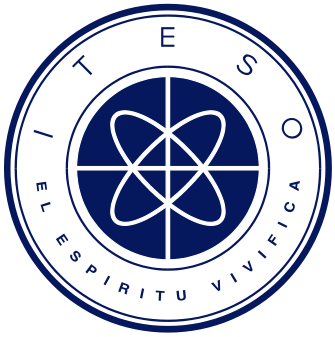
# INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

---

Departamento de Electrónica, Sistemas e Informática

INGENIERÍA EN SISTEMAS COMPUTACIONALES




# PROGRAMACIÓN CON MEMORIA DINÁMICA

## TAREA 2. MEMORIA DINÁMICA Y ARCHIVOS

Autor: Soto Pérez, Omar

Presentación: 10 pts.
Funcionalidad: 40 pts.
Pruebas: 20 pts.

12 de junio de 2018. Tlaquepaque, Jalisco,

- Falta describir las pruebas (escenario, y resultados de la experimentación).
- Falto cubrir los requerimientos funcionales sobre: transacciones.

## 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

```
<<Copie y pegue su código fuente aquí.>>
```

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

FILE* clientes;
FILE* cuentas;
FILE* transacciones;
FILE* MyDB;

typedef struct
{
	int d,m,a;
}fecha;
typedef struct
{
	int id;
	char nom[20];
	char app[20];
	char apm[20];
	fecha f;
}usuario;
typedef struct
{
	int id_c;
	int id_u;
	float saldo;
	fecha apertura;
}cuenta;
typedef struct
{
	int id_tr;
	int operacion;

}tranferencia;

char cl[50]="\\clientes.dat";
char cu[50]="\\cuentas.dat";
char tr[50]="\\transacciones.dat";

void CreateUser(char ruta[],FILE* archivo);
void PrintUsers(char ruta[],FILE*archivo);
void SearchUser(int ID,char ruta[],FILE*archivo);
void DeleteUser(int ID,char ruta[],FILE*archivo);

void CreateAccount(int ID,char ruta[],FILE*archivo);
void PrintAccount(char ruta[],FILE*archivo,FILE* archivo2,char ruta2[]);
```

```
void SearchAccount(int ID,char ruta[],FILE*archivo,FILE* archivo2,char ruta2[]);
void DeleteAccount(int ID,char ruta[],FILE*archivo);

typedef enum{true,false}bool;

int main()
{
	setvbuf(stderr, NULL, _IONBF, 0);
	setvbuf(stdout, NULL, _IONBF, 0);
	int mp,ms;
	char ruta_alternativa[100];
	char ruta_pri[100];
	char ruta_cli[100];
	char ruta_cue[100];
	char ruta_tra[100];
	MyDB=fopen("mydb.sys","r");
	int busqueda;

	if(MyDB==NULL){
		printf("<<<Esta es la primera vez de ejecucion>>>\n");
		printf("Indica donde queries guardar los archivos\n");
		scanf("%s",ruta_pri);
		strcpy(ruta_cli, ruta_pri);
		strcat(ruta_cli, cl);
		strcpy(ruta_cue, ruta_pri);
		strcat(ruta_cue, cu);
		strcpy(ruta_tra, ruta_pri);
		strcat(ruta_tra, tr);

		MyDB=fopen("mydb.sys","w+");
		fprintf(MyDB,"%s",ruta_pri);
		fclose(MyDB);

		clientes=fopen(ruta_cli,"a+b");
		if(ruta_cli==NULL)
			printf("Error al abrir el archivo: clientes.dat");
		cuentas=fopen(ruta_cue,"a+b");
		if(ruta_cue==NULL)
			printf("Error al abrir el archivo: cuentas.dat");
		transacciones=fopen(ruta_tra,"a+b");
		if(ruta_tra==NULL)
			printf("Error al abrir el archivo: transaccioness.dat");

		int a=0;
		fwrite(&a,sizeof(int),1,clientes);
		fwrite(&a,sizeof(int),1,cuentas);
		fwrite(&a,sizeof(int),1,transacciones);

		fclose(clientes);
		fclose(cuentas);
		fclose(transacciones);
	}
	else
	{
		fscanf(MyDB,"%s",ruta_alternativa);
```

