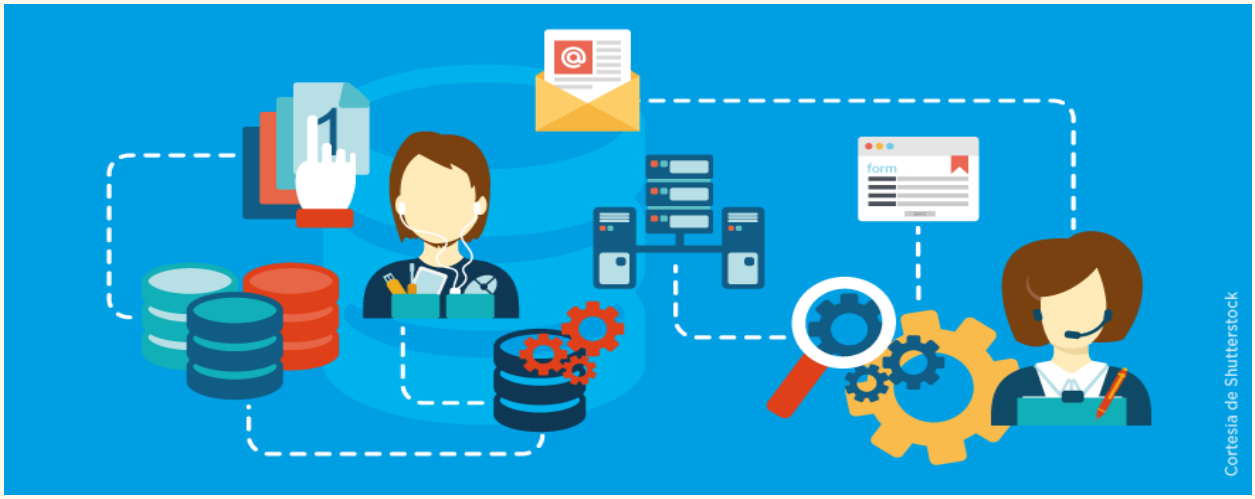
1.a Estructura de Datos.
Dr. Christian Eduardo Millán Hernández

Listas doblemente enlazada
Reproductor de música

—

Leonel Santiago Rosas, 202-A, Ing. En Computación.

https://github.com/utm-ic/pry2-playlist-leonelSantiago22 Repositorio donde se guardó.



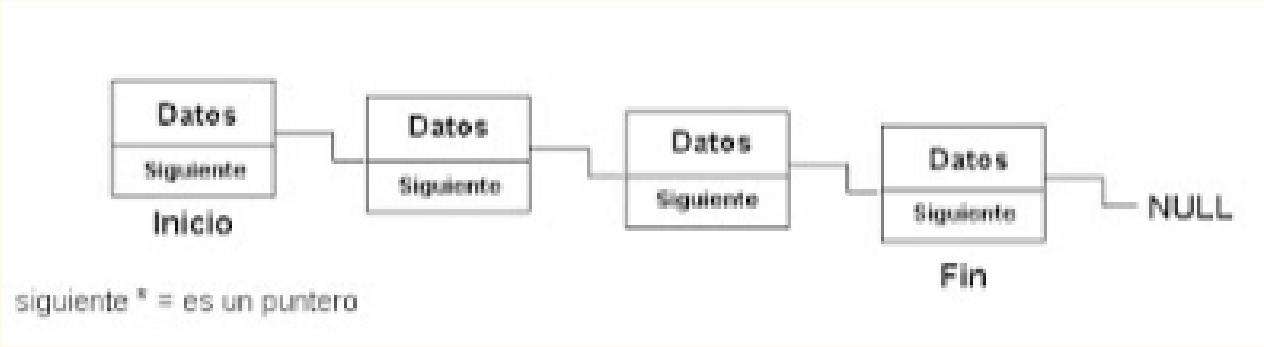
Objetivo de la práctica

Las listas en esta práctica facilitan el movimiento de datos, la modificación de los mismos y además de que siempre y cuando se haga buen eso es una buena manera de administrar datos, todo esto tomando en cuenta que desde luego uno siempre busca una manera eficiente y simple de almacenar datos, las listas en este caso te permiten retrocedes eh ir adelante.

Introducción

Una lista enlazada es una de las estructuras de datos fundamentales, y puede ser usada para implementar otras estructuras de datos. Consiste en una secuencia de nodos, en los que se guardan campos de datos arbitrarios y una o dos referencias, enlaces o punteros al nodo anterior o posterior. El principal beneficio de las listas enlazadas respecto a los vectores convencionales es que el orden de los elementos enlazados puede ser diferente al orden de almacenamiento en la memoria o el disco, permitiendo que el orden de recorrido de la lista sea diferente al de almacenamiento.

