UNIVERSIDAD TECNOLÓGICA DE SANTIAGO
UTESA, SISTEMA CORPORATIVO

TRABAJO DE LA ASIGNATURA DE:

Taller Programación I

TEMA:

Entrega Proyecto Final

MAESTRO/A:

Raul A. Toribio

GRUPO:

IET-425-001

SUSTENTANTES	CARRERA	MATRÍCULA
Ydarlyn Javier	Ingeniería Eléctrica	1-21-2290

SANTIAGO, REPÚBLICA DOMINICANA
20 DE DICIEMBRE DEL 2024

DESCRIPCION DEL PROBLEMA

Este proyecto aborda la necesidad de un sistema sencillo y eficiente para gestionar información financiera en pequeñas empresas. Actualmente, muchas organizaciones dependen de métodos manuales o herramientas básicas como hojas de cálculo, lo que genera errores, limita el control y compromete la seguridad de los datos. El sistema propuesto permite registrar clientes, realizar transacciones (depósitos y retiros), consultar saldos y garantizar la seguridad mediante una clave de acceso. Aunque no es una idea completamente original, adapta funcionalidades comunes en sistemas bancarios básicos a un formato simplificado y accesible, ideal para pequeños negocios o fines educativos. Sus limitaciones incluyen la capacidad de manejar hasta 100 clientes, la ausencia de una interfaz gráfica y la falta de integración con bases de datos avanzadas. Sin embargo, su diseño práctico lo hace adecuado para entornos de baja complejidad, ofreciendo una solución económica y funcional para la gestión financiera básica.

OBJETIVO DEL PROYECTO

El objetivo de este proyecto es diseñar un sistema computacional o cajero que resuelva la problemática de gestionar información financiera de forma eficiente en pequeñas empresas o negocios. Este sistema busca satisfacer la necesidad de registrar clientes, controlar depósitos y retiros, consultar saldos, y garantizar la seguridad de los datos mediante un acceso restringido, todo ello en un entorno simplificado y accesible. Este proyecto tiene como propósito mejorar mis habilidades en el diseño de sistemas computacionales, al tiempo que aporta una herramienta útil y eficiente que contribuye a la organización y control de la información financiera en escenarios reales.

RESUMEN TECNICO

El sistema desarrollado para la gestión financiera de clientes opera de manera estructurada y eficiente a través de varias etapas que garantizan el cumplimiento de sus objetivos. A continuación, se describe cómo funciona cada una de sus partes y cómo da solución a las necesidades operativas:

- **Inicio de Sesión (Login):** El sistema emplea un mecanismo de autenticación mediante una clave de acceso, almacenada en un archivo externo. Este proceso garantiza que solo usuarios autorizados puedan operar el sistema. En caso de olvido o necesidad, la clave puede ser modificada para mantener la seguridad.
- **Gestión de Clientes:** Permite registrar nuevos clientes con datos básicos (nombre, producto asociado) y asigna automáticamente un identificador único. Esto asegura un control organizado y evita duplicidad de registros. Los datos se almacenan de forma persistente en un archivo para su consulta futura.
- **Depósitos y Retiros:** Estas operaciones actualizan los saldos de los clientes en tiempo real. Para depósitos, el sistema suma el monto ingresado al saldo actual, mientras que para retiros verifica que haya fondos suficientes antes de realizar la operación. Todas las transacciones son registradas en un archivo, lo que asegura transparencia y facilita auditorías.
- **Consulta de Saldos:** Proporciona a los usuarios información detallada sobre el saldo actual de un cliente, utilizando su identificador único. Esto permite un acceso rápido y preciso a los datos financieros.
- **Cambio de Clave:** Permite a los usuarios modificar la clave de acceso del sistema. Esta funcionalidad refuerza la seguridad al prevenir accesos no autorizados.
- **Persistencia de Datos:** Los datos de los clientes y las transacciones se guardan en archivos externos, lo que garantiza que la información esté disponible incluso después de cerrar el sistema. Esto proporciona una solución robusta y confiable para el manejo de la información.

