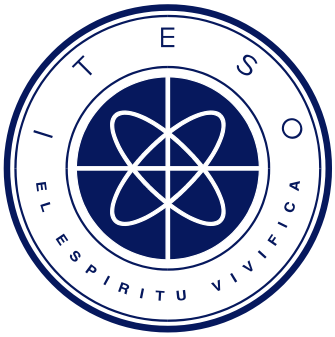
INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Departamento de Electrónica, Sistemas e Informática

INGENIERÍA EN SISTEMAS COMPUTACIONALES



PROGRAMACIÓN CON MEMORIA DINÁMICA

TAREA 2. MEMORIA DINÁMICA Y ARCHIVOS

Autor: Medina Rifas Edgar Francisco

Presentación: 5 pts.
Funcionalidad: 60 pts.
Pruebas: 20 pts.

12 de junio de 2018. Tlaquepaque, Jalisco,

- Falta describir las pruebas (escenario, y resultados de la experimentación).
- Sobre figuras, tengo los mismos comentarios que realice en tu tarea 1 y 2.

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

```
/*Edgar Francisco Medina Rifas
IS715468
Instituto Técnologico de Estudios Superiores
Programación con Memoria Dinámica
Dr. Francisco Cervantes.
*/
//Librerias y Constantes
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define enter printf("\n");
enum {Clientes=1, Cuentas, Transacciones, Salir};
//Prototipo de funciones y estructuras
typedef struct
{
	int dd, mm, aa;
}fecha;
typedef struct
{
	int id;
	char nombre[30];
	char apellidoP[20];
	char apellidoM[20];
	fecha f;
}cliente;

typedef struct
{
	int idCl, idCu, saldo;
	fecha date;
}cuentas;
```

```
typedef struct
{
	int idT;
	char tipoT[20];
	int idCueOri, idCueDes;
	fecha dateT;
	int money;
}transacciones;
//CLIENTES
void addCl (FILE * archivo, char ruta[100]);
void searchCl(int num,FILE *archivo, char ruta[200]);
void deleteCl(int num, FILE *archivo, char ruta[200], FILE *archivo2, char
ruta2[200]);
void printCl(FILE *archivo, char ruta[200]);
//CUENTAS
void addCU (int num, FILE* archivo, char ruta[200], FILE* archivo2, char
ruta2[200]);
void searchCU(int num,FILE *archivo, char ruta[200],FILE *archivo2, char
ruta2[200]);
void printCU(FILE *archivo, char ruta[200],FILE *archivo2, char ruta2[200]);
void deleteCU(int num, FILE *archivo, char ruta[200]);
void deleteClCU(int num, FILE *archivo, char ruta[200]);
//TRANSACCIONES
void Deposito(int idDest,  FILE* cu, char rutacu[200], FILE* tr, char
rutatr[200]);
void Retirar (int idOri,  FILE* cu, char rutacu[200], FILE* tr, char
rutatr[200]);
void Transaccion(int idOri, int idDest,  FILE* cu, char rutacu[200], FILE* tr,
char rutatr[200]);
void printTr(FILE *archivo, char ruta[200]);
int main()
{
//Buffer
	setvbuf(stderr, NULL, _IONBF, 0);
	setvbuf(stdout, NULL, _IONBF, 0);
	FILE *CLIENTES;
	FILE *CUENTAS;
	FILE *TRANSACCIONES;
	FILE *MYDB;
	int choice1, choice2;
	int buscar, buscar2;
	char ruta[200], ruta2[200];
	char C[30]="\\Clientes.dat";
	char D[30]="\\Cuentas.dat";
	char E[30]="\\Transacciones.dat";
	char rC[300], rD[300], rE[300];
	MYDB=fopen("mydb.sys", "r");
	if(MYDB==NULL)
	{
		printf("Como es la primera vez, ingresa la ruta donde quieres que
se guarden tus archivos: ");
		gets(ruta);
		strcpy(rC,ruta);
		strcat(rC,C);
```

```
                strcpy(rD,ruta);
                strcat(rD,D);
                strcpy(rE,ruta);
                strcat(rE,E);
                MYDB=fopen("mydb.sys","w+");
                fprintf(MYDB,"%s",ruta);
                fclose(MYDB);
                int first=0;
                CLIENTES=fopen(rC,"wb");
                fwrite(&first,sizeof(int),1,CLIENTES);
                if(CLIENTES==NULL)
                {
                        printf("Error al abrir el archivo.\n");
                }
                CUENTAS=fopen(rD,"wb");
                fwrite(&first,sizeof(int),1,CUENTAS);
                if(CUENTAS==NULL)
                {
                        printf("Error al abrir el archivo.\n");
                }
                TRANSACCIONES=fopen(rE,"wb");
                fwrite(&first,sizeof(int),1,TRANSACCIONES);
                if(TRANSACCIONES==NULL)
                {
                        printf("Error al abrir el archivo.\n");
                }
                fclose(CLIENTES);
                fclose(CUENTAS);
                fclose(TRANSACCIONES);
        }
        else
        {
                fscanf(MYDB,"%s",ruta2);
                strcpy(rC,ruta2);
                strcat(rC,C);
                strcpy(rD,ruta2);
                strcat(rD,D);
                strcpy(rE,ruta2);
                strcat(rE,E);
                fclose(MYDB);
        }
        do
        {
                printf("<<Sistema MyDB>>");
                enter
                printf("1.- Clientes\n2.- Cuentas\n3.- Transacciones\n4.- Salir");
                enter
                scanf("%d",&choice1);
                switch (choice1)
                {
                case Clientes:
                        printf("1.- Nuevo cliente\n2.- Buscar cliente\n3.- Eliminar
