

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: Franco García Fernando

Tlaquepaque, Jalisco, 12 de junio de 2018.

Instrucciones para entrega de tarea

Esta tarea, como el resto, es **IMPRESINDIBLE** 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 5 pts
- 60% para la funcionalidad 40 pts
- 30% para las pruebas 20 pts

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 autonómico. 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 autonómico. 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 transferencias, 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

```
/*
 * tarea3.c
 *
 * Created on: 13 jun. 2018
 * Author: Fernando Franco
 */
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<conio.h>

typedef struct {
    int dia;
    int mes;
    int anio;
} fecha;

typedef struct {
    int id;
    char nombre[50];
    char paterno[50];
    char materno[50];
    fecha ff;
} clientes;
typedef struct {
    int id_cuenta;
    int saldo;
    clientes cli;
    fecha fecha_ape;
} cuenta;
typedef struct {
    int id_trans;
    int cuenta_or;
    int cuenta_des;
    int cantidad;
    char tipo_mov[20];
    fecha ff;
} transac;

char dir_cli[100], dir_cuen[100], dir_trans[100], dir_aux[100];
char full_rut[100], full_rut2[100];
```

```

void crearCliente(FILE *client);
void imprimirCliente(FILE *f);
void Buscar_cliente(FILE *f);
void eliminarCliente(FILE *f);

void crearCuenta(FILE *ff, FILE *f);
void imprimirCuenta(FILE *ff, FILE *f);
void buscarCuenta(FILE *ff, FILE *f);
void eliminarCuenta(FILE *ff);

void deposito FILE *d, FILE *ff);
void retiro FILE *d, FILE *ff);
void imprimirTransfe(FILE *d, FILE *ff);

int main(){
    setvbuf(stderr, NULL, _IONBF, 0);
    setvbuf(stdout, NULL, _IONBF, 0);
    int x, y;

    FILE *clientes;
    FILE *cuentas;
    FILE *transacciones;
    FILE *MYDB;
    MYDB=fopen("mydb.sys", "r");
    if(MYDB==NULL){
        printf("¿En donde quieres guardar tus archivos?
(ejemplo: C:\\Users\\docs\\...)\\n");
        scanf("%[^\\n]", full_rut);
        strcpy(dir_cli, full_rut);
        strcat(dir_cli, "\\clientes.dat");
        strcpy(dir_cuen, full_rut);
        strcat(dir_cuen, "\\cuentas.dat");
        strcpy(dir_trans, full_rut);
        strcat(dir_trans, "\\transferencias.dat");
        strcpy(dir_aux, full_rut);
        strcat(dir_aux, "\\auxiliar_cli.dat");

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

        clientes=fopen(dir_cli, "a+b");
        if(clientes==NULL)
            printf("Error en el archivo\\n");
        cuentas=fopen(dir_cuen, "a+b");
    }
}

```



```

        if (cuentas==NULL)
            printf("Error en el archivo\n");
        transacciones=fopen(dir_trans,"a+b");
        if (transacciones==NULL)
            printf("Error en el archivo\n");

        int prim=0;
        fwrite(&prim, sizeof(int), 1, clientes);
        fwrite(&prim, sizeof(int), 1, cuentas);
        fwrite(&prim, sizeof(int), 1, transacciones);
        fclose(clientes);
        fclose(cuentas);
        fclose(transacciones);
    }
else{

    fscanf(MYDB, "%[^\n]",full_rut2);
    strcpy(dir_cli, full_rut2);
    strcat(dir_cli,"\\clientes.dat");
    strcpy(dir_cuen,full_rut2);
    strcat(dir_cuen,"\\cuentas.dat");
    strcpy(dir_trans,full_rut2);
    strcat(dir_trans,"\\transferencias.dat");
    fclose(MYDB);
}

printf("<<Sistema MyDB>>\n");
do{
    printf("1. Clientes\n2. Cuentas\n3. Transacciones\n4. Salir\n");
    scanf("%d",&x);
    while (x<1 || x>4){
        printf("Opción no valida\nIngresa nuevamente a la opción que gustes\n");
        scanf("%d", &x);
    }
    fflush(stdin);
    switch (x){
        case 1:
            printf("1. Nuevo Cliente\n2. Buscar Cliente\n3. Eliminar Cliente\n4. Imprimir Clientes\n");
            scanf("%d", &y);
            while (x<1 || x>4){
                printf("Opción no valida\nIngresa nuevamente a la opción que gustes\n");
                scanf("%d", &x);
            }

