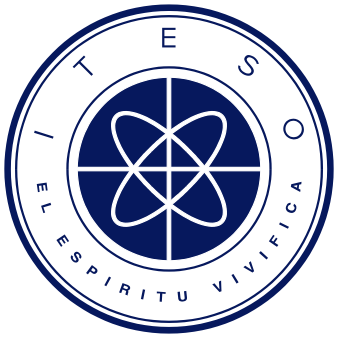
INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Departamento de Electrónica, Sistemas e Informática

INGENIERÍA EN SISTEMAS COMPUTACIONALES



PROGRAMACIÓN CON MEMORIA DINÁMICA

TAREA 2. MEMORIA DINÁMICA Y ARCHIVOS

Autor: Aguilar Aguirre, Alfonso

Presentación: 10 pts.
Funcionalidad: 50 pts.
Pruebas: 20 pts.

12 de junio de 2018. Tlaquepaque, Jalisco,

- Falta describir las pruebas (escenario, y resultados de la experimentación).
- Tienes algunos detalles en el tema de funcionalidad, verifica lo que se debe imprimir cuando consultas una cuenta.

Instrucciónes para entrega de tarea

Esta tarea, como el resto, es ***IMPRESCINDIBLE*** entregar los entregables de esta actividad de la siguiente manera:

- **Reporte:** vía *moodle* en **un archivo PDF**.
- **Código:** vía su repositorio **Github**.

La evaluación de la tarea comprende:

- 10% para la presentación
- 60% para la funcionalidad
- 30% para las pruebas

Es necesario responder el apartado de conclusiones, pero no se trata de llenarlo con paja. Si no se aprendió nada al hacer la práctica, es preferible escribir eso. Si el apartado queda vacío, se restarán puntos al porcentaje de presentación.

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;
    int mes;
    int ano;
}Fecha;
typedef struct{
    int id_cliente;
    char nombre[10];
    char apellidoP[10];
    char apellidoM[10];
    Fecha f;
}Cliente;
typedef struct{
    int id_cuenta;
    char id_usuario;
    int saldo;
    Fecha f_apertura;
}Cuenta;
typedef struct{
    int id_transaccion;
    char tipo[20];
    int cuenta_origen;
    int cuenta_destino;
    Fecha f_transaccion;
    int monto;
}Transaccion;

int main(void) {
```

```
char ruta1[400],ruta2[400],ruta3[400],ruta4[400],rutF[400],temp[400];
printf("ruta de carpeta: \n");
scanf("%[^\n]",&rutF);
scanf("%c",&temp);

strcpy(ruta1,rutF);
strcpy(ruta2,rutF);
strcpy(ruta3,rutF);
strcpy(ruta4,rutF);

strcat(ruta1,"/clientes.dat");
strcat(ruta2,"/cuentas.dat");
strcat(ruta3,"/transacciones.dat");
strcat(ruta4,"/mydb.sys");

FILE *clientes =fopen(ruta1,"ab+");
if(clientes==NULL)
    return -1;

FILE *cuentas =fopen(ruta2,"ab+");
if(cuentas==NULL)
    return -1;

FILE *transacciones =fopen(ruta3,"ab+");
if(transacciones==NULL)
    return -1;

FILE *mydb =fopen(ruta4,"wb+");
if(mydb==NULL)
    return -1;
fwrite(rutF,sizeof(char),400,mydb);

fclose(clientes);
fclose(cuentas);
fclose(transacciones);
fclose(mydb);

//    /Users/macuser/Desktop/Codes/PE_chess/PMD2018/MyDB/

int flag=1,flag2=1,option,option2;
while(flag){
    printf("<< Sistema MyDB >>\n");
    printf("1. Clientes\n");
    printf("2. Cuentas\n");
    printf("3. Transacciones\n");
    printf("4. Salir\n");
    scanf("%d",&option);
    fflush(stdout);
    switch(option){
        case 1:
            while(flag2){
                printf("1. Nuevo cliente\n");
                printf("2. Buscar cliente\n");
                printf("3. Eliminar cliente\n");
                printf("4. Imprimir clientes\n");
                printf("5. Salir\n");
                scanf("%d",&option2);
                fflush(stdout);
                switch(option2){
                    case 1:{
                        int temp3;
                        FILE *clientes = fopen(ruta1,"ab+");
                        if(clientes==NULL)
                            return -1;
                        fseek(clientes,-1*sizeof(Cliente),SEEK_END);
                        Cliente c1;
                        fread(&c1,sizeof(Cliente),1,clientes);
                        c1.id_cliente++;
                        printf("nombre: \n");
```