DISENO ESQUEMATICO

```
Proceso SistemaBancario
    // Declaración de constantes y variables globales
    Definir MAX_CLIENTES, numClientes Como Entero;
    MAX_CLIENTES <- 100;
    numClientes <- 0;

    // Declaración de matrices para almacenar datos de los clientes
    Dimension clientesID(MAX_CLIENTES);
    Dimension clientesNombre(MAX_CLIENTES);
    Dimension clientesSaldo(MAX_CLIENTES);

    // Iniciar el proceso con el login
    login(numClientes, clientesID, clientesNombre, clientesSaldo);

FinProceso

Subproceso login(numClientes, clientesID, clientesNombre, clientesSaldo)
    Definir clave, claveIngresada Como Caracter;
    Definir intentos Como Entero;
    intentos <- 0;
    clave <- "1234"; // Clave predeterminada

    Mientras intentos < 3 Hacer
        Escribir "Ingrese su clave: ";
        Leer claveIngresada;
```

```
        Si clave = claveIngresada Entonces
            Escribir "Login exitoso";
            menu(numClientes, clientesID, clientesNombre, clientesSaldo);
        SiNo
            Escribir "Clave incorrecta";
            intentos <- intentos + 1;
        FinSi
    FinMientras
FinSubproceso

Subproceso registrarCliente(numClientes, clientesID, clientesNombre, clientesSaldo)
    Definir nombre Como Caracter;
    Si numClientes >= MAX_CLIENTES Entonces
        Escribir "No se pueden agregar más clientes.";
            FinSi

            numClientes <- numClientes + 1;
            clientesID(numClientes) <- numClientes;

            Escribir "Ingrese el nombre del cliente: ";
            Leer nombre;
            clientesNombre(numClientes) <- nombre;

            clientesSaldo(numClientes) <- 0.0;

            Escribir "Cliente registrado con éxito. ID del cliente: ", clientesID(numClientes);
FinSubproceso
```

```
Subproceso depositar(numClientes, clientesID, clientesNombre, clientesSaldo)
    Definir id Como Entero;
    Definir monto Como Real;

    Escribir "Ingrese el ID del cliente: ";
    Leer id;

    Si id < 1 O id > numClientes Entonces
       Escribir "Cliente no encontrado.";
          FinSi

          Escribir "Ingrese el monto a depositar: ";
          Leer monto;

          clientesSaldo(id) <- clientesSaldo(id) + monto;

          Escribir "Depósito realizado con éxito. Nuevo saldo: ", clientesSaldo(id);
FinSubproceso

Subproceso retirar(numClientes, clientesID, clientesNombre, clientesSaldo)
    Definir id Como Entero;
    Definir monto Como Real;

    Escribir "Ingrese el ID del cliente: ";
    Leer id;
```

```
    Si id < 1 O id > numClientes Entonces
        Escribir "Cliente no encontrado.";
        FinSi

        Escribir "Ingrese el monto a retirar: ";
        Leer monto;

        Si clientesSaldo(id) >= monto Entonces
            clientesSaldo(id) <- clientesSaldo(id) - monto;
            Escribir "Retiro realizado con éxito. Nuevo saldo: ", clientesSaldo(id);
        SiNo
            Escribir "Saldo insuficiente.";
        FinSi
FinSubProceso


Subproceso consultarSaldo(numClientes, clientesID, clientesNombre, clientesSaldo)
    Definir id Como Entero;

    Escribir "Ingrese el ID del cliente: ";
    Leer id;

    Si id < 1 O id > numClientes Entonces
        Escribir "Cliente no encontrado.";
FinSi

Escribir "Cliente: ", clientesNombre(id), " - Saldo actual: ", clientesSaldo(id);
```

```
FinSubproceso


Subproceso menu(numClientes, clientesID, clientesNombre, clientesSaldo)
    Definir opcion Como Entero;
    Repetir
        Escribir "=== Menú Principal ===";
        Escribir "1. Registrar cliente";
        Escribir "2. Depositar";
        Escribir "3. Retirar";
        Escribir "4. Consultar saldo";
        Escribir "5. Salir";
        Escribir "Seleccione una opción: ";
        Leer opcion;

        Segun opcion Hacer
            Caso 1:
                registrarCliente(numClientes, clientesID, clientesNombre, clientesSaldo);
            Caso 2:
                depositar(numClientes, clientesID, clientesNombre, clientesSaldo);
            Caso 3:
                retirar(numClientes, clientesID, clientesNombre, clientesSaldo);
            Caso 4:
                consultarSaldo(numClientes, clientesID, clientesNombre, clientesSaldo);
            Caso 5:
                Escribir "Saliendo del programa...";
            De Otro Modo:
                Escribir "Opción inválida.";
```