cliente\n4.- Imprimir clientes");
                        enter
                        scanf("%d",&choice2);
```

```
                    switch(choice2)
                    {
                    case 1:
                           addCl(CLIENTES, rC);
                           break;
                    case 2:
                           printf("Ingresa el ID del usuario que quieres buscar:
");
                           scanf("%d",&buscar);
                           searchCl(buscar,CLIENTES, rC);
                           break;
                    case 3:
                           printf("Ingresa el ID del usuario que quieres
eliminar: ");
                           scanf("%d",&buscar);
                           deleteCl(buscar, CLIENTES, rC, CUENTAS, rD);
                           break;
                    case 4:
                           printCl(CLIENTES, rC);
                           break;
                    }
                    break;
             case Cuentas:
                    printf("1.- Nuevo cuenta\n2.- Buscar cuenta\n3.- Eliminar
cuenta\n4.- Imprimir cuentas\n");
                    enter
                    scanf("%d",&choice2);
                    switch(choice2)
                    {
                    case 1:
                           printf("Ingresa el id del cliente al que pertenecerá
esta cuenta: ");
                           scanf("%d",&buscar);
                           addCU(buscar, CUENTAS, rD, CLIENTES, rC);
                           break;
                    case 2:
                           printf("Ingresa el id de la cuenta que busca: ");
                           scanf("%d",&buscar);
                           searchCU(buscar,CUENTAS, rD, CLIENTES, rC);
                           break;
                    case 3:
                           printf("Ingresa el id de la cuenta que desea borrar:
");
                           scanf("%d",&buscar);
                           deleteCU(buscar, CUENTAS, rD);
                           break;
                    case 4:
                           printCU(CUENTAS, rD, CLIENTES, rC);
                           break;
                    }
                    break;
             case Transacciones:
                    printf("1.- Deposito\n2.- Retiro\n3.- Transferencia\n4.-
Imprimir transacciones");
                    enter
```

```
                    scanf("%d",&choice2);
                    switch(choice2)
                    {
                    case 1:
                          printf("Ingresa el id de la cuenta a la que deseas
depositar: ");
                          scanf("%d",&buscar);
                          Deposito(buscar,  CUENTAS, rD, TRANSACCIONES, rE);
                          break;
                    case 2:
                          printf("Ingresa el id de la cuenta de la que deseas
retirar: ");
                          scanf("%d",&buscar);
                          Retirar (buscar,  CUENTAS, rD, TRANSACCIONES, rE);
                          break;
                    case 3:
                          printf("-Transferencia-");
                          enter
                          printf("Ingresa el id de la cuenta de la que deseas
retirar: ");
                          scanf("%d",&buscar);
                          printf("Ingresa el id de la cuenta a la que deseas
depositar: ");
                          scanf("%d",&buscar2);
                          Transaccion(buscar, buscar2, CUENTAS, rD,
TRANSACCIONES, rE);
                          break;
                    case 4:
                          printTr(TRANSACCIONES, rE);
                          break;
                    }
                    break;
             case Salir:

                    return 0;
             }
             do
             {
                    printf("¿Deseas limpiar la pantalla?\n(1) Si\t(2) No");
                    enter
                    scanf("%d",&buscar2);
                    enter
                    if(buscar2==1)
                    {
                          system("cls");
                    }
             }while(buscar2<1 || buscar2>2);

      }while (choice1!=4 && (choice1>0 && choice1<4));

      return 0;
}
//Desarrollo de funciones
/*
 *********CLIENTES********
```

```
 */
void addCl (FILE * archivo, char ruta[200])
{
	int id;
	archivo=fopen(ruta,"rb");
	fread(&id, sizeof(int),1,archivo);
	fclose(archivo);
	cliente in;
	id++;
	archivo=fopen(ruta, "a+b");
	in.id=id;
	printf("Ingresa el nombre:");
	enter
	scanf("%s",in.nombre);