```

```

    }
    switch(y){
        case 1:
            clientes=fopen(dir_cli, "a+b");
            crearCliente(clientes);
            fclose(clientes);
            break;
        case 2:
            clientes=fopen(dir_cli, "a+b");
            Buscar_cliente(clientes);
            fclose(clientes);
            break;
        case 3:
            clientes=fopen(dir_cli, "a+b");
            eliminarCliente(clientes);
            fclose(clientes);
            break;
        case 4:
            clientes=fopen(dir_cli, "a+b");
            imprimirCliente(clientes);
            fclose(clientes);
            break;
    }
    break;

    case 2:
        printf("1. Nueva Cuenta\n2. Buscar Cuenta\n3. Eliminar\n4. Imprimir Cuentas\n");
        scanf("%d", &y);
        while (x<1 || x>4){
            printf("Opción no valida\nIngresa nuevamente a la opción que gustes\n");
            scanf("%d", &x);
        }
        switch(y){
            case 1:
                cuentas=fopen(dir_cuen, "a+b");
                crearCuenta(cuentas, clientes);
                fclose(cuentas);
                break;
            case 2:
                cuentas=fopen(dir_cuen, "a+b");
                buscarCuenta(cuentas, clientes);
                fclose(cuentas);
                break;

```

```

    case 3:
        cuentas=fopen(dir_cuen,"a+b");
        eliminarCuenta(cuentas);
        fclose(cuentas);
        break;
    case 4:
        cuentas=fopen(dir_cuen, "a+b");
        imprimirCuenta(cuentas, clientes);
        fclose(cuentas);
        break;
}
break;

case 3:
    printf("1. Depósito\n2. Retiro\n3. Transferencia\n");
    scanf("%d", &y);
    while (x<1 || x>3){
        printf("Opción no valida\nIngresa
nuevamente a la opción que gustes\n");
        scanf("%d", &x);
    }
    switch(y){
    case 1:
        transacciones=fopen(dir_trans,"a+b");
        deposito(transacciones, cuentas);
        fclose(transacciones);
        break;
    case 2:
        transacciones=fopen(dir_trans,"a+b");
        retiro(transacciones, cuentas);
        fclose(transacciones);
        break;
    case 3:
        transacciones=fopen(dir_trans,"a+b");
        imprimirTransfe(transacciones, cuentas);
        fclose(transacciones);

        break;
    }
    break;

case 4:
    return 0;

default:
    return 0;

```

```

    } while (x>1 || x<4);
    return 0;
}

void crearCliente(FILE *client){
    clientes cli;
    int quant;

    client=fopen(dir_cli, "rb");
    fread(&quant, sizeof(int), 1, client);
    fclose(client);
    abs(quant++);

    client=fopen(dir_cli, "a+b");
    cli.id=quant;
    fflush(stdin);
    printf("Nombre(s): ");
    scanf("%[^\n]", cli.nombre);
    printf("Apellido Paterno: ");
    scanf("%s", cli.paterno);
    printf("Apellido Materno: ");
    scanf("%s", cli.materno);
    fflush(stdin);
    printf("Ingresa tu fecha de nacimiento(dd/mm/aaa)\n");
    scanf("%d ", &cli.ff.dia);
    scanf("%d ", &cli.ff.mes);
    scanf("%d", &cli.ff.anio);

    fwrite(&cli, sizeof(clientes), 1, client);
    fclose(client);

    client=fopen(dir_cli, "r+b");
    fwrite(&quant, sizeof(int), 1, client);
    fclose(client);
}

void imprimirCliente(FILE *f){
    int x;
    f=fopen(dir_cli, "rb");
    fread(&x, sizeof(int), 1, f);
    clientes cli;
    printf("%-20s %-20s %-20s %-20s %-20s\n", "ID", "Nombre",
"Apellido Paterno", "Apellido Materno", "Fecha de nacimiento");
    while (fread(&cli, sizeof(clientes), 1, f)==1){
        printf("%-20d %-20s %-20s %-20s %02d/%02d/%d\n",
cli.id, cli.nombre, cli.paterno, cli.materno, cli.ff.dia, cli.ff.mes, cl
i.ff.anio);
    }
}