```
        scanf("%c",&temp);
        scanf("%[^\n]",c1.nombre);
        printf("apellido paterno: \n");
        scanf("%c",&temp);
        scanf("%[^\n]",c1.apellidoP);
        printf("apellido materno: \n");
        scanf("%c",&temp);
        scanf("%[^\n]",c1.apellidoM);
        scanf("%c",&temp);
        printf("dia: \n");
        scanf("%d",&temp3);
        c1.f.dia=temp3;
        printf("mes: \n");
        scanf("%d",&temp3);
        c1.f.mes=temp3;
        printf("año: \n");
        scanf("%d",&temp3);
        c1.f.ano=temp3;
        fwrite(&c1,sizeof(Cliente),1,clientes);
        fclose(clientes);
        break;
    }
    case 2:{
        int which_id;
        printf("cual quieres buscar: \n");
        scanf("%d",&which_id);
        FILE *clientes = fopen(ruta1,"rb");
        if(clientes==NULL)
            return -1;
        Cliente c2;
        fseek(clientes,-1*sizeof(Cliente),SEEK_CUR);
        fread(&c2,sizeof(Cliente),1,clientes);
        while(!feof(clientes)){
            if(c2.id_cliente==which_id){
                printf("%d  ",c2.id_cliente);
                printf("%s  ",c2.nombre);
                printf("%s  ",c2.apellidoP);
                printf("%s  ",c2.apellidoM);
                printf("%d,",c2.f.dia);
                printf("%d,",c2.f.mes);
                printf("%d\n",c2.f.ano);
                break;
            }
            fread(&c2,sizeof(Cliente),1,clientes);
        }
        fclose(clientes);
        break;
    }
    case 3:{
        int which_id;
        int i=0,j=0;
        printf("cual deseas eliminar: \n");
        scanf("%d",&which_id);
        FILE *clientes = fopen(ruta1,"rb");
        if(clientes==NULL)
            return -1;
        Cliente c2,arrcli[1000];
        fread(&c2,sizeof(Cliente),1,clientes);
        while(!feof(clientes)){
            if(c2.id_cliente!=which_id){
                arrcli[i]=c2;
                i++;
            }
            fread(&c2,sizeof(Cliente),1,clientes);
        }
        fclose(clientes);
        FILE *clientes1 = fopen(ruta1,"wb");
        for(j=0;j<i;j++)
            fwrite(&arrcli[j],sizeof(Cliente),1,clientes1);
```

```
                    fclose(clientes1);
                    break;
                }
                case 4:{
                    FILE *clientes = fopen(ruta1,"rb");
                    if(clientes==NULL)
                        return -1;
                    Cliente c2;
                    fread(&c2,sizeof(Cliente),1,clientes);
                    while(!feof(clientes)){
                        printf("%d  ",c2.id_cliente);
                        printf("%s  ",c2.nombre);
                        printf("%s  ",c2.apellidoP);
                        printf("%s  ",c2.apellidoM);
                        printf("%d,",c2.f.dia);
                        printf("%d,",c2.f.mes);
                        printf("%d\n",c2.f.ano);
                        fread(&c2,sizeof(Cliente),1,clientes);
                    }
                    fclose(clientes);
                    break;
                }
                case 5:
                    flag2=0;
                    break;
            }
        }
        flag2=1;
        break;
    case 2:
        while(flag2){
            printf("1. Nueva cuenta\n");
            printf("2. Buscar cuenta\n");
            printf("3. Eliminar cuenta\n");
            printf("4. Imprimir cuentas\n");
            printf("5. Salir\n");
            scanf("%d",&option2);
            switch(option2){
                case 1:{
                    int temp;
                    FILE *cuentas = fopen(ruta2,"ab+");
                    if(cuentas==NULL)
                        return -1;
                    fseek(cuentas,-1*sizeof(Cuenta),SEEK_END);
                    Cuenta cu1;
                    fread(&cu1,sizeof(Cuenta),1,cuentas);
                    cu1.id_cuenta++;
                    printf("a que id de cliente pertenece la cuenta: \n");
                    scanf("%d",&temp);
                    cu1.id_usuario=temp;
                    printf("saldo: \n");
                    scanf("%d",&temp);
                    cu1.saldo=temp;
                    printf("dia apertura: \n");
                    scanf("%d",&temp);
                    cu1.f_apertura.dia=temp;
                    printf("mes apertura: \n");
                    scanf("%d",&temp);
                    cu1.f_apertura.mes=temp;
                    printf("año apertura: \n");
                    scanf("%d",&temp);
                    cu1.f_apertura.ano=temp;
                    FILE *clientes = fopen(ruta1,"rb");
                    if(clientes==NULL)
                        return -1;
                    Cliente c2;
                    fseek(clientes,-1*sizeof(Cliente),SEEK_CUR);
                    fread(&c2,sizeof(Cliente),1,clientes);
                    while(!feof(clientes)){
```

