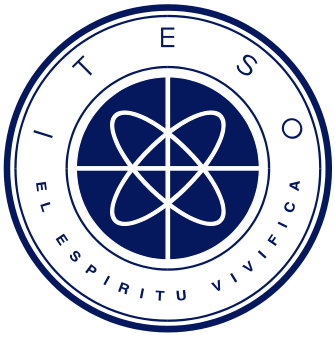
INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Departamento de Electrónica, Sistemas e Informática

INGENIERÍA EN SISTEMAS COMPUTACIONALES



PROGRAMACIÓN CON MEMORIA DINÁMICA

TAREA 2. MEMORIA DINÁMICA Y ARCHIVOS

Autor: Gallegos Velica, José Isaac

Presentación: 10 pts.
Funcionalidad: 60 pts.
Pruebas: 20 pts.

12 de junio de 2018. Tlaquepaque, Jalisco,

- Falta describir las pruebas (escenario, y resultados de la experimentación).

Objetivo de la actividad

El objetivo de la tarea es que el alumno aplique los conocimientos y habilidades adquiridos en el tema de manejo de memoria dinámica y archivos utilizando el lenguaje ANSI C.

Descripción del problema

Ahora tienes los conocimientos para enfrentarte a un nuevo proyecto llamado **MyDB**. En este proyecto vas a recrear una parte de un sistema de transacciones bancarias. Para esto vas a requerir del uso de:

- Estructuras
- Funciones y paso de parámetros
- Apuntadores
- Memoria Dinámica
- Archivos binarios

El sistema **MyDB** al ser ejecutado deberá mostrar al usuario una interfaz con el siguiente menú principal:

<< Sistema MyDB >>

1. Clientes
2. Cuentas
3. Transacciones
4. Salir

El sistema **MyDB** debe realizar automáticamente, las siguientes operaciones:

A) Si el sistema **MyDB** se ejecutó por primera vez, este deberá crear tres archivos binarios: **clientes.dat**, **cuentas.dat** y **transacciones.dat**. Para esto el sistema debe solicitar al usuario indicar la **ruta de acceso** (por ejemplo, c:\\carpeta\\) en donde se desea crear los archivos (esta información deberá ser almacenada en un archivo de texto llamado **mydb.sys**).

Clientes

La opción **Clientes** debe mostrar un submenú con las siguientes opciones:

- **Nuevo** cliente — Registra los datos de un nuevo cliente del banco
- **Buscar** cliente — Permite consultar la información de un usuario a partir de su id_cliente.
- **Eliminar** cliente — Si existe, elimina un usuario deseado del sistema. Esto implica que deben Borrarse las cuentas registradas a nombre del usuario

(utilice id_usuario para buscar).

- **Imprimir** clientes — Imprime la información de todos los clientes registrados en el sistema.

La información que el sistema requiere almacenar sobre cada cliente es la siguiente:

- Id_usuario (es un número entero que se genera de manera consecutiva, clave única)
- Nombre
- Apellido materno
- Apellido paterno
- Fecha de nacimiento (tipo de dato estructurado: dd/mm/aaaa)

Para gestionar la información de los clientes, defina un tipo de dato estructurado llamado **Usuario**, utilice instancias de Usuario para capturar la información desde el teclado y posteriormente guardarlo en el archivo usuario.dat.

Un ejemplo del contenido que se estará almacenando en el archivo **usuario.dat** es el siguiente:

id_usuario	nombre	apellido_paterno	apellido_materno	fecha_nacimiento
1	Ricardo	Perez	Perez	{3,10, 2010}
2	Luis	Rodriguez	Mejía	{2,7, 2005}
3	Gabriela	Martínez	Aguilar	{7,11,2015}

Importante: considere que no pueden existir datos **id_usuario** repetidos y que es un valor autonúmerico. Adicionalmente, recuerde que al inicio el archivo no tendrá datos.

Cuentas

La opción **Cuentas** debe mostrar un submenú con las siguientes opciones:

- **Nueva** cuenta — Registra una cuenta nueva a nombre de un usuario, utilice **id_cliente** para relacionar el usuario y la cuenta. Antes de crear la nueva cuenta se debe verificar que el usuario exista en el sistema. Adicionalmente, se debe indicar el saldo con el que se abre la cuenta. Por ejemplo; $1000.