Una lista enlazada es un tipo de dato autorreferenciado porque contienen un puntero o enlace a otro dato del mismo tipo. Las listas enlazadas permiten inserciones y eliminación de nodos en cualquier punto de la lista en tiempo constante (suponiendo que dicho punto está previamente identificado o localizado), pero no permiten un acceso aleatorio.



Desarrollo

```
Dlinked_List* create_dlinked_list();

NODE* CreateNode(char *name2, char *artista2, int year2, int id);
```

En estos dos funciones, la primera nos permite creas la lista circular doblemente enlazada, en la cual se van a almacenar los datos, la siguiente nos permite almacenar los datos en un nodo o un espacio designado para esos datos lo cual nos facilita el manejo de datos.

```
void insert(NODE **head, char *name, char *art, int year, int id);
```

Inserta las canciones en el nodo, además de eso nos permite configurar quien es el **head** o la cabecera del programa, para poder utilizar las funciones de **next** y **prev** que en mi caso son para retroceder.

```
int  Destroy (NODE **head, int item);
```

En esta nos permite borrar un dato en específico, que creamos con el, id en especifico, lo cual borrara los datos que contienen esa cancion.

Prueba de ejecución

Insertar y mostrar la canción insertada.

```
1.-Lista de Reproduccion
2.-Salir
:1

1.-Agregar una cancion:
2.-Eliminar una cancion:
3.-Mostrar lista de Reproduccion
4.-Regresar al menu principal
5.-Menu reproducir
:1

1.Ingresa el nombre del artista:        solo

2.Ingresa el nombre de la cancion:      caifanes

Ingresa el anio de lanzamiento: 2010

Se ingreso elemento 1 como nueva cabecera

1.-Agregar una cancion:
2.-Eliminar una cancion:
3.-Mostrar lista de Reproduccion
4.-Regresar al menu principal
5.-Menu reproducir
:3

ID:1    Anio de lazamiento: 2010                Artista: solo   Cancion: caifanes
1.-Agregar una cancion:
2.-Eliminar una cancion:
3.-Mostrar lista de Reproduccion
4.-Regresar al menu principal
5.-Menu reproducir
```

Reproducir eh ir hacia adelante y hacia atras.

```
1.-Agregar una cancion:
2.-Eliminar una cancion:
3.-Mostrar lista de Reproduccion
4.-Regresar al menu principal
5.-Menu reproducir
:5

1.-Reproducir
2.-siguiente
3.-Atras
4.-Regresar al editar
:1

ID: 1   Cancion: messeoneros    Artista: corazon        Anio de lanzamiento: 2010
1.-Reproducir
2.-siguiente
3.-Atras
4.-Regresar al editar
:2

ID: 2   Cancion: messeoneros    Artista: corazon        Anio de lanzamiento : 20011
1.-Reproducir
2.-siguiente
3.-Atras
4.-Regresar al editar
:2

ID: 1   Cancion: messeoneros    Artista: corazon        Anio de lanzamiento : 2010
1.-Reproducir
2.-siguiente
3.-Atras
4.-Regresar al editar
:2
```

Error

```
1.Ingresa el nombre del artista:        corazon

2.Ingresa el nombre de la cancion:      messeoneros

Ingresa el anio de lanzamiento: 20011

1.-Agregar una cancion:
2.-Eliminar una cancion:
3.-Mostrar lista de Reproduccion
4.-Regresar al menu principal
5.-Menu reproducir
:3

ID:1    Anio de lazamiento: 2010                Artista: corazon        Cancion: messeoneros
ID:2    Anio de lazamiento: 20011               Artista: corazon        Cancion: messeoneros
1.-Agregar una cancion:
2.-Eliminar una cancion:
3.-Mostrar lista de Reproduccion
4.-Regresar al menu principal
5.-Menu reproducir
:5
```

Mi único error fue que al momento de insertar una siguiente canción, la canción anterior se modifica y termina afectando a las demás canciones, pero solo afecta en el nombre, como podemos ver el año y él, id no quedan afectados, no sé mi me podría ayudar en el porqué de ese error.

Conclusión

Las listas son muy útiles al momento de utilizar datos que nos permitan almacenar, además de que naturalmente es difícil de programas, mi único fallo fue no poder encontrar mi error que la verdad no sé por qué ocurría eso, pero para mí fue una práctica que me dejo una fácil enseñanza de como manejar los datos a través de las listas.