```

```

    }
    fclose(f);
}

void Buscar_cliente(FILE *f){
    int x, id;
    f=fopen(dir_cli,"rb");
    fread(&x, sizeof int, 1, f);
    clientes cli;
    printf("Ingresa el ID del usuario que quieres\n");
    scanf("%d",&id);
    while(fread(&cli,sizeof clientes, 1,f)==1){
        if(cli.id==id){
            printf("%-20s %-20s %-20s %-20s %-20s\n", "ID",
"Nombre", "Apellido Paterno", "Apellido Materno", "Fecha de
nacimiento");
            printf("%-20d %-20s %-20s %-20s %02d/%02d/%d\n",
cli.id,cli.nombre,cli.paterno,cli.materno,cli.ff.dia,cli.ff.mes,cli
i.ff.anio);
            fclose(f);
        }
    }
}

void eliminarCliente(FILE *f){
    int a, cont=0, true=0,ID;
    printf("Ingresa el ID del usuario al que quieras borrar\n");
    scanf("%d",&ID);
    f=fopen(dir_cli, "r+b");
    fread(&a, sizeof int,1,f);
    clientes cli;
    clientes *array=(clientes *)malloc(sizeof clientes);
    while(fread(&cli,sizeof clientes,1,f)==1)
    {
        if(ID!=cli.id)
        {
            (array+cont)->id=cli.id;
            strcpy((array+cont)->nombre,cli.nombre);
            strcpy((array+cont)->paterno,cli.paterno);
            strcpy((array+cont)->materno,cli.materno);
            (array+cont)->ff=cli.ff;
            cont++;
            array=(clientes*)realloc(array,
sizeof(clientes)*(cont+1));
            true=1;
        }
    }
    if(true==1)

```

```

        printf("Este ID no está en el archivo.\n");
fclose(f);
f=fopen(dir_cli, "wb");
fwrite(&a, sizeof(int), 1, f);
fclose(f);
f=fopen(dir_cli, "a+b");
fwrite(array, sizeof(clientes), cont--, f);
fclose(f);
free(array);
}

void crearCuenta(FILE *ff, FILE *f){
    cuenta cuen;
    int id, id_cli;
    int ID;

    f=fopen(dir_cli, "rb");
    printf("Ingresa el ID al que quieres enlazar la
cuenta\n");
    scanf("%d", &id);

    ff=fopen(dir_cuen, "rb");
    fread(&ID, sizeof(int), 1, ff);
    fclose(ff);
    ID++;
    if(fread(&id_cli, sizeof(int), 1, f)==0){
        printf("Usuario no encontrado\nIntente de
nuevo\n");
        scanf("%d", &id);
    }
    ff=fopen(dir_cuen, "a+b");
    cuen.cli.id=id;
    cuen.id_cuenta=ID;
    fflush(stdin);
    printf("Ingresa el saldo inicial: \n");
    scanf("%d", &cuen.saldo);
    printf("Fecha de apertura de la cuenta(dd/mm/aaa)\n");
    scanf("%d ", &cuen.fecha_ape.dia);
    scanf("%d ", &cuen.fecha_ape.mes);
    scanf("%d", &cuen.fecha_ape.anio);

    fwrite(&cuen, sizeof(cuenta), 1, ff);
    fclose(f);
    fclose(ff);

    ff=fopen(dir_cuen, "r+b");
    fwrite(&ID, sizeof(int), 1, ff);