- **Buscar** cuenta — Permite consultar en pantalla la información de una cuenta en el sistema a partir de su **id_cuenta**. En pantalla debe mostrarse: **id_cuenta**, **nombre de cliente**, **saldo de la cuenta**.

- **Eliminar** cuenta — Si existe, elimina la cuenta deseada en el sistema.

- **Imprimir** cuentas — Imprime la información de todas las cuentas registradas en el sistema. En pantalla debe mostrarse un listado con la siguiente información de las cuentas: **id_cuenta**, **nombre de cliente**, **saldo de la cuenta**.

La información que el sistema requiere almacenar sobre cada cuenta es la siguiente:

- id_cuenta (es un número entero que se genera de manera consecutiva, clave única)
- id_usuario (indica a quien pertenece la cuenta)
- Saldo
- Fecha de apertura (tipo de dato estructurado: dd/mm/aaaa)

Para gestionar la información de las cuentas, defina un tipo de dato estructurado llamado **Cuenta**, utilice instancias de **Cuenta** para capturar la información desde el teclado y posteriormente guardarlo en el archivo **cuenta.dat**.

Un ejemplo del contenido que se estará almacenando en el archivo **cuenta.dat** es el siguiente:

id_cuenta	Id_usuario	Saldo	fecha_apertura
1	1	Perez	{12,6, 2018}
2	2	Rodriguez	{2,7, 2018}
3	1	Martínez	{7,3,2018}

Importante: considere que no pueden existir valores de **id_cuenta** repetidos y que es un valor autonúmerico. Adicionalmente, observe que un usuario puede tener más de una cuenta.

Transacciones

La opción **Transacciones** debe mostrar un submenú con las siguientes opciones:

- **Depósito** Permite incrementar el saldo de la cuenta, para esto el sistema requiere: **id_cuenta**, **monto a depositar** (valide que la cuenta exista).

- **Retiro** Permite a un cliente disponer del dinero que tiene una cuenta bancaria. Para esto el sistema requiere: **id_cuenta**, **monto a retirar** (valide que la cuenta existe y que tiene fondos suficientes).

- **Transferencia** Permite a un cliente transferir dinero de una cuenta origen a una cuenta destino. Para esto el sistema requiere: **id_cuenta origen**, **id_cuenta destino**, **monto a transferir** (valide que existan ambas cuentas y que la cuenta origen tiene fondos suficientes).

La información que el sistema requiere almacenar sobre cada transacción es la siguiente:

- id_transacción (es un número entero que se genera de manera consecutiva, no se puede repetir)
- Tipo de operación (depósito, retiro, transferencia)
- Cuenta origen
- Cuenta destino (se utiliza para las operaciones de transferencia, en otro caso, NULL)

- Fecha de la transacción
- Monto de la transacción

Para gestionar la información de las trasferencias, defina un tipo de dato estructurado llamado **Transferencia**, utilice instancias de Transferencia para capturar la información desde el teclado y posteriormente guardarlo en el archivo transferencia.dat.

Un ejemplo del contenido que se estará almacenando en el archivo **transferencia.dat** es el siguiente:

id_transaccion	tipo_transaccion	Id_cuenta_origen	Id_cuenta_destino	fecha_transaccion	monto_transaccion
1	Retiro	1	Null	{12,6, 2018}	$100
2	Deposito	2	Null	{12,6, 2018}	$5000
3	Transferencia	2	1	{12,6,2018}	$1500

Importante: considere que no pueden existir datos **id_transaccion** repetidos y que es un valor autonúmerico. Adicionalmente, recuerde que al inicio el archivo no tendrá datos y que los saldos de las cuentas deberán afectarse por las transacciones realizadas.

