nodo * sig;} nodo;
typedef struct nodo * TListaC;
```

Ej 2.- Se tiene una lista circular que almacena palabras (ordenada por este criterio) y cantidad de apariciones de cada una de ellas.

a) Insertar una palabra en la lista.
b) Eliminar una aparición de una palabra

TIPO DE DATOS

```
typedef struct {
    int cant;
    char palabra [15];
    struct nodo * sig;} nodo;
typedef struct nodo * TListaC;
```