```

```

        fclose(ff);
    }
    void imprimirCuenta(FILE *ff, FILE *f){
        int x;
        int id_cli;
        ff=fopen(dir_cuen,"rb");
        f=fopen(dir_cli,"rb");
        fread(&x, sizeof int, 1, ff);
        fread(&id_cli, sizeof int, 1, f);
        cuenta cuen;
        clientes cli;
        printf("%-20s %-20s %-20s %-20s %-20s\n", "ID.Cuenta",
        "ID.Usuario", "Nombre del usuario", "Saldo", "Fecha de apertura");
        while(fread(&cuen, sizeof cuenta, 1, ff)==1){
            fread(&cli, sizeof clientes, 1, f);
            printf("%-20d %-20d %-20s $%-20d
            %02d/%02d/%d\n", cuen.id_cuenta, cli.id, cli.nombre, cuen.saldo,
            cuen.fecha_ape.dia, cuen.fecha_ape.mes, cuen.fecha_ape.anio);
        }
        fclose(f);
        fclose(ff);
    }
    void buscarCuenta(FILE *ff, FILE *f){
        int x, id, id_cli;
        ff=fopen(dir_cuen,"rb");
        f=fopen(dir_cli,"rb");
        fread(&x, sizeof int, 1, ff);
        fread(&id_cli, sizeof int, 1, f);
        cuenta cuen;
        clientes cli;
        printf("Ingresa el ID del usuario que quieres la cuenta\n");
        scanf("%d", &id);
        if(fread(&cuen, sizeof cuenta, 1, ff)==0)
            printf("Cuenta no encontrada\n");
        while(fread(&cuen, sizeof cuenta, 1, ff)==1){
            fread(&cli, sizeof clientes, 1, f);
            printf("%-20s %-20s %-20s %-20s %-
            20s\n", "ID.Cuenta", "ID.Usuario", "Nombre del usuario", "Saldo",
            "Fecha de apertura");
            printf("%-20d %-20d %-20s $%-20d
            %02d/%02d/%d\n", cuen.id_cuenta, cli.id, cli.nombre, cuen.saldo,
            cuen.fecha_ape.dia, cuen.fecha_ape.mes, cuen.fecha_ape.anio);
        }
        fclose(f);
        fclose(ff);
    }
}

```

```

void eliminarCuenta(FILE *ff){
    int a, cont=0, ID;
    printf("Ingresa el ID del usuario al que quieras borrar la
cuenta\n");
    scanf("%d",&ID);
    while(fread(&ID, sizeof(int), 1, ff) == 0){
        printf("Error en el ID de la cuenta a
depositar\nIntente de nuevo\n");
        scanf("%d",&ID);
    }

    ff=fopen(dir_cuen, "r+b");
    fread(&a, sizeof(int), 1, ff);
    cuenta cuen;
    cuenta *array=(cuenta *)malloc(sizeof(cuenta));

    while(fread(&cuén, sizeof(clientes), 1, ff) == 1)
    {
        if(ID != cuén.id_cuenta)
        {
            (array+cont)->id_cuenta=cuén.id_cuenta;
            (array+cont)->saldo=cuén.saldo;
            (array+cont)->fecha_ape=cuén.fecha_ape;
            cont++;
            array=(cuenta *)realloc(array,
sizeof(cuenta)*(cont+1));
        }
    }

    fclose(ff);
    ff=fopen(dir_cuen, "wb");
    fwrite(&a, sizeof(int), 1, ff);
    fclose(ff);
    ff=fopen(dir_cuen, "a+b");
    fwrite(array, sizeof(clientes), cont--, ff);
    fclose(ff);
    free array;
}

void deposito FILE *d, FILE *ff){
    int cantidad, id_cuentas, id_transacciones;
    transac deposito;
    cuenta mome;
    strcpy(deposito.tipo_mov, "Deposito");

    d=fopen(dir_trans, "rb");
    fread(&id_transacciones, sizeof(int), 1, ff);
    fclose(d);