SOLUCIÓN DEL ALUMNO, PRUEBAS Y CONCLUSIONES

Código fuente

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

typedef struct{
        int dia, mes, anio;
}Fecha;
typedef struct{
        int id;
        char nombre[20];
        char ap_paterno[20];
        char ap_materno[20];
        Fecha nacimiento;
}Cliente;
typedef struct{
        int id_cuenta;
        int id_usuario;
        int saldo;
        Fecha apertura;
}Cuenta;
typedef struct{
        int id_transaccion;
        int tipo;
        int id_origen;
        int id_destino;
        Fecha fecha;
        int monto;
}Transaccion;


int extraerdireccion(char** clientes,char** cuentas,char** transacciones);
int menu_cliente(char *ubicacion, char *cuentas);
Cliente new_cliente(int id);
Fecha get_fecha();
void imprimir_cliente(FILE* archivo);
void lista_clientes(FILE* archivo);
FILE* buscar_cliente(FILE* archivo, int id);
int menu_cuenta(char *ubicacion, char *clientes);
Cuenta new_cuenta(int id, FILE *clientes);
void imprimir_cuenta(FILE* archivo);
void lista_cuentas(FILE* archivo);
FILE* buscar_cuenta(FILE* archivo, int id);
int eliminar_cuenta(int ,FILE* archivo,char* ubicacion);
int eliminar_cliente(int ,FILE* archivo,char* ubicacion);
int menu_transacciones(char* ubicacion, char* cuentas);
Transaccion new_transaccion(int id, FILE* cuentas);
FILE* buscar_transaccion(FILE* archivo, int id);
void imprimir_transaccion(FILE* archivo);
void lista_transacciones(FILE* archivo);
int eliminar_cuentaxcliente(int id_eliminado, char * ubicacion);

int main(){
        setvbuf(stderr, NULL, _IONBF, 0);
        setvbuf(stdout, NULL, _IONBF, 0);
        char *clientes, *cuentas, *transacciones;
        int menu;
        extraerdireccion(&clientes,&cuentas,&transacciones);

        //MENU PRINCIPAL
        do{
        printf("\n<<Sistema MyDB>>\n");
```

```
        printf("[1]\tClientes\n[2]\tCuentas\n[3]\tTransacciones\n[4]\tSalir\n");
        scanf("%d", &menu);
        switch(menu){
                case 1:
                        menu_cliente(clientes, cuentas);
                        break;
                case 2:
                        menu_cuenta(cuentas,clientes);
                        break;
                case 3:
                        menu_transacciones(transacciones,cuentas);
                        break;
        }
        }while(menu!=4);
        printf("FIN");
        return 0;
}
Fecha get_fecha(){
        Fecha new;
        printf("(dd/mm/aaaa): ");
        scanf("%d/%d/%d", &new.dia, &new.mes, &new.anio);
        return new;
}
int extraerdireccion(char** clientes,char** cuentas,char** transacciones){
        FILE*  direccion;
        char *ubicacion;
        long len;
        direccion=fopen("mydb.sys", "r");                       //Apertura en modo lectura
        if(direccion==NULL){
                direccion=fopen("mydb.sys", "w+");              //Apertura de archivo escribir/leer
                if(direccion==NULL)
                        return -1;
                char c;
                printf("Introdusca la dirección de los archivos:\n");
                while((c=getchar())!='\n')                              //Entrada de direccion
                        fputc(c,direccion);
        }
        fseek(direccion,0,SEEK_END);
        len=ftell(direccion);
        ubicacion=(char*)malloc(len);
        *clientes=(char*)malloc((len+15));
        *cuentas=(char*)malloc((len+14));
        *transacciones=(char*)malloc((len+20));
        rewind(direccion);
        fscanf(direccion,"%[^EOF]",ubicacion);
        strcpy(*clientes,ubicacion);
        strcpy(*cuentas,ubicacion);
        strcpy(*transacciones,ubicacion);
        strcpy(*clientes+len,"\\\\clientes.dat");
        strcpy(*cuentas+len,"\\\\cuentas.dat");
        strcpy(*transacciones+len,"\\\\transaccioness.dat");
        free(ubicacion);
        fclose(direccion);
        return 0;
}

int menu_cliente(char *ubicacion, char*cuentas){
        FILE* f_clientes, *f_cuentas;
        int operacion,id;
        printf("<<CLIENTES>>\n");
        printf("[1]\tNuevo\n[2]\tBuscar\n[3]\tEliminar\n[4]\tImprimir\n");
        scanf("%d",&operacion);
        switch(operacion){
                case 1:
                        f_clientes=fopen(ubicacion,"rb+");
                        if(f_clientes==NULL){
                                f_clientes=fopen(ubicacion,"wb");
                                if(f_clientes==NULL)
                                        return -1;
```