```
            strcpy(ruta_cli, ruta_alternativa);
            strcat(ruta_cli, cl);
            strcpy(ruta_cue, ruta_alternativa);
            strcat(ruta_cue, cu);
            strcpy(ruta_tra, ruta_alternativa);
            strcat(ruta_tra, tr);
            fclose(MyDB);
      }
      printf("<<[1]Clientes>> <<[2]Cuentas>> <<[3]Transacciones>>
<<[4]Salir>>\n");
      scanf("%d",&mp);
      do
      {
            switch(mp)
            {
                  case 1:
                        printf("<<[1]Nuevo>> <<[2]Buscar>> <<[3]Eliminar>>
<<[4]Imprimir>>\n");
                        scanf("%d",&ms);
                        switch(ms)
                        {
                              case 1:
                                    clientes=fopen(ruta_cli,"a+b");
                                    CreateUser(ruta_cli,clientes);
                                    fclose(clientes);
                                    break;
                              case 2:
                                    printf("Ingrese el ID del usuario que
desea buscar: ");
                                    scanf("%d",&busqueda);
                                    clientes=fopen(ruta_cli,"a+b");
                                    SearchUser(busqueda,ruta_cli,clientes);
                                    fclose(clientes);
                                    break;
                              case 3:
                                    printf("Ingrese el ID del usuario que
desea eliminar: ");
                                    scanf("%d",&busqueda);
                                    clientes=fopen(ruta_cli,"a+b");
                                    DeleteUser(busqueda,ruta_cli,clientes);
                                    fclose(clientes);
                                    break;
                              case 4:
                                    clientes=fopen(ruta_cli,"a+b");
                                    PrintUsers(ruta_cli,clientes);
                                    fclose(clientes);
                                    break;
                        }
                  break;
                  case 2:
                        printf("<<[1]Nuevo>> <<[2]Buscar>> <<[3]Eliminar>>
<<[4]Imprimir>>\n");
                        scanf("%d",&ms);
                        switch(ms)
                        {
```

```
                                        case 1:
                                              printf("Indique el ID de usuario: ");
                                              scanf("%d",&busqueda);
                                              cuentas=fopen(ruta_cue,"a+b");
                                              CreateAccount(busqueda,ruta_cue,cuentas);
                                              fclose(cuentas);
                                              break;
                                        case 2:
                                              printf("Indique el ID de usuario: ");
                                              scanf("%d",&busqueda);
                                              cuentas=fopen(ruta_cue,"a+b");

      SearchAccount(busqueda,ruta_cue,cuentas,clientes,ruta_cli);
                                              fclose(cuentas);
                                              break;
                                        case 3:
                                              printf("Ingrese el ID de la cuenta que
desea eliminar: ");
                                              scanf("%d",&busqueda);
                                              clientes=fopen(ruta_cli,"a+b");

      DeleteAccount(busqueda,ruta_cli,clientes);
                                              fclose(clientes);
                                              break;
                                        case 4:
                                              cuentas=fopen(ruta_cue,"a+b");

      PrintAccount(ruta_cue,cuentas,clientes,ruta_cli);
                                              fclose(cuentas);
                                              break;
                                  }
                        break;
                        case 3:
                              printf("<<[1]Deposito>> <<[2]Retiro>>
<<[3]Transacción>>\n");
                        break;
                        case 4: return 0;
                        break;
                  }
                  printf("<<[1]Clientes>> <<[2]Cuentas>> <<[3]Transacciones>>
<<[4]Salir>>\n");
                  scanf("%d",&mp);

      }while(mp>=1 && mp<=4);

      system("pause");
      return 0;
}
void CreateUser(char ruta[],FILE* archivo)
{
      usuario user_temp;
      int id;
      archivo=fopen(ruta,"rb");
      if(archivo==NULL)
            printf("Error al abrir el archivo");
```