```



```

        id_transacciones++;
        ff=fopen(dir_cuen, "rb");
        if (d==NULL)
            printf("Error en el archivo\n");
        printf("Ingresa el ID de la cuenta a la que vas a
depositar\n");
        scanf("%d", &id_cuentas);
        while(!fread(&id_cuentas, sizeof int), 1, ff)){
            printf("Error en el ID de la cuenta a depositar\nIntente
de nuevo\n");
            scanf("%d", &id_cuentas);
        }

        fread(&id_cuentas, sizeof int, 1, ff);
        printf("Ingresa la fecha de la operación(dd/mm/aaaa)\n");
        scanf("%d %d %d",
&deposito.ff.dia, &deposito.ff.mes, &deposito.ff.ano);
        printf("Ingresa la cantidad a depositar\n");
        scanf("%d", &cantidad);
        deposito.cantidad=cantidad;
        deposito.id_trans=id_transacciones;

        ff=fopen(dir_cuen, "r+b");
        fseek(ff, sizeof int, 1);
        while(fread(&mome, sizeof cuenta, 1, ff)==1){
            if(mome.id_cuenta==id_cuentas){
                mome.saldo= mome.saldo+cantidad;
                fseek(ff, -sizeof cuenta, SEEK_CUR);
                fwrite(&mome, sizeof cuenta, 1, ff);
            }
        }
        fclose(ff);
        fclose(ff);
        fclose(d);
    }

void retiro FILE *d, FILE *ff){
    int id, cantidad;
    printf("Ingresa el ID de la cuenta que haras el retiro");
    scanf("%d", &id);
    ff=fopen(dir_cuen, "rb");
    while(!fread(&id, sizeof int, 1, ff)){
        printf("Error en el ID de la cuenta a
depositar\nIntente de nuevo\n");
        scanf("%d", &id);
    }
    fclose(ff);

```

```

transac retiro;
sprcpy(retiro.tipo_mov, "Retiro");
printf("¿Cuánto deseas retirar?");
scanf("%d", &cantidad);

ff=fopen(dir_cuen, "rb");
cuenta temp;
fseek(ff, sizeof(int), SEEK_SET);
while(fread(&temp, sizeof(cuenta), 1, ff) == 1){
    if(temp.id_cuenta == id){
        if(temp.saldo >= cantidad){

        }
    }
}

}

void imprimirTransfe(FILE *d, FILE *ff){
    int x;
    int id_cli;

    d=fopen(dir_trans, "rb");
    ff=fopen(dir_cuen, "rb");
    fread(&id_cli, sizeof(int), 1, d);
    fread(&x, sizeof(int), 1, ff);
    transac tran_1;
    cuenta cuen;
    printf("%-20s %-20s %-20s %-20s %-20s %-20s\n", "ID.Transacción", "Tipo de transacción", "ID.Cuenta de origen", "ID.Cuenta de destino", "Fecha de la operación", "Cantidad de la operación");
    while(fread(&cuen, sizeof(cuenta), 1, ff) == 1){
        fread(&tran_1, sizeof(transac), 1, d);
        printf("%-20d %-20s %-20d %-20d %-20d/%02d/%02d/%d %-20d\n", tran_1.id_trans, tran_1.tipo_mov, tran_1.cuenta_or, tran_1.cuenta_des, tran_1.ff.dia, tran_1.ff.mes, tran_1.ff.anio, tran_1.cantidad);
    }
    fclose(ff);
    fclose(d);
}

```

Ejecución

The image displays two sequential screenshots of the Eclipse IDE running a C++ application named 'tarea_3.exe'. The IDE's title bar indicates the project is 'Programación con Memoria Dinámica - tarea_3/tarea3.c - Eclipse'. The console window shows the program's output and user input.