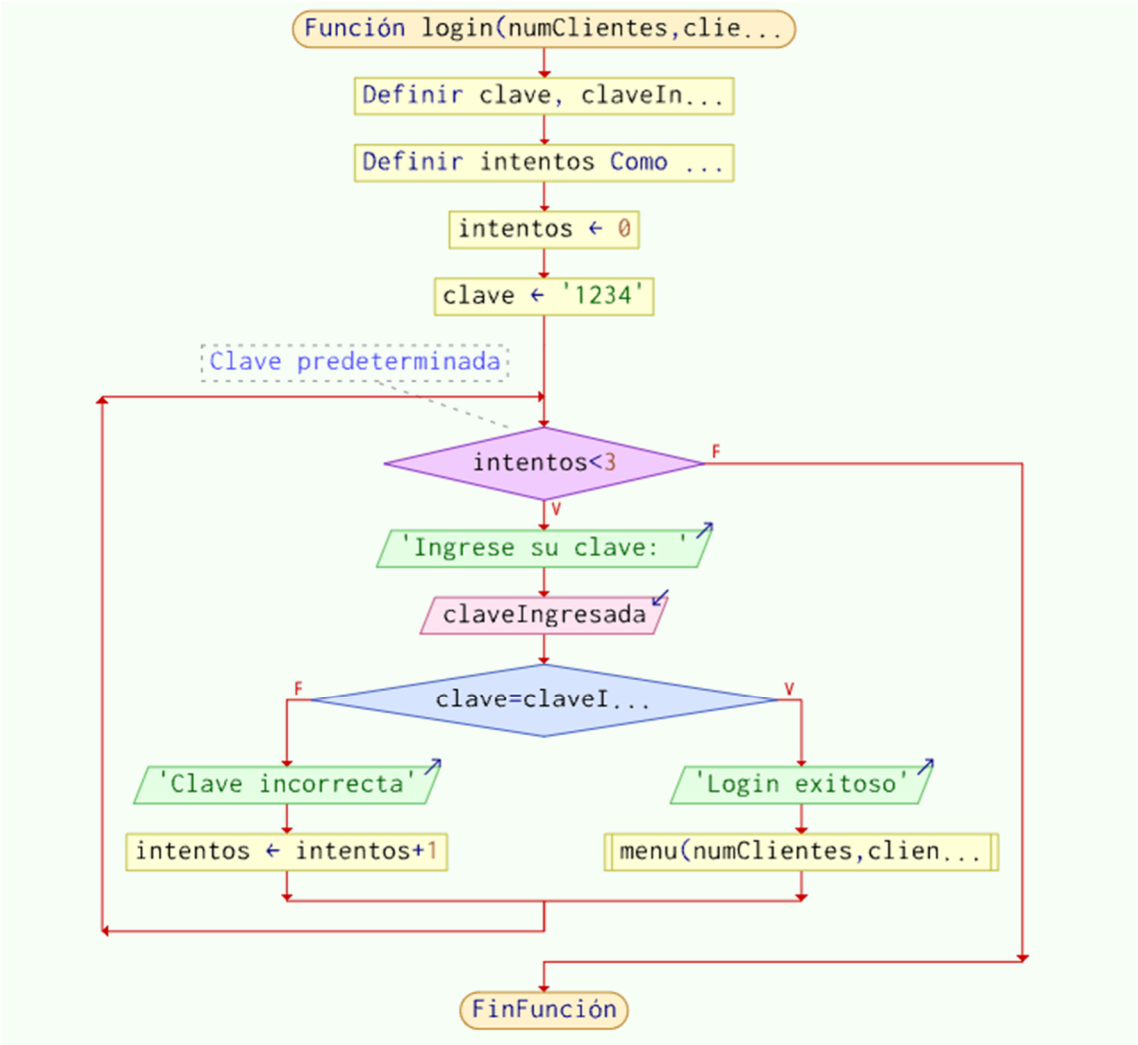
FinSegun

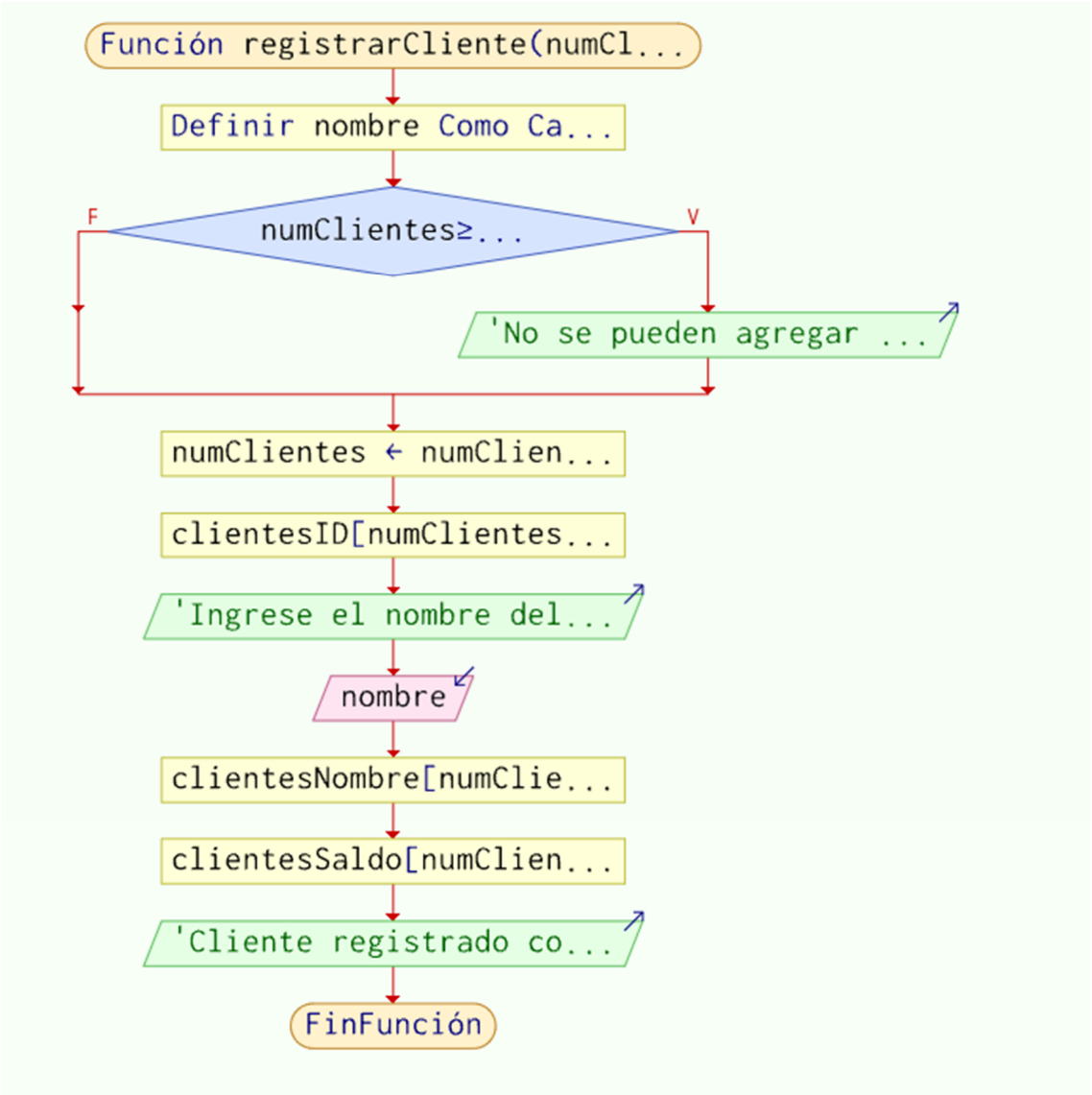
Hasta Que opcion = 5

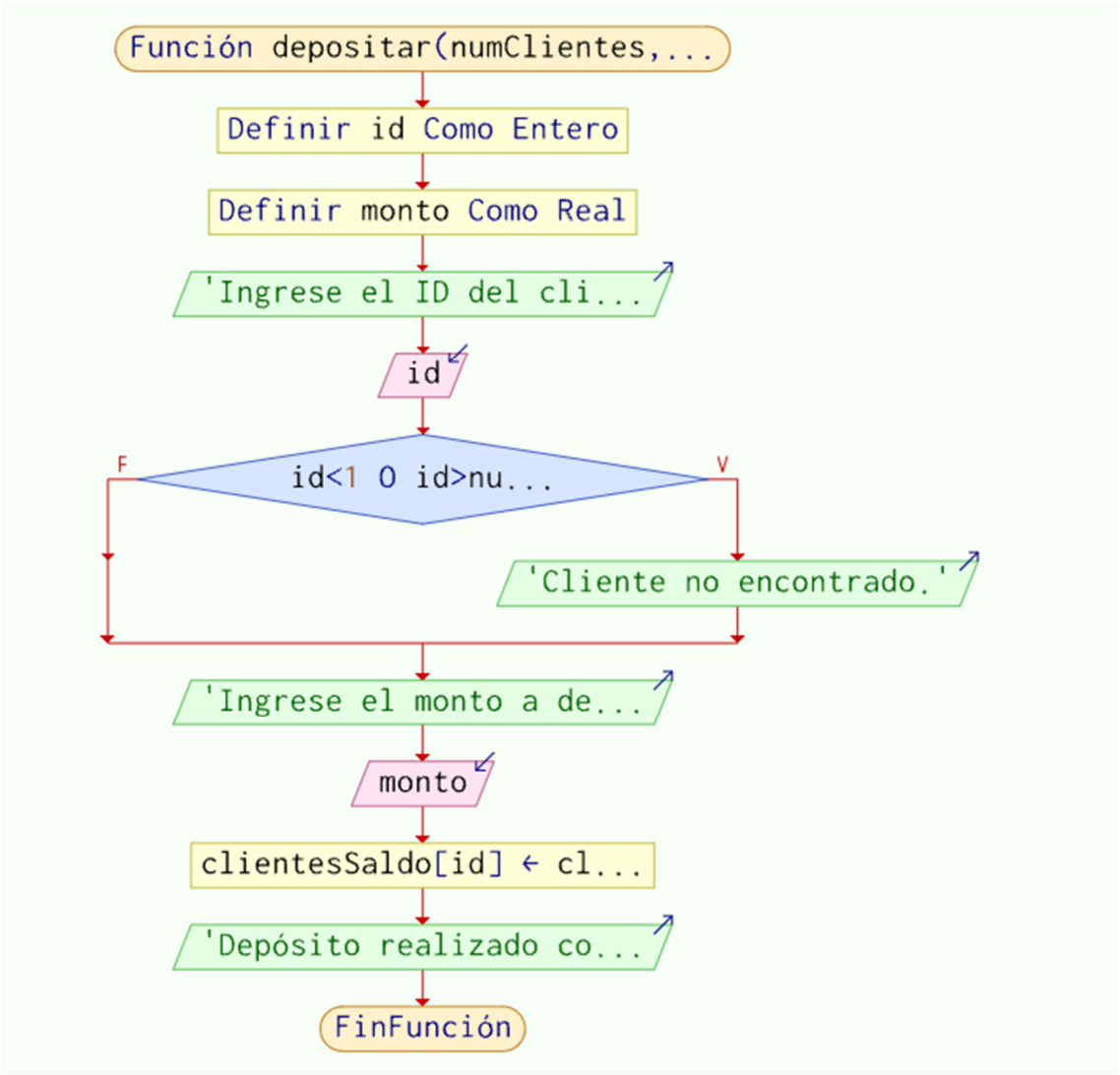
FinSubproceso

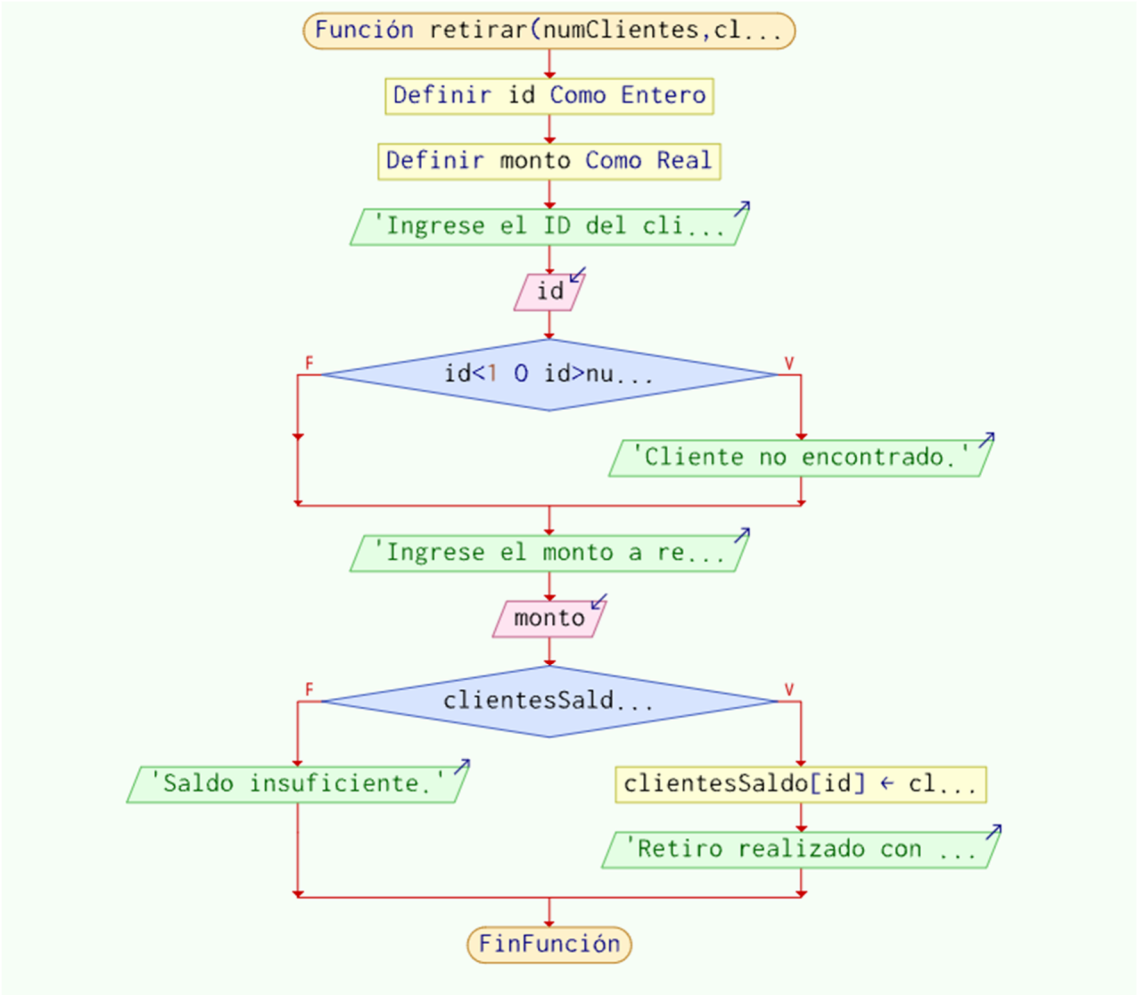
UNIVERSIDAD TECNOLOGICA DE SANTIAGO