```
                        id=1;
                }
                else{
                        fread(&id,sizeof(int),1,f_clientes);
                        rewind(f_clientes);
                        id++;
                }
                fwrite(&id,sizeof(int),1,f_clientes);
                Cliente new=new_cliente(id);
                fseek(f_clientes,0,SEEK_END);
                fwrite(&new,sizeof(Cliente),1,f_clientes);
                break;
        case 2:
                f_clientes=fopen(ubicacion,"rb");
                if(f_clientes==NULL)
                        return -1;
                printf("Id de usuario: ");scanf("%d",&id);
                printf("%-13s%-23s%-23s%-23sFECHA_NACIMIENTO\n","ID_USUARIO","NOMBRE",
"APELLIDO_PATERNO","APELLIDO_MATERNO");
                imprimir_cliente(buscar_cliente(f_clientes, id));
                break;
        case 3:
                f_clientes=fopen(ubicacion,"rb");
                if(f_clientes==NULL)
                        return -1;
                FILE* Cliente_eliminar;
                printf("Id de usuario: ");scanf("%d",&id);
                Cliente_eliminar=buscar_cliente(f_clientes,id);
                if(Cliente_eliminar!=NULL){
                        eliminar_cliente(id,f_clientes,ubicacion);
                        eliminar_cuentaxcliente(id, cuentas);
                }
                break;
        case 4:
                f_clientes=fopen(ubicacion,"rb");
                if(f_clientes==NULL)
                        return -1;
                lista_clientes(f_clientes);
                break;
        }
        fclose(f_clientes);
        return 0;
}
Cliente new_cliente(int id){
        Cliente new;
        new.id=id;
        printf("Nombre: ");scanf("%s", new.nombre);
        printf("Apellido Paterno: ");scanf("%s", new.ap_paterno);
        printf("Apellido Materno: ");scanf("%s", new.ap_materno);
        printf("Fecha de nacimiento");
        new.nacimiento=get_fecha();
        return new;
}
void imprimir_cliente(FILE* archivo){
        if(archivo!=NULL){
                Cliente cliente;
                fread(&cliente, sizeof(Cliente), 1,archivo);
                printf("%-13d%-23s%-23s%-
23s{%02d,%02d,%04d}\n",cliente.id,cliente.nombre,cliente.ap_paterno,cliente.ap_materno,

        cliente.nacimiento.dia,cliente.nacimiento.mes,cliente.nacimiento.anio);
        }
}
void lista_clientes(FILE* archivo){
        char c;
        fseek(archivo,sizeof(int),SEEK_SET);
        printf("%-13s%-23s%-23s%-23sFECHA_NACIMIENTO\n","ID_USUARIO","NOMBRE",
"APELLIDO_PATERNO","APELLIDO_MATERNO");
        while((c=fgetc(archivo))!=EOF){
```

```
                fseek(archivo,-sizeof(char),SEEK_CUR);
                imprimir_cliente(archivo);
        }
}
FILE* buscar_cliente(FILE* archivo, int id){
        int temp;
        char c;
        fseek(archivo,sizeof(int),SEEK_SET);
        while((c=fgetc(archivo))!=EOF){
                fseek(archivo,-1,SEEK_CUR);
                fread(&temp,sizeof(int),1,archivo);
                if(temp==id){
                        fseek(archivo,-(sizeof(int)),SEEK_CUR);
                        return archivo;
                }
                fseek(archivo,sizeof(Cliente)-sizeof(int),SEEK_CUR);
        }
        printf("Cliente no encontrado\n");
        return NULL;
}
int eliminar_cliente(int id_eliminado, FILE *archivo, char * ubicacion){
        int posicion, n_clientes,id_actual;
        Cliente *clientes;
        posicion=(ftell(archivo)-4)/sizeof(Cliente);
        fseek(archivo,0,SEEK_END);
        n_clientes=(ftell(archivo)-4)/sizeof(Cliente);
        rewind(archivo);
        fread(&id_actual,sizeof(int),1,archivo);
        clientes=(Cliente*)malloc(n_clientes*sizeof(Cliente));
        fread(clientes,sizeof(Cliente),n_clientes,archivo);
        for(int i=posicion;i<n_clientes-1;i++)
                clientes[i]=clientes[i+1];
        fclose(archivo);

