

## Ejercicio Operaciones de una LDL

### Descripcion

Desarrolla un programa para administrar una LLS de enteros, sobre la que podamos ejecutar las 5 operaciones básicas descritas en la unidad 3. Luego, desde un módulo "main()" debemos ofrecer al usuario la posibilidad de ejecutar cualquiera de las instrucciones soportadas. Para la operación 2, puedes tomar el mismo código del ciclo de recorrido que propusimos en la Tarea 3.

El código está compuesto por 3 archivos

```
Comando de ejecución gcc mainT4.c lista.c -o t4 ./t4
```

### CODIGOS

ARCHIVO: .h [funciones.h](#)

```
#ifndef FUNCIONES_H_
#define FUNCIONES_H_

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

typedef struct snodo{           //snodo es el nombre de la estructura
    int valor;
    struct snodo *sig;         //El puntero siguiente para recorrer la lista enlazada
}tnodo;                        //tnodo es el tipo de dato para declarar la estructura

typedef tnodo *tpuntero;       //Puntero al tipo de dato tnodo para no utilizar
                                punteros de punteros

//funciones vistas en la clase
void insertarfin(tpuntero *cabeza, int valor);
void insertar(tpuntero *cabeza, int valor);
int recorrer(tpuntero *cabeza);           //recorre la lista y devuelve su
longitud
int buscarDato(tpuntero *cabeza, int valor); //devuelve la posición en que se
encuentra un valor o 1 en otro caso
tnodo *buscaNodo(tpuntero *cabeza, int pos); //devuelve el apuntador al
nodo en una posición en particular
void visita(tnodo *actual);               //recorrido recursivo de la lista
void eliminar(tnodo *actual);

#endif
```

ARCHIVO: main.c [mainT3E1](#)

```
//se compila junto con lista.c
#include "funciones.h"
int main(){
    int e,f;
    tpuntero cabeza,search;          //Indica la cabeza de la lista enlazada, si
    la perdemos no podremos acceder a la lista
    cabeza = NULL;                   //Se inicializa la cabeza como NULL ya que
    no hay ningun nodo cargado en la lista

    do
    {
        printf("\n [1] Ingresar datos");
        printf("\n [2] Imprimir valores");
        printf("\n [3] Numero acual de nodos");
        printf("\n [4] Buscar dato");
        printf("\n [5] Dato en la posicion [i]");
        printf("\n [6] Eliminar dato");
        printf("\n [-1] Salir");
        printf("\n");
        scanf("%d",&f);
        switch (f)
        {
            case 1:
                while(e!=-1){
                    printf("Ingrese elementos, -1 para terminar: ");
                    scanf("%d",&e);
                    if(e!=-1)insertar(&cabeza, e);
                    printf ("Ingresado correctamente");
                    printf ("\n");
                }
                break;
            case 2:
                printf("opcion imprimir en curso");
                recorrer(&cabeza);
                break;
            case 3:
                printf ("\nAl momento hay %d nodos: ",recorrer(&cabeza));
                break;
            case 4:

                printf("Ingresa el dato a buscar");
                int j,dato;
                scanf("%d",&dato);
                j=buscarDato(&cabeza,dato);
                if(j== -1)
                    printf("Dato no encontrado");
                else
                    printf("el dato esta en la posicion %d",j);
                break;
        }
    }
}
```

```

        case 5:
            printf("Ingresa el indice del dato que quieres extraer");
            int i;scanf("%d",&i);
            tnode *ps=buscaNodo(&cabeza,i);
            printf("\nEl dato en la posicion [%d] tiene la direccion de
memoria %p\n",i,ps);
            printf("\nEl dato en la posicion [%d] tiene un valor %d\n",i,ps-
>valor);
            break;
        case 6:
            eliminar(cabeza);
            break;
        default:
            printf("BYE");
            break;
    }

    } while (f!=-1);

return 0;
}

```

## ARCHIVO lista.c [lista](#)

```

#include "funciones.h"

//funciones vistas en la clase

// funcion que inserta al principio
void insertar(tpuntero *cabeza, int valor){
    tnode *nuevo;
    nuevo= (tnodo*)malloc(sizeof(tnode));
    nuevo->valor= valor;
    nuevo->sig= *cabeza;
    nuevo->ant= NULL;
    if (*cabeza==NULL)
        *cabeza=nuevo;
    else
        (*cabeza)->ant=nuevo;
    *cabeza=nuevo;
}

// funcion que inserta antes
void insertarAntes(tpuntero *cabeza, int valor){
    tnode *nuevo, *sig;

```

```
nuevo= (tnodo*)malloc(sizeof(tnodo));
nuevo->valor= valor;
nuevo->ant=*cabeza;
nuevo->sig=(*cabeza)->sig;
sig=nuevo->sig;

if(sig !=NULL)
    sig->ant=nuevo;
nuevo->sig=nuevo;

}

//devuelve la longitud de la lista
int recorrer(tpuntero *cabeza){
    tnodo *aux=*cabeza;
    int longitud =0;

    printf("\nInicio del recorrido\n");
    if(aux != NULL){
        while (aux->sig != NULL)
        {
            printf("%d ",aux->valor);
            aux=aux->sig;
            longitud++;
        }
        longitud++;
        printf("%d",aux->valor);

    }
    return longitud;
}

//devuelve la posicion en que se encuentra un valor o -1 en otro caso
int buscarDato(tpuntero *cabeza,int valor){
    tnodo *aux= *cabeza;
    int posicion=1;

    printf("\nInicio de la busqueda\n");
    while ((aux!=NULL)&&(aux->valor!=valor))
    {
        aux= aux->sig;
        posicion++;
    };
    if (aux!=NULL)
        return posicion;
    else
        return -1;

}
```

```
tnodo *buscaNodo(tpuntero *cabeza,int pos){
    int i=1;
    tnodo *aux=*cabeza;
    while (i<pos && aux!=NULL)
    {
        aux=aux->sig;
        i++;
    }
    return aux;
}

void visita(tnodo *actual){

    if (actual!=NULL)
    {
        printf("%d ",actual->valor);
        if (actual->sig !=NULL)
        {
            visita(actual->sig);
            //printf("%d ",actual->valor);
        }
    }

}

void eliminar(tnodo *actual){
    int val,pos;
    tnodo *objetivo,*previo;

    printf("\n que valor deseas eliminar: ");
    scanf("%d",&val);
    pos=buscarDato(&actual,val);
    objetivo =buscaNodo(&actual,pos);

    if (pos==1)
    {
        actual =objetivo->sig;
    }else{
        previo=buscaNodo(&actual,pos-1);
        previo->sig=objeto->sig;
    }
    free(objetivo);
}
```

## EJECUCION DEL PROGRAMA