DEPARTAMENTO DE INGENIERIA ELECTRONICA

TALLER PROGRAMACION 1
IET-725-001



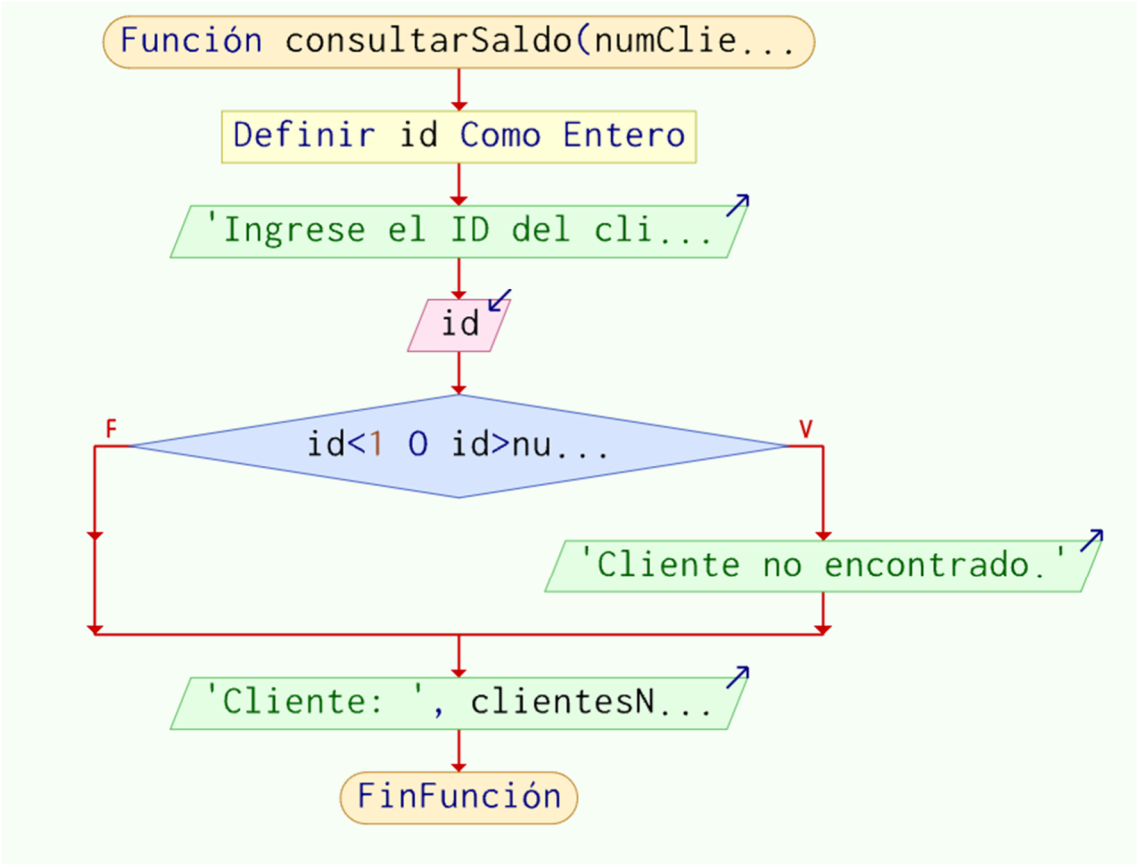

Función registrarCliente(numCl...
Definir nombre Como Ca...
F
numClientes≥...
V
'No se pueden agregar ...