First Screenshot:

```
tarea_3.exe [C/C++ Application] C:\Users\Fernando Franco\Documents\Tercer Semestre\Programación con Memoria Dinámica\tarea_3\Debug\tarea_3.exe (22/6/18 04:35)
¿En donde quieres guardar tus archivos? (ejemplo: C:\Users\docs\...)
C:\Users\Fernando Franco\Desktop\tarea_3
```

Second Screenshot:

```
tarea_3.exe [C/C++ Application] C:\Users\Fernando Franco\Documents\Tercer Semestre\Programación con Memoria Dinámica\tarea_3\Debug\tarea_3.exe (22/6/18 04:35)
¿En donde quieres guardar tus archivos? (ejemplo: C:\Users\docs\...)
C:\Users\Fernando Franco\Desktop\tarea_3
<<Sistema MyDB>>
1. Clientes
2. Cuentas
3. Transacciones
4. Salir
```

Programación con Memoria Dinámica - tarea_3\tarea3.c - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Console

tarea_3.exe [C:\C++ Application] C:\Users\Fernando Franco\Documents\Tercer Semestre\Programación con Memoria Dinámica\tarea_3\Debug\tarea_3.exe (22/6/18 04:35)

¿En donde quieres guardar tus archivos? (ejemplo: C:\Users\docs\...)
C:\Users\Fernando Franco\Desktop\tarea_3

<<Sistema MyDB>>

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

1

1. Nuevo Cliente
2. Buscar Cliente
3. Eliminar Cliente
4. Imprimir Clientes

1

Nombre(s): Fernando
Apellido Paterno: Franco
Apellido Materno: García
Ingresa tu fecha de nacimiento(dd/mm/aaa)
07/07/1999

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

Programación con Memoria Dinámica - tarea_3\tarea3.c - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Console

tarea_3.exe [C:\C++ Application] C:\Users\Fernando Franco\Documents\Tercer Semestre\Programación con Memoria Dinámica\tarea_3\Debug\tarea_3.exe (22/6/18 04:36)

<<Sistema MyDB>>

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

1

1. Nuevo Cliente
2. Buscar Cliente
3. Eliminar Cliente
4. Imprimir Clientes

2

Ingresa el ID del usuario que quieres

1

ID	Nombre	Apellido Paterno	Apellido Materno	Fecha de nacimiento
1	Fernando	Franco	García	07/07/1999

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

Programación con Memoria Dinámica - tarea_3/tarea3.c - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Console

tarea_3.exe [C:/C++ Application] C:\Users\Fernando Franco\Documents\Tercer Semestre\Programación con Memoria Dinámica\tarea_3\Debug\tarea_3.exe (22/6/18 04:42)

```
<<Sistema MyDB>>
1. Clientes
2. Cuentas
3. Transacciones
4. Salir
1
1. Nuevo Cliente
2. Buscar Cliente
3. Eliminar Cliente
4. Imprimir Clientes
3
Ingresa el ID del usuario al que quieras borrar
1
Eliminado exitosamente
1. Clientes
2. Cuentas
3. Transacciones
4. Salir
```

Windows Search

Programación con Memoria Dinámica - tarea_3/tarea3.c - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Console

tarea_3.exe [C:/C++ Application] C:\Users\Fernando Franco\Documents\Tercer Semestre\Programación con Memoria Dinámica\tarea_3\Debug\tarea_3.exe (22/6/18 04:43)

```
<<Sistema MyDB>>
1. Clientes
2. Cuentas
3. Transacciones
4. Salir
2
1. Nueva Cuenta
2. Buscar Cuenta
3. Eliminar Cuenta
4. Imprimir Cuentas
1
Ingresa el ID al que quieres enlazar la cuenta
2
Ingresa el saldo inicial:
500
Fecha de apertura de la cuenta(dd/mm/aaa)
22 06 2018
1. Clientes
2. Cuentas
3. Transacciones
4. Salir
2
1. Nueva Cuenta
2. Buscar Cuenta
3. Eliminar Cuenta
4. Imprimir Cuentas
4
ID.Cuenta      ID.Usuario      Nombre del usuario      Saldo      Fecha de apertura
1              2               Arturo                  $500       22/06/2018
```