	printf("Ingresa el apellido paterno:");
	enter
	scanf("%s",in.apellidoP);
	printf("Ingresa el apellido materno:");
	enter
	scanf("%s",in.apellidoM);
	printf("Ingresa la fecha de nacimiento dd/mm/aa:");
	enter
	scanf("%d/%d/%d",&in.f.dd,&in.f.mm,&in.f.aa);
	fwrite(&in,sizeof(cliente),1,archivo);
	fclose(archivo);
	archivo=fopen(ruta, "r+b");
	fwrite(&id, sizeof(int),1, archivo);
	fclose(archivo);
}

void searchCl(int num,FILE *archivo, char ruta[200])
{
	int x, true=0;
	archivo=fopen(ruta, "r+b");
	fread(&x, sizeof(int),1,archivo);
	cliente prueba;
	while(fread(&prueba,sizeof(cliente),1,archivo)==1)
	{
		if(num==prueba.id)
		{
			printf("%-3s\t%-30s\t%-30s\t%-30s\t%-
5s\n","ID","Nombre","Apellido Paterno", "Apellido Materno", "Fecha");
			printf("%d\t%-30s\t%-30s\t%-
30s\t%02d/%02d/%04d\n",prueba.id,prueba.nombre,prueba.apellidoP,prueba.apellidoM
,prueba.f.dd,prueba.f.mm,prueba.f.aa);
			true=1;
		}
	}
	if(true==0)
	{
		printf("Este ID no existe.\n");
	}
	fclose(archivo);
}
```

```
void deleteCl(int num, FILE *archivo, char ruta[200], FILE *archivo2, char
ruta2[200])
{
	int x, cont=0, true=0;
	archivo=fopen(ruta, "r+b");
	fread(&x, sizeof(int),1,archivo);
	cliente prueba;
	cliente *array;
	cliente *arrayt=(cliente *)malloc(sizeof(cliente));
	while(fread(&prueba,sizeof(cliente),1,archivo)==1)
	{
		if(num!=prueba.id && arrayt!=NULL)
		{
			(arrayt+cont)->id=prueba.id;
			strcpy((arrayt+cont)->nombre,prueba.nombre);
			strcpy((arrayt+cont)->apellidoP,prueba.apellidoP);
			strcpy((arrayt+cont)->apellidoM,prueba.apellidoM);
			(arrayt+cont)->f.dd=prueba.f.dd;
			(arrayt+cont)->f.mm=prueba.f.mm;
			(arrayt+cont)->f.aa=prueba.f.aa;
			cont++;
			if(arrayt!=NULL)
			{
				array=arrayt;
			}
			arrayt=(cliente *) realloc(array, sizeof(cliente)*(cont+1));
		}
		else
		{
			true=1;
		}
	}
	fclose(archivo);
	if(true==0)
	{
		printf("Este ID no está en el archivo.\n");
		return;
	}
	archivo=fopen(ruta, "wb");
	fwrite(&x, sizeof(int),1,archivo);
	fclose(archivo);
	archivo=fopen(ruta,"a+b");
	fwrite(array,sizeof(cliente),cont--,archivo);
	fclose(archivo);
	free(array);
	free(arrayt);
	deleteClCU(num, archivo2, ruta2);
}

void printCl(FILE *archivo, char ruta[200])
{
	int x;
	archivo=fopen(ruta, "rb");
	fread(&x, sizeof(int),1,archivo);
	cliente prueba;
```

```
	printf("%-3s\t%-30s\t%-30s\t%-30s\t%-5s\n","ID","Nombre","Apellido
Paterno", "Apellido Materno", "Fecha");
	while(fread(&prueba,sizeof(cliente),1,archivo)==1)
	{
		printf("%d\t%-30s\t%-30s\t%-
30s\t%02d/%02d/%04d\n",prueba.id,prueba.nombre,prueba.apellidoP,prueba.apellidoM
,prueba.f.dd,prueba.f.mm,prueba.f.aa);
	}
	fclose(archivo);
}

/*
 *******CUENTAS*******
 */
void addCU (int num, FILE* archivo, char ruta[200], FILE* archivo2, char
ruta2[200])
{
	int x, y, true=0;
	archivo=fopen(ruta,"r+b");
	fread(&x, sizeof(int),1,archivo);
	archivo2=fopen(ruta2,"r+b");
	fread(&y, sizeof(int),1,archivo2);
	cuentas prueba;
	cliente prueba2;
	while(fread(&prueba2,sizeof(cliente),1,archivo2)==1)
		{
		if(prueba2.id == num)
			{
				true=1;
			}
		}
	if(true==0)
	{
		printf("Este ID no existe.\n");
		return;
	}
	fclose(archivo);
	fclose(archivo2);
	x++;
	archivo=fopen(ruta, "a+b");
	prueba.idCl=num;
	prueba.idCu=x;
	printf("Ingresa saldo de cuenta:");
	enter
	scanf("%d",&prueba.saldo);
	printf("Ingresa la fecha de registro dd/mm/aa:");
	enter
	scanf("%d/%d/%d",&prueba.date.dd,&prueba.date.mm,&prueba.date.aa);
	fwrite(&prueba,sizeof(cuentas),1,archivo);
	fclose(archivo);
	archivo=fopen(ruta, "r+b");
	fwrite(&x, sizeof(int),1, archivo);
	fclose(archivo);
}
```

```
void searchCU(int num,FILE *archivo, char ruta[200],FILE *archivo2, char
ruta2[200])
{
	int x, y, true=0;
	archivo=fopen(ruta, "r+b");
	fread(&x, sizeof(int),1,archivo);
	archivo2=fopen(ruta2, "r+b");
	fread(&y, sizeof(int),1,archivo2);
	cuentas prueba;
	cliente prueba2;
	while(fread(&prueba,sizeof(cuentas),1,archivo)==1)
	{
		if(num==prueba.idCu)
		{
			printf("%3s\t%-25s\t%-25s\t%-20s\n","ID Cuenta","Nombre del
Cliente","Saldo", "Fecha registro");
			printf("%-16d\t",prueba.idCu);
			while(fread(&prueba2,sizeof(cliente),1,archivo2)==1)
			{
				if(prueba2.id == prueba.idCl)
				{
					printf("%-20s\t",prueba2.nombre);
				}
			}
			printf("%-
30d\t%02d/%02d/%04d\n",prueba.saldo,prueba.date.dd,prueba.date.mm,
prueba.date.aa);
			true=1;
		}
	}
	if(true==0)
	{
		printf("Este ID no existe.\n");
	}
	fclose(archivo);
	fclose(archivo2);
}

void printCU(FILE *archivo, char ruta[200],FILE *archivo2, char ruta2[200])
{
	int x, y;
	archivo=fopen(ruta, "r+b");
	fread(&x, sizeof(int),1,archivo);