```
	fread(&id,sizeof(int),1,archivo);
	fclose(archivo);
	id++;
	archivo=fopen(ruta,"a+b");
	user_temp.id=id;
	printf("Nombre: ");
	scanf("%s",user_temp.nom);

	printf("Apellido Paterno: ");
	scanf("%s",user_temp.app);

	printf("Apellido Materno: ");
	scanf("%s",user_temp.apm);

	printf("Fecha:\n");
	printf(">>Dia: ");
	scanf("%i",&user_temp.f.d);
	printf(">>Mes: ");
	scanf("%i",&user_temp.f.m);
	printf(">>Año: ");
	scanf("%i",&user_temp.f.a);

	fwrite(&user_temp,sizeof(usuario),1,archivo);
	fclose(archivo);
	archivo=fopen(ruta,"r+b");
	fwrite(&id,sizeof(int),1,archivo);
	fclose(archivo);
}
void PrintUsers(char ruta[],FILE*archivo)
{
	int id;
	archivo=fopen(ruta,"rb");
	printf("ID\t\tNOMBRE\t\tAPELLIDO P.\tAPELLIDO M.\tFECHA\n");
	fread(&id,sizeof(int),1,archivo);
	usuario user_temp;
	while(fread(&user_temp,sizeof(usuario),1,archivo)==1)

	printf("%d\t\t%s\t\t%s\t\t%s\t\t%d/%d/%d\n",user_temp.id,user_temp.nom,us
er_temp.app,user_temp.apm,user_temp.f.d,user_temp.f.m,user_temp.f.a);
}
void SearchUser(int ID,char ruta[],FILE*archivo)
{
	int id;
	archivo=fopen(ruta,"rb");
	fread(&id,sizeof(int),1,archivo);
	usuario user_temp;
	while(fread(&user_temp,sizeof(usuario),1,archivo)==1)
	{
		if(user_temp.id==ID)
		{
			printf("ID\t\tNOMBRE\t\tAPELLIDO P.\tAPELLIDO M.\tFECHA\n");

	printf("%d\t\t%s\t\t%s\t\t%s\t\t%d/%d/%d\n",user_temp.id,user_temp.nom,us
er_temp.app,user_temp.apm,user_temp.f.d,user_temp.f.m,user_temp.f.a);
		}
```

```
	}
}
void DeleteUser(int ID,char ruta[],FILE*archivo)
{
	int a, cont=0, true=0;
	archivo=fopen(ruta, "r+b");
	fread(&a, sizeof(int),1,archivo);
	usuario user_temp;
	usuario *temp=(usuario *)malloc(sizeof(usuario));
	while(fread(&user_temp,sizeof(usuario),1,archivo)==1)
	{
		if(ID!=user_temp.id)
		{
			(temp+cont)->id=user_temp.id;
			strcpy((temp+cont)->nom,user_temp.nom);
			strcpy((temp+cont)->app,user_temp.app);
			strcpy((temp+cont)->apm,user_temp.apm);
			(temp+cont)->f=user_temp.f;
			cont++;
			temp=(usuario*)realloc(temp, sizeof(usuario)*(cont+1));
			true=1;
		}
	}
	if(true==1)
		printf("Este ID no está en el archivo.\n");
	fclose(archivo);
	archivo=fopen(ruta, "wb");
	fwrite(&a, sizeof(int),1,archivo);
	fclose(archivo);
	archivo=fopen(ruta,"a+b");
	fwrite(temp,sizeof(usuario),cont--,archivo);
	fclose(archivo);
	free(temp);
}
//void idClientes(int id,char ruta[],FILE* archivo,FILE* archivo2,char ruta2[])
//{
//	usuario temp;
//	cuenta aux;
//	while(fread(&aux,sizeof(cuenta),1,archivo)==1)
//		while(fread(&temp,sizeof(usuario),1,archivo2)==1){
//			if(temp.id==aux.id_u)
//				printf("%s",temp.nom);
//		}
//}
void CreateAccount(int ID,char ruta[],FILE*archivo)
{
	cuenta account_temp;
	int id;
	archivo=fopen(ruta,"rb");
	if(archivo==NULL)
		printf("Error al abrir el archivo");
	fread(&id,sizeof(int),1,archivo);
	fclose(archivo);
	id++;
	archivo=fopen(ruta,"a+b");
```