numClientes ← numClien...
clientesID[numClientes...
'Ingrese el nombre del...
nombre
clientesNombre[numClie...
clientesSaldo[numClien...
'Cliente registrado co...
FinFunción

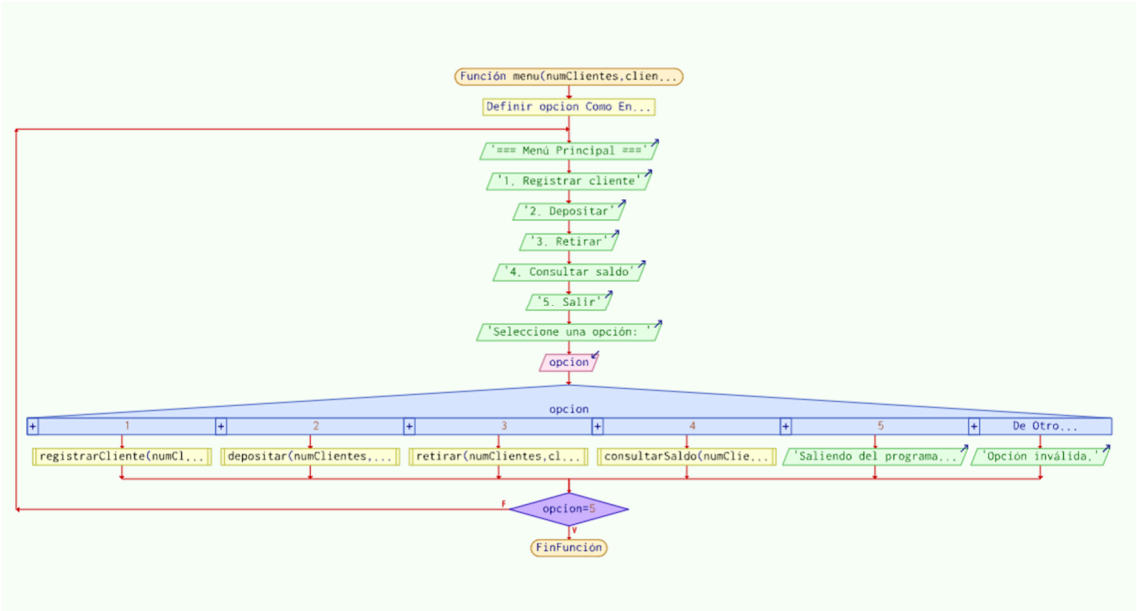

Función depositar(numClientes,...
Definir id Como Entero
Definir monto Como Real
'Ingrese el ID del cli...
id
F
id<1 O id>nu...
V
'Cliente no encontrado.'
'Ingrese el monto a de...
monto
clientesSaldo[id] ← cl...
'Depósito realizado co...
FinFunción