Código

Declaración de funciones

```
#ifndef SLLIST

#define SLLIST



typedef struct node NODE;

typedef struct dlinked_list Dlinked_List;



struct node

{

    char *nombre;

    char *artista;

    int  year;

    int id;

    NODE *next;
```

```
    NODE *prev;
};


struct dlinked_list
{
    NODE *head;
};


Dlinked_List* create_dlinked_list();
NODE* CreateNode(char *name2, char *artista2, int year2, int id);
int  Destroy (NODE **head, int item);
void insert(NODE **head, char *name, char *art, int year, int id);
void node(NODE *aux);
NODE* back(NODE *dlinked_list);
NODE* nexty(NODE *dlinked_list);
void repro(Dlinked_List *dlinked_list);
void display(Dlinked_List *dlinked_list);
#endif
```

Programacion de las funciones

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

```c
#include "../include/list.h"


Dlinked_List* create_dlinked_list()

{

    Dlinked_List *dlinked_list = (Dlinked_List *)
malloc(sizeof(Dlinked_List));

    dlinked_list->head = NULL;

    return dlinked_list;

}

NODE* CreateNode(char *name, char *artista, int year2, int id)

{

    NODE *new_node = (NODE *) malloc(sizeof(NODE));

    new_node->id = id;

    new_node->year = year2;

    new_node->nombre = name;

    new_node->artista = artista;

    new_node->next = new_node;

    new_node->prev = new_node;

    return new_node;


}


int Destroy(NODE **head, int item)
```

```
{
NODE *aux;
    int data = INT_MIN;
    int id;
    char *song;
    char *artist;
    if (*head==NULL)
    {
        data = INT_MIN;
    }else
    {
        if ((*head)->id == item)
        {



        if (((*head)->next == *head)&& ((*head)->prev== *head))
        {
            data =(*head)->year;
            id = (*head)->id;
            song = (*head)->nombre;
            artist = (*head)->artista;
            free(*head);
            *head = NULL;
```

```
        }else{
            aux = *head;
            while ((aux->next != *head) && (aux->prev->id != item))
            {
                aux = aux->next;
            }
                aux->next = (*head)->next;
                (*head)->next->prev =aux;
                NODE *temp = (*head);
                *head =(*head)->next;
                data = temp->year;
                song = temp->nombre;
                artist = temp->artista;
                id = temp->id;
                free(temp);
        }
        }else
        {
            NODE *prev = *head;
            while((prev->next != *head) && (prev->next->id != item))
            {
                prev = prev->next;
            }
```

```
            if (prev->next->id == item)
            {
                NODE *ax = prev->next;
                prev->next = ax->next;
                ax->next->prev = prev;
                data = ax->year;
                id = ax->id;
                song = ax->nombre;
                artist = ax->artista;
                free(ax);
            }else{
                data = INT_MAX;
            }
        }

    }
    printf("\nSe elimino %s de %s lazanda en %d con id %d ", song, artist,
data, id);

    return id;
}

void insert(NODE **head, char *name, char *art, int year, int id)
```

```
{
    NODE *new_node = CreateNode(name, art, year, id);
    NODE *aux = *head;
    if (*head == NULL)
    {
        *head = new_node;
        printf("\nSe ingreso elemento %d como nueva cabecera\n", id);
    }else{
        if (aux->next->year >= new_node->year){
        while((aux->next != *head) )
        {
            aux = aux->next;
        }
            new_node->next = *head;
            new_node->prev = aux;
            aux->next = new_node;
            (*head)->prev = new_node;
            *head = new_node;
        }else{
            while((aux->next != *head) && (aux->next->year <
new_node->year))
            {
                aux = aux->next;
```

```
            }

            aux->next->prev = new_node;

            new_node->next = aux->next;

            new_node->prev = aux;

            aux->next = new_node;

        }

            //aux = aux->next;

        }

 }


void display(Dlinked_List *dlinked_list)

{

    NODE *temporal;

    if (dlinked_list->head !=NULL)

    {

        temporal = dlinked_list->head;

        do

        {

            printf("\nID:%d", temporal->id);

            printf("\tAnio de lazamiento: %d\t", temporal->year);

            printf("\tArtista: %s", temporal->artista);

            printf("\tCancion: %s", temporal->nombre);
```

```
            temporal = temporal->next;
        } while ((temporal != 
dlinked_list->head)&&(temporal->year!=0)&&(temporal != NULL));
    }
}

NODE* back(NODE *dlinked_list)
{
    NODE *temporal;
    temporal = dlinked_list;
    temporal = temporal->prev;
    if (dlinked_list != NULL)
    {

        printf("\nID: %d", temporal->id);
        printf("\tCancion: %s", temporal->nombre);
        printf("\tArtista: %s", temporal->artista);
        printf("\tAnio de lanzamiento : %d", temporal->year);

    }else{
        printf("\nList is empty\n");
    }
    return temporal;
```

```
}
NODE* nexty(NODE *dlinked_list)
{
    NODE *temporal;
    if (dlinked_list != NULL)
    {
        temporal = dlinked_list;
        temporal = temporal->next;
        printf("\nID: %d", temporal->id);
        printf("\tCancion: %s", temporal->nombre);
        printf("\tArtista: %s", temporal->artista);
        printf("\tAnio de lanzamiento : %d", temporal->year);

    }else{
        printf("\nList is empty\n");
    }
    return temporal;
}
void repro(Dlinked_List *dlinked_list)
{
    NODE *temporal;

    if (dlinked_list->head != NULL)
```

```
    {
        temporal = dlinked_list->head;
        printf("\nID: %d", temporal->id);
        printf("\tCancion: %s", temporal->nombre);
        printf("\tArtista: %s", temporal->artista);
        printf("\tAnio de lanzamiento: %d", temporal->year);

    }else{
        printf("\nList is empty\n");
    }

}
```