```
	account_temp.id_c=id;
	account_temp.id_u=ID;
	printf("Saldo de apertura: ");
	scanf("%f",&account_temp.saldo);
	printf("Fecha:\n");
	printf(">>Dia: ");
	scanf("%i",&account_temp.apertura.d);
	printf(">>Mes: ");
	scanf("%i",&account_temp.apertura.m);
	printf(">>Año: ");
	scanf("%i",&account_temp.apertura.a);

	fwrite(&account_temp,sizeof(cuenta),1,archivo);
	fclose(archivo);
	archivo=fopen(ruta,"r+b");
	fwrite(&id,sizeof(int),1,archivo);
	fclose(archivo);
}
void PrintAccount(char ruta[],FILE*archivo,FILE* archivo2,char ruta2[])
{
	int id;
	archivo=fopen(ruta,"rb");
	archivo2=fopen(ruta2,"rb");
	fread(&id,sizeof(int),1,archivo);
	cuenta account_temp;
	usuario usu;
	printf("USUARIO\t\tID CUENTA\tSALDO\t\t\tFECHA\n");
	while(fread(&account_temp,sizeof(cuenta),1,archivo)==1)
	{
		//idClientes(account_temp.id_u,ruta,archivo,archivo2,ruta2);
		fseek(archivo2,sizeof(int),SEEK_SET);
		while(fread(&usu,sizeof(usuario), 1,archivo2)==1)
			if(usu.id==account_temp.id_u)
				printf("%s",usu.nom);

	printf("\t\t%d\t\t%.2f\t\t\t%d/%d/%d\n",account_temp.id_c,account_temp.sa
ldo,account_temp.apertura.d,account_temp.apertura.m,account_temp.apertura.a);
	}
	fclose(archivo2);
	fclose(archivo);

}
void SearchAccount(int ID,char ruta[],FILE*archivo,FILE* archivo2,char ruta2[])
{
	int id;
	archivo=fopen(ruta,"rb");
	archivo2=fopen(ruta2,"rb");
	fread(&id,sizeof(int),1,archivo);
	cuenta account_temp;
	usuario user_temp;
	printf("USUARIO\t\tID CUENTA\tSALDO\t\tFECHA\n");
	while(fread(&account_temp,sizeof(cuenta),1,archivo)==1)
	{
		if(account_temp.id_u==ID){
			fseek(archivo2,sizeof(int),SEEK_SET);
```

```
                        while(fread(&user_temp,sizeof(usuario),1,archivo2))
                                if(user_temp.id==account_temp.id_u)
                                        printf("%s",user_temp.nom);

        printf("\t\t%d\t\t%.2f\t\t%d/%d/%d\n",account_temp.id_c,account_temp.sald
o,account_temp.apertura.d,account_temp.apertura.m,account_temp.apertura.a);
                }

        }
        fclose(archivo);
        fclose(archivo2);
}
void DeleteAccount(int ID,char ruta[],FILE*archivo)
{
        int a, cont=0, true=0;
        archivo=fopen(ruta, "r+b");
        fread(&a, sizeof(int),1,archivo);
        cuenta account_temp;
        cuenta *temp=(cuenta*)malloc(sizeof(cuenta));
        while(fread(&account_temp,sizeof(cuenta),1,archivo)==1)
        {
                if(ID!=account_temp.id_u)
                {
                        (temp+cont)->id_u=account_temp.id_u;
                        (temp+cont)->id_c=account_temp.id_c;
                        (temp+cont)->saldo=account_temp.saldo;
                        (temp+cont)->apertura=account_temp.apertura;
                        cont++;
                        temp=(cuenta*)realloc(temp, sizeof(cuenta)*(cont+1));
                        true=1;
                }
        }
        if(true==1)
                printf("Este ID no está en el archivo.\n");
        fclose(archivo);
        archivo=fopen(ruta, "wb");
        fwrite(&a, sizeof(int),1,archivo);
        fclose(archivo);
        archivo=fopen(ruta,"a+b");
        fwrite(temp,sizeof(cuenta),cont--,archivo);
        fclose(archivo);
        free(temp);
}
```

## Ejecución

<<Inserte capturas de pantalla de una ejecución para todos los casos>>