        archivo=fopen(ubicacion,"wb");
        if(archivo==NULL)
                return -1;
        fwrite(&id_actual,sizeof(int),1,archivo);
        fwrite(clientes,sizeof(Cliente),n_clientes-1,archivo);
        free(clientes);
        return 0;
}

int menu_cuenta(char *ubicacion, char *clientes){
        FILE *f_cuentas, *f_clientes;
        int operacion,id;
        printf("<<CUENTAS>>\n");
        printf("[1]\tNueva\n[2]\tBuscar\n[3]\tEliminar\n[4]\tImprimir\n");
        scanf("%d",&operacion);
        switch(operacion){
                case 1:
                        f_cuentas=fopen(ubicacion,"rb+");
                        if(f_cuentas==NULL){
                                f_cuentas=fopen(ubicacion,"wb");
                                if(f_cuentas==NULL)
                                        return -1;
                                id=1;
                        }
                        else{
                                fread(&id,sizeof(int),1,f_cuentas);
                                rewind(f_cuentas);
                                id++;
                        }
                        f_clientes=fopen(clientes,"rb");
                        if(f_clientes==NULL)
                                return -1;
                        Cuenta new=new_cuenta(id,f_clientes);
                        if(new.id_cuenta!=0){
                                fwrite(&id,sizeof(int),1,f_cuentas);
```

```
                                fseek(f_cuentas,0,SEEK_END);
                                fwrite(&new,sizeof(Cuenta),1,f_cuentas);
                        }
                        fclose(f_clientes);
                        break;
                case 2:
                        f_cuentas=fopen(ubicacion,"rb");
                        if(f_cuentas==NULL)
                                return -1;
                        printf("Id de cuenta: ");scanf("%d",&id);
                        printf("%-12s%-13s%-13sFECHA_APERTURA\n","ID_CUENTA","ID_USUARIO","SALDO");
                        imprimir_cuenta(buscar_cuenta(f_cuentas, id));
                        break;
                case 3:
                        f_cuentas=fopen(ubicacion,"rb");
                        if(f_cuentas==NULL)
                                return -1;
                        FILE* Cuenta_eliminar;
                        printf("Id de cuenta: ");scanf("%d",&id);
                        Cuenta_eliminar=buscar_cuenta(f_cuentas,id);
                        if(Cuenta_eliminar!=NULL)
                                eliminar_cuenta(id,Cuenta_eliminar,ubicacion);
                        break;
                case 4:
                        f_cuentas=fopen(ubicacion,"rb");
                        if(f_cuentas==NULL)
                                return -1;
                        lista_cuentas(f_cuentas);
                        break;
        }
        fclose(f_cuentas);
        return 0;
}
Cuenta new_cuenta(int id, FILE* cuentas){
        Cuenta new;
        new.id_cuenta=0;
        printf("Id del cliente: ");scanf("%d", &new.id_usuario);
        if(buscar_cliente(cuentas,new.id_usuario)==NULL)
                return new;
        new.id_cuenta=id;
        printf("Saldo de apertura: ");scanf("%d", &new.saldo);
        printf("Fecha de apertura");
        new.apertura=get_fecha();
        return new;
}
void imprimir_cuenta(FILE* archivo){
        if(archivo!=NULL){
                Cuenta cuenta;
                fread(&cuenta, sizeof(Cuenta), 1,archivo);
                printf("%-12d%-13d%-
13d{%02d,%02d,%04d}\n",cuenta.id_cuenta,cuenta.id_usuario,cuenta.saldo,
                                cuenta.apertura.dia,cuenta.apertura.mes, cuenta.apertura.anio);
        }
}
void lista_cuentas(FILE* archivo){
        char c;
        fseek(archivo,sizeof(int),SEEK_SET);
        printf("%-12s%-13s%-13sFECHA_APERTURA\n","ID_CUENTA","ID_USUARIO","SALDO");
        while((c=fgetc(archivo))!=EOF){
                fseek(archivo,-sizeof(char),SEEK_CUR);
                imprimir_cuenta(archivo);
        }
}
FILE* buscar_cuenta(FILE* archivo, int id){
        int temp;
        char c;
        fseek(archivo,sizeof(int),SEEK_SET);
        while((c=fgetc(archivo))!=EOF){
                fseek(archivo,-1,SEEK_CUR);
```