```
            if(c2.id_cliente==cu1.id_usuario){
                fwrite(&cu1,sizeof(Cuenta),1,cuentas);
                fclose(cuentas);
                break;
            }
            fread(&c2,sizeof(Cliente),1,clientes);
        }
        fclose(clientes);
        break;
    }


    case 2:{
        int which_id;
        printf("que cuanta quieres buscar: \n");
        scanf("%d",&which_id);
        FILE *cuentas = fopen(ruta2,"rb");
        if(cuentas==NULL)
            return -1;
        Cuenta cu2;
        fseek(cuentas,-1*sizeof(Cliente),SEEK_CUR);
        fread(&cu2,sizeof(Cuenta),1,cuentas);
        while(!feof(cuentas)){
            if(cu2.id_cuenta==which_id){
                printf("%d  ",cu2.id_cuenta);
                printf("%d  ",cu2.id_usuario);
                printf("%d  ",cu2.saldo);
                printf("%d,",cu2.f_apertura.dia);
                printf("%d,",cu2.f_apertura.mes);
                printf("%d\n",cu2.f_apertura.ano);
                break;
            }
            fread(&cu2,sizeof(Cuenta),1,cuentas);
        }
        fclose(cuentas);
        break;
    }
    case 3:{
        int which_id;
        int i=0,j=0;
        printf("que cuenta quieres eliminar: \n");
        scanf("%d",&which_id);
        FILE *cuentas = fopen(ruta2,"rb");
        if(cuentas==NULL)
            return -1;
        Cuenta cu2,arrcue[1000];
        fread(&cu2,sizeof(Cuenta),1,cuentas);
        while(!feof(cuentas)){
            if(cu2.id_cuenta!=which_id){
                arrcue[i]=cu2;
                i++;
            }
            fread(&cu2,sizeof(Cuenta),1,cuentas);
        }
        fclose(cuentas);
        FILE *cuentas1 = fopen(ruta2,"wb");
        for(j=0;j<i;j++)
            fwrite(&arrcue[j],sizeof(Cuenta),1,cuentas1);
        fclose(cuentas1);
        break;
    }
    case 4:{
        FILE *cuentas = fopen(ruta2,"rb");
        if(cuentas==NULL)
            return -1;
        Cuenta cu2;
        fread(&cu2,sizeof(Cuenta),1,cuentas);
        while(!feof(cuentas)){
            printf("%d  ",cu2.id_cuenta);
```

```
                    printf("%d  ",cu2.id_usuario);
                    printf("%d  ",cu2.saldo);
                    printf("%d,",cu2.f_apertura.dia);
                    printf("%d,",cu2.f_apertura.mes);
                    printf("%d\n",cu2.f_apertura.ano);
                    fread(&cu2,sizeof(Cuenta),1,cuentas);
                }
                fclose(cuentas);
                break;
            }
            case 5:
                flag2=0;
                break;
        }
    }
    flag2=1;

    break;
case 3:
    while(flag2){
        printf("1. Déposito\n");
        printf("2. Retiro\n");
        printf("3. Transferencia\n");
        printf("4. Salir\n");
        scanf("%d",&option2);
        switch(option2){
            case 1:{

                int temp3;
                FILE *transacciones = fopen(ruta3,"ab+");
                if(transacciones==NULL)
                    return -1;
                fseek(transacciones,-1*sizeof(Transaccion),SEEK_END);
                Transaccion t1;
                fread(&t1,sizeof(Transaccion),1,transacciones);
                t1.id_transaccion++;
                strcpy(t1.tipo,"deposito");
                t1.cuenta_origen=NULL;
                printf("cuenta destino: \n");
                scanf("%d",&t1.cuenta_destino);
                printf("dia transaccion: \n");
                scanf("%d",&temp3);
                t1.f_transaccion.dia=temp3;
                printf("mes transaccion: \n");
                scanf("%d",&temp3);
                t1.f_transaccion.mes=temp3;
                printf("año transaccion: \n");
                scanf("%d",&temp3);
                t1.f_transaccion.ano=temp3;
                printf("monto: \n");
                scanf("%d",&t1.monto);