```
<<<Esta es la primera vez de ejecucion>>>
Indica donde queries guardar los archivos
C:\Users\OMAR\Desktop\archivos_banco
<<[1]Clientes>> <<[2]Cuentas>> <<[3]Transacciones>> <<[4]Salir>>
1
<<[1]Nuevo>> <<[2]Buscar>> <<[3]Eliminar>> <<[4]Imprimir>>
1
Nombre: Omar
Apellido Paterno: Soto
Apellido Materno: Perez
Fecha:
>>Dia: 1
>>Mes: 6
>>Ato: 1999
<<[1]Clientes>> <<[2]Cuentas>> <<[3]Transacciones>> <<[4]Salir>>
1
<<[1]Nuevo>> <<[2]Buscar>> <<[3]Eliminar>> <<[4]Imprimir>>
1
Nombre: Edgar
Apellido Paterno: Medina
Apellido Materno: Rifas
Fecha:
>>Dia: 22
>>Mes: 5
>>Ato: 1998
<<[1]Clientes>> <<[2]Cuentas>> <<[3]Transacciones>> <<[4]Salir>>
1
<<[1]Nuevo>> <<[2]Buscar>> <<[3]Eliminar>> <<[4]Imprimir>>
4
ID              NOMBRE          APELLIDO P.     APELLIDO M.     FECHA
1               Omar            Soto            Perez           1/6/1999
2               Edgar           Medina          Rifas           22/5/1998
<<[1]Clientes>> <<[2]Cuentas>> <<[3]Transacciones>> <<[4]Salir>>
1
<<[1]Nuevo>> <<[2]Buscar>> <<[3]Eliminar>> <<[4]Imprimir>>
2
Ingrese el ID del usuario que desea buscar: 2
ID              NOMBRE          APELLIDO P.     APELLIDO M.     FECHA
2               Edgar           Medina          Rifas           22/5/1998
<<[1]Clientes>> <<[2]Cuentas>> <<[3]Transacciones>> <<[4]Salir>>
1
<<[1]Nuevo>> <<[2]Buscar>> <<[3]Eliminar>> <<[4]Imprimir>>
3
Ingrese el ID del usuario que desea eliminar: 1
Este ID no estß en el archivo.
<<[1]Clientes>> <<[2]Cuentas>> <<[3]Transacciones>> <<[4]Salir>>
1
<<[1]Nuevo>> <<[2]Buscar>> <<[3]Eliminar>> <<[4]Imprimir>>
4
ID              NOMBRE          APELLIDO P.     APELLIDO M.     FECHA
2               Edgar           Medina          Rifas           22/5/1998
<<[1]Clientes>> <<[2]Cuentas>> <<[3]Transacciones>> <<[4]Salir>>
```

```
<<[1]Clientes>> <<[2]Cuentas>> <<[3]Transacciones>> <<[4]Salir>>
2
<<[1]Nuevo>> <<[2]Buscar>> <<[3]Eliminar>> <<[4]Imprimir>>
1
Indique el ID de usuario: 2
Saldo de apertura: 1500
Fecha:
>>Dia: 22
>>Mes: 6
>>Ato: 2018
<<[1]Clientes>> <<[2]Cuentas>> <<[3]Transacciones>> <<[4]Salir>>
2
<<[1]Nuevo>> <<[2]Buscar>> <<[3]Eliminar>> <<[4]Imprimir>>
4
USUARIO         ID CUENTA       SALDO                   FECHA
Edgar           1               1500.00                 22/6/2018
<<[1]Clientes>> <<[2]Cuentas>> <<[3]Transacciones>> <<[4]Salir>>
2
<<[1]Nuevo>> <<[2]Buscar>> <<[3]Eliminar>> <<[4]Imprimir>>
2
Indique el ID de usuario: 2
USUARIO         ID CUENTA       SALDO           FECHA
Edgar           1               1500.00         22/6/2018
<<[1]Clientes>> <<[2]Cuentas>> <<[3]Transacciones>> <<[4]Salir>>
```

Conclusiones (obligatorio):

Esta practica me hizo darme cuenta de que aun me hace mucha falta practicar el uso de manejo de archivos, debo decir que reforcé mis escasos conocimientos del tema como la apertura, cerrar, lectura y escritura.

En si, me costo trabajo entender el funcionamiento de los parámetros de cada función, lo cual lo logre entender con ayuda de las diapositivas compañeros y asesorías. Realmente no logre solucionar toda la problemática que se planteaba en la tarea, pues no solucione el ultimo menú, el de transferencia.