```
                fread(&temp,sizeof(int),1,archivo);
                if(temp==id){
                        fseek(archivo,-(sizeof(int)),SEEK_CUR);
                        return archivo;
                }
                fseek(archivo,sizeof(Cuenta)-sizeof(int),SEEK_CUR);
        }
        printf("Cuenta no encontrado\n");
        return NULL;
}
int eliminar_cuenta(int id_eliminado, FILE *archivo, char * ubicacion){
        int posicion, n_cuentas,id_actual;
        Cuenta *cuentas;
        posicion=(ftell(archivo)-4)/sizeof(Cuenta);
        fseek(archivo,0,SEEK_END);
        n_cuentas=(ftell(archivo)-4)/sizeof(Cuenta);
        rewind(archivo);
        fread(&id_actual,sizeof(int),1,archivo);
        cuentas=(Cuenta*)malloc(n_cuentas*sizeof(Cuenta));
        fread(cuentas,sizeof(Cuenta),n_cuentas,archivo);
        for(int i=posicion;i<n_cuentas-1;i++)
                cuentas[i]=cuentas[i+1];
        fclose(archivo);

        archivo=fopen(ubicacion,"wb");
        if(archivo==NULL){
                printf("Error al abrir archivo\n");
                return -1;
        }
        fwrite(&id_actual,sizeof(int),1,archivo);
        fwrite(cuentas,sizeof(Cuenta),n_cuentas-1,archivo);
        free(cuentas);
        return 0;
}
int eliminar_cuentaxcliente(int id_eliminado, char * ubicacion){
        FILE *archivo;
        Cuenta* cuentas;
        int id_actual, n_cuentas;
        archivo=fopen(ubicacion,"rb");
        if(archivo==NULL)
                return -1;
        fseek(archivo,0,SEEK_END);
        n_cuentas=(ftell(archivo)-4)/sizeof(Cuenta);
        rewind(archivo);
        fread(&id_actual,sizeof(int),1,archivo);
        cuentas=(Cuenta*)malloc(n_cuentas*sizeof(Cuenta));
        fread(cuentas,sizeof(Cuenta),n_cuentas,archivo);
        fclose(archivo);
        archivo=fopen(ubicacion,"wb");
        if(archivo==NULL){
                printf("Error al abrir archivo\n");
                return -1;
        }
        fwrite(&id_actual,sizeof(int),1,archivo);
        for(int i=0;i<n_cuentas;i++){
                if(cuentas[i].id_usuario!=id_eliminado)
                        fwrite(cuentas+i,sizeof(Cuenta),1,archivo);
        }
        fclose(archivo);
        free(cuentas);
        return 0;
}


int menu_transacciones(char* ubicacion, char* cuentas){
        FILE *f_trans,*f_cuentas;
        int operacion, id;

        printf("<<TRANSACCIONES>>\n");
```

```
        printf("[1]\tNueva\n[2]\tBuscar\n[3]\tImprimir\n");
        scanf("%d",&operacion);
        switch(operacion){
        case 1:
                f_trans=fopen(ubicacion,"rb+");
                if(f_trans==NULL){
                        f_trans=fopen(ubicacion,"wb");
                        if(f_trans==NULL)
                                return -1;
                        id=1;
                }
                else{
                        fread(&id,sizeof(int),1,f_trans);
                        rewind(f_trans);
                        id++;
                }
                f_cuentas=fopen(cuentas,"rb+");
                if(f_cuentas==NULL)
                        return -1;
                Transaccion new;
                new=new_transaccion(id,f_cuentas);
                if(new.id_transaccion==0)
                        break;
                fwrite(&id,sizeof(int),1,f_trans);
                fseek(f_trans,0,SEEK_END);
                fwrite(&new,sizeof(Transaccion),1,f_trans);
                fclose(f_cuentas);
                break;
        case 2:
                f_trans=fopen(ubicacion,"rb");
                if(f_trans==NULL)
                        return -1;
                printf("Id de transaccion: ");scanf("%d",&id);
                printf("ID_TRANSACCION   TIPO_TRANSACCION   ID_CUENTA_ORIGEN   ID_CUENTA_DESTINO
MONTO_TRANSACCION   FECHA_TRANSACCION\n");
                imprimir_transaccion(buscar_transaccion(f_trans, id));
                break;
        case 3:
                f_trans=fopen(ubicacion,"rb");
                if(f_trans==NULL)
                        return -1;
                lista_transacciones(f_trans);
                break;
        }
        fclose(f_trans);
        return 0;
}
Transaccion new_transaccion(int id, FILE* cuentas){
        Transaccion new;
        int saldo_origen, saldo_destino;
        new.id_transaccion=0;
        printf("Tipo de transaccion:\n\t[1] Retiro\n\t[2] Deposito\n\t[3] Transferencia\n");
        scanf("%d",&new.tipo);
        switch(new.tipo){
        case 1:
                printf("Id cuenta origen:");scanf("%d",&new.id_origen);
                if(buscar_cuenta(cuentas,new.id_origen)==NULL)
                        return new;
                fseek(cuentas,8,SEEK_CUR);
                fread(&saldo_origen,sizeof(int),1,cuentas);
                new.id_destino=0;
                printf("Monto:");scanf("%d",&new.monto);
                if(new.monto>saldo_origen){
                        printf("Saldo insufisiente\n");
                        return new;
                }
                saldo_origen-=new.monto;
                printf("Fecha");
                new.fecha=get_fecha();
```