Windows Search

Programación con Memoria Dinámica - tarea_3\tarea3.c - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Console

tarea_3.exe [C:\C++ Application] C:\Users\Fernando Franco\Documents\Tercer Semestre\Programación con Memoria Dinámica\tarea_3\Debug\tarea_3.exe (22/6/18 05:10)

```
<<Sistema MyDB>>
1. Clientes
2. Cuentas
3. Transacciones
4. Salir
2
1. Nueva Cuenta
2. Buscar Cuenta
3. Eliminar Cuenta
4. Imprimir Cuentas
2
Ingresa el ID del usuario que quieres la cuenta
1
ID.Cuenta      ID.Usuario      Nombre del usuario      Saldo      Fecha de apertura
2              1              Omar                    $5000      22/06/2018
1. Clientes
2. Cuentas
3. Transacciones
4. Salir
```

Windows taskbar: Escribe aquí para buscar, taskbar icons, system tray: 05:10, 22/6/2018

Programación con Memoria Dinámica - tarea_3\tarea3.c - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Console

tarea_3.exe [C:\C++ Application] C:\Users\Fernando Franco\Documents\Tercer Semestre\Programación con Memoria Dinámica\tarea_3\Debug\tarea_3.exe (22/6/18 05:16)

```
<<Sistema MyDB>>
1. Clientes
2. Cuentas
3. Transacciones
4. Salir
2
1. Nueva Cuenta
2. Buscar Cuenta
3. Eliminar Cuenta
4. Imprimir Cuentas
4
ID.Cuenta      ID.Usuario      Nombre del usuario      Saldo      Fecha de apertura
1              1              Omar                    $500       22/06/2018
2              1              Omar                    $5000      22/06/2018
1. Clientes
2. Cuentas
3. Transacciones
4. Salir
```

Windows taskbar: Escribe aquí para buscar, taskbar icons, system tray: 05:10, 22/6/2018

Programación con Memoria Dinámica - tarea_3/tarea3.c - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

```

tarea_3.exe [C/C++ Application] C:\Users\Fernando Franco\Documents\Tercer Semestre\Programación con Memoria Dinámica\tarea_3\Debug\tarea_3.exe
<<Sistema MyDB>>
1. Clientes
2. Cuentas
3. Transacciones
4. Salir
2
1. Nueva Cuenta
2. Buscar Cuenta
3. Eliminar Cuenta
4. Imprimir Cuentas
3
Ingresa el ID del usuario al que quieras borrar la cuenta
2
1. Clientes
2. Cuentas
3. Transacciones
4. Salir
2
1. Nueva Cuenta
2. Buscar Cuenta
3. Eliminar Cuenta
4. Imprimir Cuentas
4
ID.Cuenta      ID.Usuario      Nombre del usuario  Saldo
1              1              Omar                $500
1. Clientes
2. Cuentas
3. Transacciones
4. Salir

```

Escribe aquí para buscar

ERRORES

Programación con Memoria Dinámica - tarea_3/tarea3.c - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

```

tarea_3.exe [C/C++ Application] C:\Users\Fernando Franco\Documents\Tercer Semestre\Programación con Memoria Dinámica\tarea_3\Debug\tarea_3.exe (22/6/18 05:20)
1. Clientes
2. Cuentas
3. Transacciones
4. Salir
3
1. Depósito
2. Retiro
3. Transferencia
1
Ingresa el ID de la cuenta a la que vas a depositar
1
Ingresa la fecha de la operación(dd/mm/aaaa)
22 06 2018
Ingresa la cantidad a depositar
200
1. Clientes
2. Cuentas
3. Transacciones
4. Salir
2
1. Nueva Cuenta
2. Buscar Cuenta
3. Eliminar Cuenta
4. Imprimir Cuentas
4
ID.Cuenta      ID.Usuario      Nombre del usuario  Saldo      Fecha de apertura
700            1              Omar                $1699946610  1970063904/2022618658/1
220398805     1              Omar                $-2          00/00/0
a              1              Omar                $a           aa/aa/a

```

Escribe aquí para buscar

05:21 22/6/2018