                Cuenta cu2,arrcue[1000];
                int i=0,j;
                FILE *cuentas = fopen(ruta2,"rb");
                if(cuentas==NULL)
                    return -1;
                fseek(cuentas,-1*sizeof(Cuenta),SEEK_CUR);
                fread(&cu2,sizeof(Cuenta),1,cuentas);
                while(!feof(cuentas)){
                    if(cu2.id_cuenta==t1.cuenta_destino){
                        cu2.saldo+=t1.monto;
                        arrcue[i]=cu2;
                        fwrite(&t1,sizeof(Transaccion),1,transacciones);
                        fclose(transacciones);
                    }
                    arrcue[i]=cu2;
                    i++;
```

```
            fread(&cu2,sizeof(Cuenta),1,cuentas);
        }
        fclose(cuentas);
        FILE *cuentas1 = fopen(ruta2,"wb");
        for(j=0;j<i;j++)
            fwrite(&arrcue[j],sizeof(Cuenta),1,cuentas1);
        fclose(cuentas1);
        break;
    }
    case 2:{
        int temp3;
        FILE *transacciones = fopen(ruta3,"ab+");
        if(transacciones==NULL)
            return -1;
        fseek(transacciones,-1*sizeof(Transaccion),SEEK_END);
        Transaccion t1;
        fread(&t1,sizeof(Transaccion),1,transacciones);
        t1.id_transaccion++;
        strcpy(t1.tipo,"retiro");
        t1.cuenta_destino=NULL;
        printf("cuenta origen: \n");
        scanf("%d",&t1.cuenta_origen);
        printf("dia transferencia: \n");
        scanf("%d",&temp3);
        t1.f_transaccion.dia=temp3;
        printf("mes transferencia: \n");
        scanf("%d",&temp3);
        t1.f_transaccion.mes=temp3;
        printf("año transferencia: \n");
        scanf("%d",&temp3);
        t1.f_transaccion.ano=temp3;
        printf("monto: \n");
        scanf("%d",&t1.monto);


        Cuenta cu2,arrcue[1000];
        int i=0,j;
        FILE *cuentas = fopen(ruta2,"rb");
        if(cuentas==NULL)
            return -1;
        fseek(cuentas,-1*sizeof(Cuenta),SEEK_CUR);
        fread(&cu2,sizeof(Cuenta),1,cuentas);
        while(!feof(cuentas)){
            if(cu2.id_cuenta==t1.cuenta_origen){
                if(cu2.saldo>=t1.monto){
                    cu2.saldo-=t1.monto;
                    arrcue[i]=cu2;
                    fwrite(&t1,sizeof(Transaccion),1,transacciones);
                    fclose(transacciones);
                }
                else
                    printf("saldo insuficiente");
            }
            arrcue[i]=cu2;
            i++;
            fread(&cu2,sizeof(Cuenta),1,cuentas);
        }
        fclose(cuentas);
        FILE *cuentas1 = fopen(ruta2,"wb");
        for(j=0;j<i;j++)
            fwrite(&arrcue[j],sizeof(Cuenta),1,cuentas1);
        fclose(cuentas1);
        break;
    }
    case 3:{
        int temp3;
        FILE *transacciones = fopen(ruta3,"ab+");
        if(transacciones==NULL)
            return -1;
```

```
                            fseek(transacciones,-1*sizeof(Transaccion),SEEK_END);
                            Transaccion t1;
                            fread(&t1,sizeof(Transaccion),1,transacciones);
                            t1.id_transaccion++;
                            strcpy(t1.tipo,"transferencia");
                            printf("cuenta origen: \n");
                            scanf("%d",&t1.cuenta_origen);
                            printf("cuenta destino: \n");
                            scanf("%d",&t1.cuenta_destino);
                            printf("dia transferencia: \n");
                            scanf("%d",&temp3);
                            t1.f_transaccion.dia=temp3;
                            printf("mes transferencia: \n");
                            scanf("%d",&temp3);
                            t1.f_transaccion.mes=temp3;
                            printf("año transferencia: \n");
                            scanf("%d",&temp3);
                            t1.f_transaccion.ano=temp3;
                            printf("monto: \n");
                            scanf("%d",&t1.monto);