	archivo2=fopen(ruta2, "r+b");
	fread(&y, sizeof(int),1,archivo2);
	cuentas prueba;
	cliente prueba2;
	printf("%3s\t%-25s\t%-25s\t%-20s\n","ID Cuenta","Nombre del
Cliente","Saldo", "Fecha registro");
	while(fread(&prueba,sizeof(cuentas),1,archivo)==1)
	{
		printf("%-16d\t",prueba.idCu);
		while(fread(&prueba2,sizeof(cliente),1,archivo2)==1)
		{
			if(prueba2.id == prueba.idCl)
```

```
                        {
                                printf("%-20s\t",prueba2.nombre);
                        }
                }
                printf("%-
30d\t%02d/%02d/%04d\n",prueba.saldo,prueba.date.dd,prueba.date.mm,
prueba.date.aa);
                fseek(archivo2,sizeof(int),SEEK_SET);
        }
        fclose(archivo);
        fclose(archivo2);
}

void deleteCU(int num, FILE *archivo, char ruta[200])
{
        int x, cont=0, true=0;
        archivo=fopen(ruta, "r+b");
        fread(&x, sizeof(int),1,archivo);
        cuentas prueba;
        cuentas *array;
        cuentas *arrayt=(cuentas *)malloc(sizeof(cuentas));
        while(fread(&prueba,sizeof(cuentas),1,archivo)==1)
        {
                if(num!=prueba.idCu)
                {
                        (arrayt + cont)->idCl=prueba.idCl;
                        (arrayt + cont)->idCu=prueba.idCu;
                        (arrayt + cont)->saldo=prueba.saldo;
                        (arrayt + cont)->date.dd=prueba.date.dd;
                        (arrayt + cont)->date.mm=prueba.date.mm;
                        (arrayt + cont)->date.aa=prueba.date.aa;
                        cont++;
                        if(arrayt!=NULL)
                        {
                                array=arrayt;
                        }
                        arrayt=(cuentas *) realloc(array, sizeof(cuentas)*(cont+1));
                }
                else
                {
                        true=1;
                }
        }
        fclose(archivo);
        if(true==0)
        {
                printf("Este ID no está en el archivo.\n");
                return;
        }
        archivo=fopen(ruta, "wb");
        fwrite(&x, sizeof(int),1,archivo);
        fclose(archivo);
        archivo=fopen(ruta,"a+b");
        fwrite(array,sizeof(cuentas),cont--,archivo);
        fclose(archivo);
```

```
	free(array);
	free(arrayt);
}


void deleteClCU(int num, FILE *archivo, char ruta[200])
{
	int x, cont=0, true=0;
	archivo=fopen(ruta, "r+b");
	fread(&x, sizeof(int),1,archivo);
	cuentas prueba;
	cuentas *array;
	cuentas *arrayt=(cuentas *)malloc(sizeof(cuentas));
	while(fread(&prueba,sizeof(cuentas),1,archivo)==1)
	{
		if(num!=prueba.idCl)
		{
			(arrayt + cont)->idCl=prueba.idCl;
			(arrayt + cont)->idCu=prueba.idCu;
			(arrayt + cont)->saldo=prueba.saldo;
			(arrayt + cont)->date.dd=prueba.date.dd;
			(arrayt + cont)->date.mm=prueba.date.mm;
			(arrayt + cont)->date.aa=prueba.date.aa;
			if(arrayt!=NULL)
			{
				array=arrayt;
			}
			cont++;
			array=(cuentas *) realloc(array, sizeof(cuentas)*(cont+1));
		}
		else
		{
			true=1;
		}
	}
	fclose(archivo);
	if(true==0)
	{
		printf("Este ID no está en el archivo cuentas.\n");
		return;
	}
	archivo=fopen(ruta, "wb");
	fwrite(&x, sizeof(int),1,archivo);
	fclose(archivo);
	archivo=fopen(ruta,"a+b");
	fwrite(array,sizeof(cuentas),cont--,archivo);
	fclose(archivo);
	free(array);
	free(arrayt);
}

/*
 * ********* TRANSACCIONES ***********
 */
```

```c
void Deposito(int idDest,  FILE* cu, char rutacu[200], FILE* tr, char
rutatr[200])
{
	int x, y, true=0;
	tr=fopen(rutatr,"r+b");
	fread(&x, sizeof(int),1,tr);
	cu=fopen(rutacu,"r+b");
	fread(&y, sizeof(int),1,cu);
	transacciones prueba;
	cuentas prueba2;
	while(fread(&prueba2,sizeof(cuentas),1,cu)==1)
		{
		if(prueba2.idCu == idDest)
			{
				true=1;
			}
		}
	if(true==0)
	{
		printf("Este ID no existe.\n");
		return;
	}
	fclose(tr);
	fclose(cu);
	x++;
	tr=fopen(rutatr, "a+b");
	strcpy(prueba.tipoT,"Deposito");
	prueba.idT=x;
	prueba.idCueOri=0;
	prueba.idCueDes=idDest;
	printf("Monto:");
	enter
	scanf("%d",&prueba.money);
	printf("Ingresa la fecha del deposito dd/mm/aa:");
	enter
	scanf("%d/%d/%d",&prueba.dateT.dd,&prueba.dateT.mm,&prueba.dateT.aa);
	fwrite(&prueba,sizeof(transacciones),1,tr);
	fclose(tr);
	tr=fopen(rutatr, "r+b");
	fwrite(&x, sizeof(int),1, tr);
	fclose(tr);
	cu=fopen(rutacu, "r+b");
	cuentas temp;
	fseek(cu, sizeof(int), 1);
	while(fread(&temp, sizeof(cuentas),1,cu)==1)
	{
		if(temp.idCu==idDest)
		{
			temp.saldo+=prueba.money;
			fseek(cu, -sizeof(cuentas), SEEK_CUR);
			fwrite(&temp, sizeof(cuentas), 1, cu);
			fclose(cu);
			return;
		}
	}
```