Función retirar(numClientes,cl...
Definir id Como Entero
Definir monto Como Real
'Ingrese el ID del cli...
id
F
id<1 O id>nu...
V
'Cliente no encontrado.'
'Ingrese el monto a re...
monto
F
clientesSald...
V
'Saldo insuficiente.'
clientesSaldo[id] ← cl...
'Retiro realizado con ...
FinFunción

UNIVERSIDAD TECNOLOGICA DE SANTIAGO
DEPARTAMENTO DE INGENIERIA ELECTRONICA

TALLER PROGRAMACION 1
IET-725-001





CODIGO FUENTE

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

#define MAX_CLIENTES 100
#define FICHERO_TRANSACCIONES "transacciones.txt"
#define FICHERO_CLIENTES "clientes.txt"
#define FICHERO_CLAVE "clave.txt"

typedef struct {
    int id;
    char nombre[50];
    float saldo;
    char producto[50]; // Producto asociado al cliente
} Cliente;

Cliente clientes[MAX_CLIENTES];
int numClientes = 0;

// Prototipos
void login();
void menu();
void registrarCliente();
void depositar();
void retirar();
void consultarSaldo();
void cambiarClave();
void guardarTransaccion(int id, const char *transaccion, float monto);
void guardarClientesEnArchivo();

int main() {
    login();
    return 0;
}

void login() {
    char clave[10], claveIngresada[10];
    int intentos = 0;

    FILE *archivo = fopen(FICHERO_CLAVE, "r");
    if (!archivo) {
        archivo = fopen(FICHERO_CLAVE, "w");
        fprintf(archivo, "1234"); // Clave predeterminada
        fclose(archivo);
```

```
        archivo = fopen(FICHERO_CLAVE, "r");
    }
    fscanf(archivo, "%s", clave);
    fclose(archivo);

    while (intentos < 3) {
        printf("Ingrese su clave: ");
        scanf("%s", claveIngresada);
        if (strcmp(clave, claveIngresada) == 0) {
            printf("Login exitoso\n");
            menu();
            return;
        } else {
            printf("Clave incorrecta\n");
            intentos++;
        }
    }
    printf("Demasiados intentos fallidos. Programa finalizado.\n");
}

void menu() {
    int opcion;
    do {
        printf("\n=== Menu Principal ===\n");
        printf("1. Registrar cliente\n");
        printf("2. Depositar\n");
        printf("3. Retirar\n");
        printf("4. Consultar saldo\n");
        printf("5. Cambiar clave\n");
        printf("6. Salir\n");
        printf("Seleccione una opcion: ");
        scanf("%d", &opcion);

        switch (opcion) {
            case 1: registrarCliente(); break;
            case 2: depositar(); break;
            case 3: retirar(); break;
            case 4: consultarSaldo(); break;
            case 5: cambiarClave(); break;
            case 6: printf("Saliendo del programa...\n"); break;
            default: printf("Opcion invalida.\n");
        }
    } while (opcion != 6);
}

void registrarCliente() {
```

```
    if (numClientes >= MAX_CLIENTES) {
        printf("No se pueden agregar mas clientes.\n");
        return;
    }
    Cliente nuevoCliente;
    nuevoCliente.id = numClientes + 1;
    printf("Ingrese el nombre del cliente: ");
    scanf(" %[^\n]s", nuevoCliente.nombre);
    printf("Ingrese el producto asociado al cliente: ");
    scanf(" %[^\n]s", nuevoCliente.producto);
    nuevoCliente.saldo = 0.0;
    clientes[numClientes++] = nuevoCliente;
    printf("Cliente registrado con ID %d.\n", nuevoCliente.id);
    guardarClientesEnArchivo();
}

void depositar() {
    int id;
    float monto;
    printf("Ingrese el ID del cliente: ");
    scanf("%d", &id);
    if (id < 1 || id > numClientes) {
        printf("Cliente no encontrado.\n");
        return;
    }
    printf("Ingrese el monto a depositar: ");
    scanf("%f", &monto);
    clientes[id - 1].saldo += monto;
    printf("Deposito exitoso. Nuevo saldo: %.2f\n", clientes[id -
1].saldo);
    guardarTransaccion(id, "Deposito", monto);
}

void retirar() {
    int id;
    float monto;
    printf("Ingrese el ID del cliente: ");
    scanf("%d", &id);
    if (id < 1 || id > numClientes) {
        printf("Cliente no encontrado.\n");
        return;
    }
    printf("Ingrese el monto a retirar: ");
    scanf("%f", &monto);
    if (clientes[id - 1].saldo < monto) {
        printf("Saldo insuficiente.\n");
```