                            Cuenta cu2,cu3,arrcue[1000];
                            int i=0,j,k=0;
                            FILE *cuentas = fopen(ruta2,"rb");
                            if(cuentas==NULL)
                                return -1;
                            fseek(cuentas,-1*sizeof(Cuenta),SEEK_CUR);
                            fread(&cu2,sizeof(Cuenta),1,cuentas);
                            while(!feof(cuentas)){

                                if(cu2.id_cuenta==t1.cuenta_origen){
                                    rewind(cuentas);
                                    fread(&cu3,sizeof(Cuenta),1,cuentas);
                                    while(!feof(cuentas)){
                                        if(cu3.id_cuenta==t1.cuenta_destino){
                                            if(cu2.saldo>=t1.monto){
                                                cu2.saldo-=t1.monto;
                                                cu3.saldo+=t1.monto;
                                                arrcue[k]=cu2;
                                                arrcue[i]=cu3;

fwrite(&t1,sizeof(Transaccion),1,transacciones);
                                                fclose(transacciones);
                                            }
                                            else
                                                printf("saldo insuficiente");
                                        }
                                        arrcue[i]=cu3;
                                        i++;
                                        fread(&cu3,sizeof(Cuenta),1,cuentas);
                                    }
                                    break;
                                }
                                k++;
                                fread(&cu2,sizeof(Cuenta),1,cuentas);
                            }
                            fclose(cuentas);
                            FILE *cuentas1 = fopen(ruta2,"wb");
                            for(j=0;j<i;j++)
                                fwrite(&arrcue[j],sizeof(Cuenta),1,cuentas1);
                            fclose(cuentas1);
                            break;
                        }
                        case 4:
                            flag2=0;
                            break;
                    }
                }
```

```
                flag2=1;
                break;
            case 4:
                flag=0;
                break;
        }
    }
    return 0;
}
```

Ejecución

```
ruta de carpeta:
/Users/macuser/Desktop/Codes/PE_chess/PMD2018/
MyDB/
<< Sistema MyDB >>
1. Clientes
2. Cuentas
3. Transacciones
4. Salir

1
1. Nuevo cliente
2. Buscar cliente
3. Eliminar cliente
4. Imprimir clientes
5. Salir

1
nombre:
alfonso
apellido paterno:
aguilar
apellido materno:
aguirre
dia:
3
mes:
4
año:
2018

cual quieres buscar:
3
3  paco  aguilar  aguirre  5,6,4018

3
cual deseas eliminar:
3

1  alfonso  aguilar  aguirre  1,2,2018
4  rigoberto  perez  chave  4,5,7018
5  alfonso  aguilar  aguirre  3,4,2018
```

```
2
1. Nueva cuenta
2. Buscar cuenta
3. Eliminar cuenta
4. Imprimir cuentas
5. Salir


1
a que id de cliente pertenece la cuenta:
5
saldo:
9000
dia apertura:
2
mes apertura:
2
año apertura:
2020
2
que cuanta quieres buscar:
4
4  5  9000  2,2,2020

que cuenta quieres eliminar:
4


4
2  3  9000  6,6,3040
3  1  20000  3,3,2018
3
1. Déposito
2. Retiro
3. Transferencia
4. Salir


1
cuenta destino:
2
dia transaccion:
2
mes transaccion:
5
año transaccion:
2019
monto:
5000
```

```
2
cuenta origen:
4
dia transferencia:
2
mes transferencia:
7
año transferencia:
2019
monto:
3000

3
cuenta origen:
2
cuenta destino:
3
dia transferencia:
5
mes transferencia:
6
año transferencia:
2019
monto:
2000

2  3  12000  6,6,3040
3  1  22000  3,3,2018
```

Conclusiones (obligatorio):

- ✓ Lo que aprendí con esta práctica. Lo que ya sabía.
  Aprendi a manejar archivos binarios, ya que anteriormente habia trabajado con archivos de texto.
- ✓ Lo que me costó trabajo y cómo lo solucioné.
  Lo que mas me costo trabajo fue el manejo adecuado con las cadenas de texto al leerlas del teclado, lo solucione con variables temporales que escaneaban cualquier cosa que se haya quedado anteriormente en el buffer.
- ✓ Lo que no pude solucionar.
  No pude solucionar una implementacion mas simple con TDA debido al tiempo, comence una implementacion con la intención de manejar todo en memoria dinámica y al final pasarlo al archivo, aunque como recomendación del profesor era mas seguro estar manejando directamente con los archivos escribiendo y leyendo.