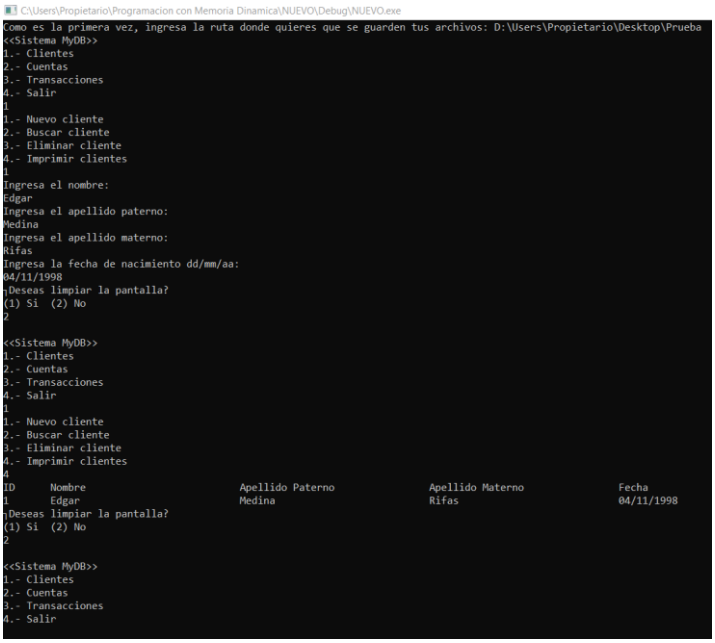
```
}

void Retirar (int idOri,  FILE* cu, char rutacu[200], FILE* tr, char
rutatr[200])
{
	int x, y, true=0;
	tr=fopen(rutatr,"r+b");
	fread(&x, sizeof(int),1,tr);
	cu=fopen(rutacu,"r+b");
	fread(&y, sizeof(int),1,cu);
	transacciones prueba;
	cuentas prueba2;
	while(fread(&prueba2,sizeof(cuentas),1,cu)==1)
		{
		if(prueba2.idCu == idOri)
			{
				true=1;
			}
		}
	if(true==0)
	{
		printf("Este ID no existe.\n");
		return;
	}
	fclose(tr);
	fclose(cu);
	x++;
	tr=fopen(rutatr, "a+b");
	strcpy(prueba.tipoT,"Retiro");
	prueba.idT=x;
	prueba.idCueOri=idOri;
	prueba.idCueDes=0;
	printf("Monto:");
	enter
	scanf("%d",&prueba.money);
	printf("Ingresa la fecha del retiro dd/mm/aa:");
	enter
	scanf("%d/%d/%d",&prueba.dateT.dd,&prueba.dateT.mm,&prueba.dateT.aa);
	cu=fopen(rutacu,"rb");
	fseek(cu, sizeof(int), SEEK_SET);
	cuentas temp;
	while(fread(&temp,sizeof(cuentas),1,cu)==1)
	{
		if(temp.idCu==prueba.idCueOri)
		{
			if(temp.saldo<prueba.money)
			{
				printf("Saldo insuficiente\n");
				fclose(cu);
				return;
			}
		}
	}
	fwrite(&prueba,sizeof(transacciones),1,tr);
	fclose(tr);
```

```
	tr=fopen(rutatr, "r+b");
	fwrite(&x, sizeof(int),1, tr);
	fclose(tr);
	cu=fopen(rutacu, "r+b");
	cuentas temp2;
	fseek(cu, sizeof(int), 1);
	while(fread(&temp2, sizeof(cuentas),1,cu)==1)
	{
		if(temp2.idCu==idOri)
		{
			temp2.saldo+=-1*prueba.money;
			fseek(cu, -sizeof(cuentas), SEEK_CUR);
			fwrite(&temp2, sizeof(cuentas), 1, cu);
			fclose(cu);
			return;
		}
	}
}

void Transaccion(int idOri, int idDest,  FILE* cu, char rutacu[200], FILE* tr,
char rutatr[200])
{
	if(idOri==idDest)
	{
		printf("No se pueden hacer transferencias a una cuenta misma.\n");
		return;
	}
	int x, y, true=0;
	tr=fopen(rutatr,"r+b");
	fread(&x, sizeof(int),1,tr);
	cu=fopen(rutacu,"r+b");
	fread(&y, sizeof(int),1,cu);
	transacciones prueba;
	cuentas prueba2;
	while(fread(&prueba2,sizeof(cuentas),1,cu)==1)
		{
		if(prueba2.idCu == idOri)
			{
				true=1;
			}
		}
	if(true==0)
	{
		printf("Este ID no existe.\n");
		return;
	}