```
                buscar_cuenta(cuentas,new.id_origen);
                fseek(cuentas,8,SEEK_CUR);
                fwrite(&saldo_origen,sizeof(int),1,cuentas);
                break;
        case 2:
                printf("Id cuenta destino:");scanf("%d",&new.id_destino);
                if(buscar_cuenta(cuentas,new.id_destino)==NULL)
                        return new;
                fseek(cuentas,8,SEEK_CUR);
                fread(&saldo_destino,sizeof(int),1,cuentas);
                new.id_origen=0;
                printf("Monto:");scanf("%d",&new.monto);
                if(new.monto>saldo_origen){
                        printf("Saldo insufisiente\n");
                        return new;
                }
                saldo_destino+=new.monto;
                printf("Fecha");
                new.fecha=get_fecha();
                buscar_cuenta(cuentas,new.id_destino);
                fseek(cuentas,8,SEEK_CUR);
                fwrite(&saldo_destino,sizeof(int),1,cuentas);
                break;
        case 3:
                printf("Id cuenta origen:");scanf("%d",&new.id_origen);
                if(buscar_cuenta(cuentas,new.id_origen)==NULL)
                        return new;
                fseek(cuentas,8,SEEK_CUR);
                fread(&saldo_origen,sizeof(int),1,cuentas);
                printf("Id cuenta destino:");scanf("%d",&new.id_destino);
                if(buscar_cuenta(cuentas,new.id_destino)==NULL)
                        return new;
                fseek(cuentas,8,SEEK_CUR);
                fread(&saldo_destino,sizeof(int),1,cuentas);
                printf("Monto:");scanf("%d",&new.monto);
                if(new.monto>saldo_origen){
                        printf("Saldo insufisiente\n");
                        return new;
                }
                saldo_origen-=new.monto;
                saldo_destino+=new.monto;
                printf("Fecha");
                new.fecha=get_fecha();
                buscar_cuenta(cuentas,new.id_destino);
                fseek(cuentas,8,SEEK_CUR);
                fwrite(&saldo_destino,sizeof(int),1,cuentas);
                buscar_cuenta(cuentas,new.id_origen);
                fseek(cuentas,8,SEEK_CUR);
                fwrite(&saldo_origen,sizeof(int),1,cuentas);
                break;
        }
        new.id_transaccion=id;
        return new;
}
FILE* buscar_transaccion(FILE* archivo, int id){
        int temp;
        char c;
        fseek(archivo,sizeof(int),SEEK_SET);
        while((c=fgetc(archivo)!=EOF)){
                fseek(archivo,-1,SEEK_CUR);
                fread(&temp,sizeof(int),1,archivo);
                if(temp==id){
                        fseek(archivo,-(sizeof(int)),SEEK_CUR);
                        return archivo;
                }
                fseek(archivo,sizeof(Transaccion)-sizeof(int),SEEK_CUR);
        }
        printf("Transaccion no encontrado\n");
        return NULL;
```