Menu

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <limits.h>
#include "../include/list.h"
void lista(Dlinked_List *dlinked_list);
void reproducir(Dlinked_List *dlinked_list);
int menu();
//int cont  = 1;
int main()
```

```
{
    Dlinked_List *dlinked_list = create_dlinked_list();
    //Dlinked_List *dlinked_list = create_dlinked_list();
    int op;
    while(1)
    {
        op = menu();
       switch (op)
       {
       case 1: lista(dlinked_list); break;
       case 2: exit(-1);
       default:
           break;
       }
    }

}

int menu(){
     int op;
    printf("\n1.-Lista de Reproduccion");
    printf("\n2.-Salir\n:");
    scanf("%d", &op);
```

```
    return op;
}
int id=1;
void lista(Dlinked_List *dlinked_list )
{
    int op;

    int del_node;
    while(1){

    char *nombre=NULL, *artista=NULL;
    char cancion[40], banda[40];
    int year;
    printf("\n1.-Agregar una cancion: ");
    printf("\n2.-Eliminar una cancion: ");
    printf("\n3.-Mostrar lista de Reproduccion");
    printf("\n4.-Regresar al menu principal ");
    printf("\n5.-Menu reproducir\n:");
    scanf("%d",&op);

        switch (op)
        {
        case 1: /*printf("\nIngresa el numero de la cancion: ");
```

```
                fflush(stdin);
                scanf("%d",&id);*/
                printf("\n1.Ingresa el nombre del artista:\t");
                fflush(stdin);
                scanf("%s", banda);
                printf("\n2.Ingresa el nombre de la cancion:\t");
                fflush(stdin);
                scanf("%s", cancion);
                printf("\nIngresa el anio de lanzamiento:\t");
                fflush(stdin);
                scanf("%d", &year);
                nombre = cancion;
                artista = banda;
                insert(&dlinked_list->head, nombre, artista, year, id);
                id++;
                break;
        case 2: display(dlinked_list);
                int item;
                printf("\nIngresa el Numero de la cancion que deseas
eliminar: ");
                scanf("%d", &item);
                del_node = Destroy(&dlinked_list->head, item);
                if (del_node == 1)
```

```
                {
                    printf("\nList is empty");
                }else{
                    if (del_node == INT_MAX)
                    {
                        printf("\nElemento no encontrado");
                    }else{
                        printf("\nElemento Eliminado correctamente");
                    }
                }
                id--;
                break;

        case 3: display(dlinked_list);break;
        case 4: main(); break;
        case 5: reproducir(dlinked_list);
        default:
            break;
        }
    }


}
```

```
void reproducir(Dlinked_List *dlinked_list)
{
    NODE * temp;
    int op;
    do
    {
        temp = dlinked_list->head;
        printf("\n1.-Reproducir");
        printf("\n2.-siguiente");
        printf("\n3.-Atras");
        printf("\n4.-Regresar al editar\n:");
        scanf("%d", &op);
        switch (op)
        {
        case 1 : repro(dlinked_list); break;
        case 2 : temp = back(temp); break;
        case 3 : temp = nexty(temp); break;
        case 4 : lista(dlinked_list); break;
        default: printf("\nOpcion invalida"); break;
        }
       dlinked_list->head = temp;
    } while (op<=4);
```

```
}
```