	true=0;
	fseek(cu, sizeof(int),SEEK_SET);
	while(fread(&prueba2,sizeof(cuentas),1,cu)==1)
		{
		if(prueba2.idCu == idDest)
			{
				true=1;
			}
		}
```

```
if(true==0)
{
	printf("Este ID no existe.\n");
	return;
}
fclose(tr);
fclose(cu);
x++;
tr=fopen(rutatr, "a+b");
strcpy(prueba.tipoT,"Transaccion");
prueba.idT=x;
prueba.idCueOri=idOri;
prueba.idCueDes=idDest;
printf("Monto:");
enter
scanf("%d",&prueba.money);
printf("Ingresa la fecha del retiro dd/mm/aa:");
enter
scanf("%d/%d/%d",&prueba.dateT.dd,&prueba.dateT.mm,&prueba.dateT.aa);
cu=fopen(rutacu,"rb");
fseek(cu, sizeof(int), SEEK_SET);
cuentas temp;
while(fread(&temp,sizeof(cuentas),1,cu)==1)
{
	if(temp.idCu==prueba.idCueOri)
	{
		if(temp.saldo<prueba.money)
		{
			printf("Saldo insuficiente\n");
			fclose(cu);
			return;
		}
	}
}
fwrite(&prueba,sizeof(transacciones),1,tr);
fclose(tr);
tr=fopen(rutatr, "r+b");
fwrite(&x, sizeof(int),1, tr);
fclose(tr);
cu=fopen(rutacu, "r+b");
cuentas temp2;
fseek(cu, sizeof(int), 1);
while(fread(&temp2, sizeof(cuentas),1,cu)==1)
{
	if(temp2.idCu==idOri)
	{
		temp2.saldo+=-1*prueba.money;
		fseek(cu, -sizeof(cuentas), SEEK_CUR);
		fwrite(&temp2, sizeof(cuentas), 1, cu);
		fclose(cu);
	}
}
cu=fopen(rutacu, "r+b");
fseek(cu, sizeof(int), 1);
while(fread(&temp2, sizeof(cuentas),1,cu)==1)
```

```
	{
		if(temp2.idCu==idDest)
		{
			temp2.saldo+=prueba.money;
			fseek(cu, -sizeof(cuentas), SEEK_CUR);
			fwrite(&temp2, sizeof(cuentas), 1, cu);
			fclose(cu);
			return;
		}
	}
}

void printTr(FILE *archivo, char ruta[200])
{
	int x;
	archivo=fopen(ruta, "rb");
	fread(&x, sizeof(int),1,archivo);
	transacciones prueba;
	printf("%-15s\t%-20s\t%-20s\t%-26s\t%-15s\t%-
15s\n","ID_Transaccion","Tipo Transaccion","ID_Cuenta_Origen",
"ID_Cuenta_Destino", "Fecha","Monto");
	while(fread(&prueba,sizeof(transacciones),1,archivo)==1)
	{
		printf("%-17d\t%-20s\t%-20d\t%-18d\t%02d/%02d/%04d\t%-17d\n",
prueba.idT, prueba.tipoT,prueba.idCueOri,prueba.idCueDes,
prueba.dateT.dd,prueba.dateT.mm, prueba.dateT.aa, prueba.money);
	}
	fclose(archivo);
}
```