```
}
void imprimir_transaccion(FILE* archivo){
        if(archivo!=NULL){
                Transaccion trans;
                char tipo[14];
                fread(&trans, sizeof(Transaccion), 1,archivo);
                switch(trans.tipo){
                case 1:
                        strcpy(tipo,"RETIRO");break;
                case 2:
                        strcpy(tipo,"DEPOSITO");break;
                case 3:
                        strcpy(tipo,"TRANSFERENCIA");break;
                }
                printf("%-17d%-19s%-19d%-20d%-20d{%02d,%02d,%04d}\n",trans.id_transaccion
,tipo,trans.id_origen,trans.id_destino,trans.monto,
                                trans.fecha.dia,trans.fecha.mes,trans.fecha.anio);
        }
}
void lista_transacciones(FILE* archivo){
        char c;
        fseek(archivo,sizeof(int),SEEK_SET);
        printf("ID_TRANSACCION   TIPO_TRANSACCION   ID_CUENTA_ORIGEN   ID_CUENTA_DESTINO
MONTO_TRANSACCION   FECHA_TRANSACCION\n");
        while((c=fgetc(archivo))!=EOF){
                fseek(archivo,-sizeof(char),SEEK_CUR);
                imprimir_transaccion(archivo);
        }
}
```

Ejecución

```
<<Sistema MyDB>>
[1]     Clientes
[2]     Cuentas
[3]     Transacciones
[4]     Salir
1
<<CLIENTES>>
[1]     Nuevo
[2]     Buscar
[3]     Eliminar
[4]     Imprimir
1
Nombre: Jorge
Apellido Paterno: Paredes
Apellido Materno: Tizano
Fecha de nacimiento(dd/mm/aaaa): 28/09/2007
```

```
<<CLIENTES>>
[1]     Nuevo
[2]     Buscar
[3]     Eliminar
[4]     Imprimir
2
Id de usuario: 2
ID_USUARIO   NOMBRE              APELLIDO_PATERNO      APELLIDO_MATERNO      FECHA_NACIMIENTO
2            Luis                Enrique               Manzano               {14,02,1987}
<<CLIENTES>>
[1]     Nuevo
[2]     Buscar
[3]     Eliminar
[4]     Imprimir
3
Id de usuario: 2
```

```
<<CLIENTES>>
[1]     Nuevo
[2]     Buscar
[3]     Eliminar
[4]     Imprimir
4
ID_USUARIO   NOMBRE              APELLIDO_PATERNO      APELLIDO_MATERNO      FECHA_NACIMIENTO
1            Jose                Gallegos              Velica                {05,01,1999}
4            Aurelio             Rojas                 Parza                 {14,12,1994}
5            Jorge               Paredes               Tizano                {28,09,2007}
```

```
<<Sistema MyDB>>
[1]     Clientes
[2]     Cuentas
[3]     Transacciones
[4]     Salir
2
<<CUENTAS>>
[1]     Nueva
[2]     Buscar
[3]     Eliminar
[4]     Imprimir
1
Id del cliente: 4
Saldo de apertura: 7895
Fecha de apertura(dd/mm/aaaa): 19/05/1987
```

```
<<CUENTAS>>
[1]     Nueva
[2]     Buscar
[3]     Eliminar
[4]     Imprimir
2
Id de cuenta: 2
ID_CUENTA   ID_USUARIO   SALDO        FECHA_APERTURA
2           1            12000        {15,12,2005}
```

```
<<CUENTAS>>
[1]     Nueva
[2]     Buscar
[3]     Eliminar
[4]     Imprimir
3
Id de cuenta: 2
```

```
<<CUENTAS>>
[1]     Nueva
[2]     Buscar
[3]     Eliminar
[4]     Imprimir
4
ID_CUENTA   ID_USUARIO   SALDO        FECHA_APERTURA
1           1            1500         {12,14,1248}
2           1            12000        {15,12,2005}
3           4            7895         {19,05,1987}
```

Conclusiones :

Este trabajo me ayudo demasiado para manejar archivos, fue complicado y necesite de consultar algunas fuentes para entender algunas de las funciones y

varias pruebas para predecir los comportamientos del puntero al archivo, sobre todo con los modos de apertura. Estoy muy satisfecho de mi trabajo final.