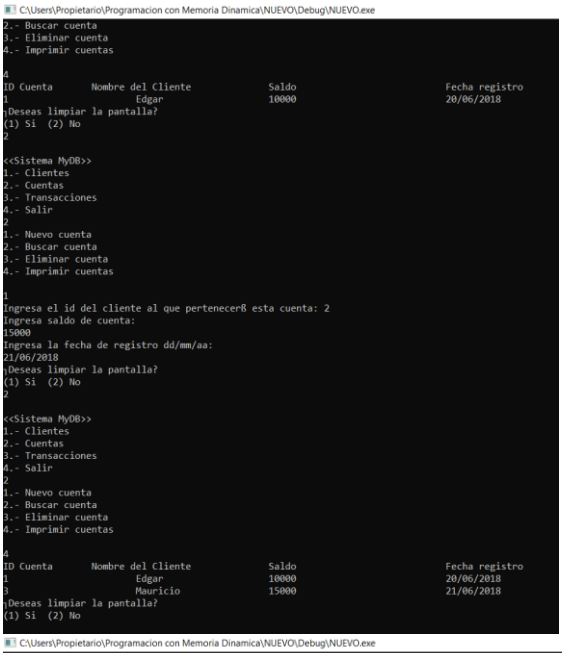
Ejecución

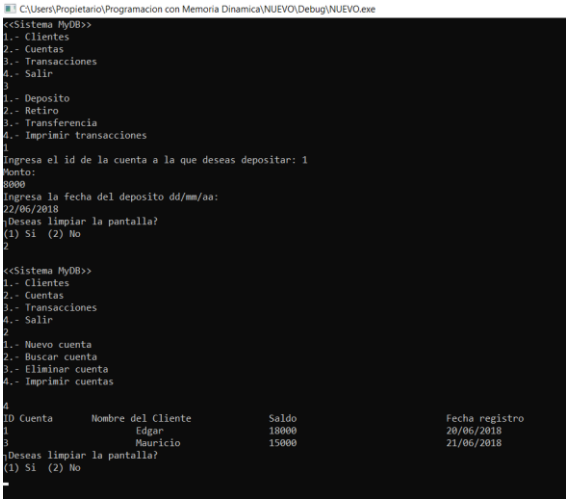


```
C:\Users\Propietario\Programacion con Memoria Dinamica\NUEVO\Debug\NUEVO.exe
¿Deseas limpiar la pantalla?
(1) Si  (2) No
2

<<Sistema MyDB>>
1.- Clientes
2.- Cuentas
3.- Transacciones
4.- Salir
1
1.- Nuevo cliente
2.- Buscar cliente
3.- Eliminar cliente
4.- Imprimir clientes
4
ID      Nombre                  Apellido Paterno        Apellido Materno        Fecha
1       Edgar                   Medina                  Rifas                   04/11/1998
2       Mauricio                Duran                   Padilla                 21/10/1998
¿Deseas limpiar la pantalla?
(1) Si  (2) No
```

```
C:\Users\Propietario\Programacion con Memoria Dinamica\NUEVO\Debug\NUEVO.exe
3.- Transacciones
4.- Salir
2
1.- Nuevo cuenta
2.- Buscar cuenta
3.- Eliminar cuenta
4.- Imprimir cuentas

1
Ingresa el id del cliente al que pertenecerß esta cuenta: 1
Ingresa saldo de cuenta:
10000
Ingresa la fecha de registro dd/mm/aa:
20/06/2018
¿Deseas limpiar la pantalla?
(1) Si  (2) No
2

<<Sistema MyDB>>
1.- Clientes
2.- Cuentas
3.- Transacciones
4.- Salir
2
1.- Nuevo cuenta
2.- Buscar cuenta
3.- Eliminar cuenta
4.- Imprimir cuentas

1
Ingresa el id del cliente al que pertenecerß esta cuenta: 25000
Este ID no existe.
¿Deseas limpiar la pantalla?
(1) Si  (2) No
2

<<Sistema MyDB>>
1.- Clientes
2.- Cuentas
3.- Transacciones
4.- Salir
2
1.- Nuevo cuenta
2.- Buscar cuenta
3.- Eliminar cuenta
4.- Imprimir cuentas

4
ID Cuenta       Nombre del Cliente              Saldo                   Fecha registro
1                       Edgar                   10000                   20/06/2018
```

```
C:\Users\Propietario\Programacion con Memoria Dinamica\NUEVO\Debug\NUEVO.exe
<<Sistema MyDB>>
1.- Clientes
2.- Cuentas
3.- Transacciones
4.- Salir
2
1.- Nuevo cuenta
2.- Buscar cuenta
3.- Eliminar cuenta
4.- Imprimir cuentas

3
Ingresa el id de la cuenta que desea borrar: 2
¿Deseas limpiar la pantalla?
(1) Si  (2) No
2

<<Sistema MyDB>>
1.- Clientes
2.- Cuentas
3.- Transacciones
4.- Salir
2
1.- Nuevo cuenta
2.- Buscar cuenta
3.- Eliminar cuenta
4.- Imprimir cuentas

4
ID Cuenta       Nombre del Cliente              Saldo                   Fecha registro
1                       Edgar                   10000                   20/06/2018
¿Deseas limpiar la pantalla?
(1) Si  (2) No
```


C:\Users\Propietario\Programacion con Memoria Dinamica\NUEVO\Debug\NUEVO.exe
2.- Buscar cuenta
3.- Eliminar cuenta
4.- Imprimir cuentas
4
ID Cuenta Nombre del Cliente Saldo Fecha registro
1 Edgar 10000 20/06/2018
¿Deseas limpiar la pantalla?
(1) Si (2) No
2
<<Sistema MyDB>>
1.- Clientes
2.- Cuentas
3.- Transacciones
4.- Salir
2
1.- Nuevo cuenta
2.- Buscar cuenta
3.- Eliminar cuenta
4.- Imprimir cuentas
1
Ingresa el id del cliente al que pertenecerß esta cuenta: 2
Ingresa saldo de cuenta:
15000
Ingresa la fecha de registro dd/mm/aa:
21/06/2018
¿Deseas limpiar la pantalla?
(1) Si (2) No
2
<<Sistema MyDB>>
1.- Clientes
2.- Cuentas
3.- Transacciones
4.- Salir
2
1.- Nuevo cuenta
2.- Buscar cuenta
3.- Eliminar cuenta
4.- Imprimir cuentas
4
ID Cuenta Nombre del Cliente Saldo Fecha registro
1 Edgar 10000 20/06/2018
3 Mauricio 15000 21/06/2018
¿Deseas limpiar la pantalla?
(1) Si (2) No


C:\Users\Propietario\Programacion con Memoria Dinamica\NUEVO\Debug\NUEVO.exe
<<Sistema MyDB>>
1.- Clientes
2.- Cuentas
3.- Transacciones
4.- Salir
3
1.- Deposito
2.- Retiro
3.- Transferencia
4.- Imprimir transacciones
1
Ingresa el id de la cuenta a la que deseas depositar: 1
Monto:
8000
Ingresa la fecha del deposito dd/mm/aa:
22/06/2018
¿Deseas limpiar la pantalla?
(1) Si (2) No
2
<<Sistema MyDB>>
1.- Clientes
2.- Cuentas
3.- Transacciones
4.- Salir
2
1.- Nuevo cuenta
2.- Buscar cuenta
3.- Eliminar cuenta
4.- Imprimir cuentas
4
ID Cuenta Nombre del Cliente Saldo Fecha registro
1 Edgar 18000 20/06/2018
3 Mauricio 15000 21/06/2018
¿Deseas limpiar la pantalla?
(1) Si (2) No

C:\Users\Propietario\Programacion con Memoria Dinamica\NUEVO\Debug\NUEVO.exe

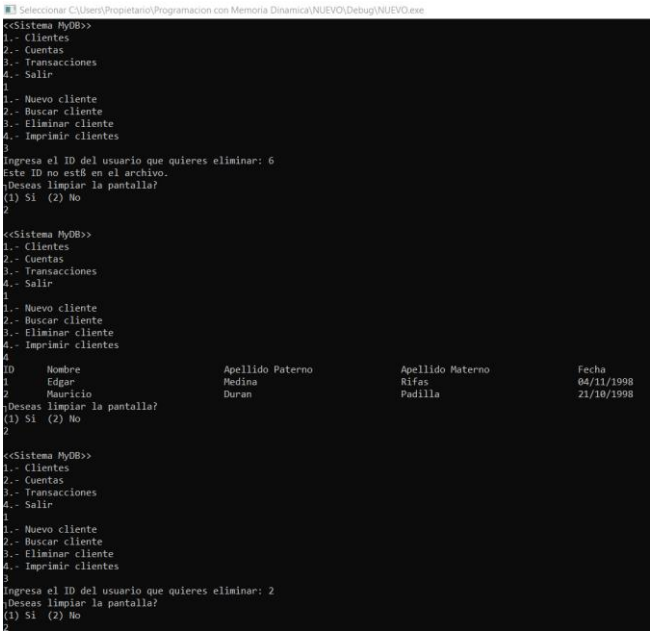
```
<<Sistema MyDB>>
1.- Clientes
2.- Cuentas
3.- Transacciones
4.- Salir
3
1.- Deposito
2.- Retiro
3.- Transferencia
4.- Imprimir transacciones
2
Ingresa el id de la cuenta de la que deseas retirar: 1
Monto:
5000
Ingresa la fecha del retiro dd/mm/aa:
22/06/2018
¿Deseas limpiar la pantalla?
(1) Si  (2) No
2

<<Sistema MyDB>>
1.- Clientes
2.- Cuentas
3.- Transacciones
4.- Salir
2
1.- Nuevo cuenta
2.- Buscar cuenta
3.- Eliminar cuenta
4.- Imprimir cuentas

4
ID Cuenta       Nombre del Cliente              Saldo                           Fecha registro
1                       Edgar                   13000                           20/06/2018
3                       Mauricio                10000                           21/06/2018
¿Deseas limpiar la pantalla?
(1) Si  (2) No
2

<<Sistema MyDB>>
1.- Clientes
2.- Cuentas
3.- Transacciones
4.- Salir
```

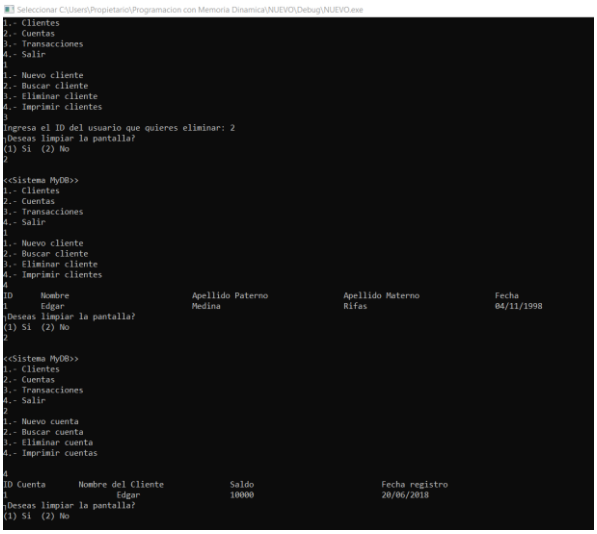
Seleccionar C:\Users\Propietario\Programacion con Memoria Dinamica\NUEVO\Debug\NUEVO.exe

```
<<Sistema MyDB>>
1.- Clientes
2.- Cuentas
3.- Transacciones
4.- Salir
3
1.- Deposito
2.- Retiro
3.- Transferencia
4.- Imprimir transacciones
2
Ingresa el id de la cuenta de la que deseas retirar: 3
Monto:
7000
Ingresa la fecha del retiro dd/mm/aa:
18/06/2018
¿Deseas limpiar la pantalla?
(1) Si  (2) No
2

<<Sistema MyDB>>
1.- Clientes
2.- Cuentas
3.- Transacciones
4.- Salir
2
1.- Nuevo cuenta
2.- Buscar cuenta
3.- Eliminar cuenta
4.- Imprimir cuentas

4
ID Cuenta       Nombre del Cliente              Saldo                           Fecha registro
1                       Edgar                   13000                           20/06/2018
3                       Mauricio                3000                            21/06/2018
¿Deseas limpiar la pantalla?
(1) Si  (2) No
2

<<Sistema MyDB>>
1.- Clientes
2.- Cuentas
3.- Transacciones
4.- Salir
```

C:\Users\Propietario\Programacion con Memoria Dinamica\NUEVO\Debug\NUEVO.exe

```
<<Sistema MyDB>>
1.- Clientes
2.- Cuentas
3.- Transacciones
4.- Salir
3
1.- Deposito
2.- Retiro
3.- Transferencia
4.- Imprimir transacciones
3
-Transferencia-
Ingresa el id de la cuenta de la que deseas retirar: 1
Ingresa el id de la cuenta a la que deseas depositar: 3
Monto:
3000
Ingresa la fecha del retiro dd/mm/aa:
22/06/2018
¿Deseas limpiar la pantalla?
(1) Si  (2) No
2

<<Sistema MyDB>>
1.- Clientes
2.- Cuentas
3.- Transacciones
4.- Salir
2
1.- Nuevo cuenta
2.- Buscar cuenta
3.- Eliminar cuenta
4.- Imprimir cuentas

4
ID Cuenta       Nombre del Cliente              Saldo                           Fecha registro
1                       Edgar                   10000                           20/06/2018
3                       Mauricio                6000                            21/06/2018
¿Deseas limpiar la pantalla?
(1) Si  (2) No
```





Conclusiones (obligatorio):

- ✓ Lo que aprendí con esta práctica. Lo que ya sabía.
  - El uso de la memoria dinámica lo mejoré.
  - Aprendí a manejar archivos, como mover el cursor y saber cual modo de uso de archivo es el que mas me conviene utilizar.
- ✓ Lo que me costó trabajo y cómo lo solucioné.
  - Me costó trabajo el manejo de archivos ya que lo que había visto era básico. Pero comencé a leer las presentaciones de la clase con más profundidad y también a investigar en foros en internet y logré resolver mis problemas.
- ✓ Lo que no pude solucionar.