```
    } else {
        clientes[id - 1].saldo -= monto;
        printf("Retiro exitoso. Nuevo saldo: %.2f\n", clientes[id -
1].saldo);
        guardarTransaccion(id, "Retiro", monto);
    }
}

void consultarSaldo() {
    int id;
    printf("Ingrese el ID del cliente: ");
    scanf("%d", &id);
    if (id < 1 || id > numClientes) {
        printf("Cliente no encontrado.\n");
        return;
    }
    printf("Saldo del cliente %s: %.2f\n", clientes[id - 1].nombre,
clientes[id - 1].saldo);
}

void cambiarClave() {
    char nuevaClave[10];
    FILE *archivo = fopen(FICHERO_CLAVE, "w");
    if (!archivo) {
        printf("Error al cambiar la clave.\n");
        return;
    }
    printf("Ingrese la nueva clave: ");
    scanf("%s", nuevaClave);
    fprintf(archivo, "%s", nuevaClave);
    fclose(archivo);
    printf("Clave cambiada exitosamente.\n");
}

void guardarTransaccion(int id, const char *transaccion, float monto) {
    FILE *archivo = fopen(FICHERO_TRANSACCIONES, "a");
    if (!archivo) {
        printf("Error al guardar la transaccion.\n");
        return;
    }
    fprintf(archivo, "ID: %d | Cliente: %s | Transaccion: %s | Monto:
%.2f\n",
            clientes[id - 1].id, clientes[id - 1].nombre, transaccion,
monto);
    fclose(archivo);
}
```

```
void guardarClientesEnArchivo() {
    FILE *archivo = fopen(FICHERO_CLIENTES, "w");
    if (!archivo) {
        printf("Error al guardar los datos de los clientes.\n");
        return;
    }
    for (int i = 0; i < numClientes; i++) {
        fprintf(archivo, "ID: %d | Nombre: %s | Producto: %s | Saldo:
%.2f\n",
                clientes[i].id, clientes[i].nombre, clientes[i].producto,
clientes[i].saldo);
    }
    fclose(archivo);
    printf("Datos de los clientes guardados exitosamente.\n");
}
```

PRUEBAS DE INTEGRACION

El sistema diseñado fue sometido a diversas pruebas de funcionalidad para garantizar que las distintas partes trabajen correctamente. A continuación, se describen tres escenarios clave de prueba, incluyendo los valores de entrada, salida esperada, resultados obtenidos y observaciones:

Escenario 1: Registro de Cliente y Consulta de Saldo

Valores de Entrada:

- Nombre del cliente: "Raúl Toribio"
- Producto asociado: "Cuenta de Ahorros"
- Pasos de Prueba:

1. Registrar un cliente con los datos indicados.
2. Consultar el saldo inicial del cliente registrado.

Resultados Esperados:

- El sistema debe asignar un ID único al cliente y registrar sus datos con un saldo inicial de 0.00.
- Al consultar el saldo, debe mostrar el nombre del cliente y un saldo de 0.00.

Resultados Obtenidos:

- Cliente registrado correctamente con ID 1.
- La consulta de saldo muestra: "Saldo del cliente Raúl Toribio: 0.00".

Observaciones:

- Funcionalidad operativa y sin errores.

Escenario 2: Depósito y Registro de Transacción

Valores de Entrada:

- ID del cliente: 1
- Monto del depósito: 500.00

Pasos de Prueba:

1. Realizar un depósito al cliente con ID 1.
2. Consultar nuevamente el saldo del cliente.
3. Verificar el archivo de transacciones para confirmar el registro.

Resultados Esperados:

- El saldo del cliente debe aumentar a 500.00.
- El archivo de transacciones debe registrar correctamente el depósito realizado.

Resultados Obtenidos:

- El sistema muestra: "Depósito exitoso. Nuevo saldo: 500.00".
- El archivo registra: "ID: 1 | Cliente: Raul Toribio | Transacción: Depósito | Monto: 500.00".

Observaciones:

- Operación realizada correctamente y registro en archivo funcional.

Escenario 3: Retiro con Saldo Insuficiente y Cambio de Clave

Valores de Entrada:

- ID del cliente: 1
- Monto del retiro: 600.00
- Nueva clave: "5678"

Pasos de Prueba:

- Intentar retirar más dinero del saldo disponible.
- Cambiar la clave del sistema.
- Realizar un nuevo inicio de sesión con la clave modificada.

Resultados Esperados:

- El sistema debe mostrar un mensaje de "Saldo insuficiente" y no modificar el saldo del cliente.
- La clave debe actualizarse y permitir el acceso al sistema solo con la nueva clave.

Resultados Obtenidos:

- El sistema muestra: "Saldo insuficiente". El saldo permanece en 500.00.
- Cambio de clave exitoso. La nueva clave permitió iniciar sesión correctamente.

Observaciones:

- Las restricciones de saldo y el cambio de clave funcionan como se espera.

Las pruebas realizadas confirmaron la funcionalidad integral del sistema en distintas etapas.

MANUAL DE USUARIO

1. Inicio de Sesión (Login)

1. Al iniciar el programa, el sistema solicitará una clave de acceso.
2. Ingrese la clave correcta (predeterminada: 1234).
    - Si la clave es incorrecta, puede intentar nuevamente hasta tres veces.
    - Después de tres intentos fallidos, el programa se cerrará automáticamente.
3. Si la clave es correcta, accederá al **Menú Principal**

2. Menú Principal

El menú ofrece las siguientes opciones:

1. **Registrar Cliente**
2. **Depositar**
3. **Retirar**
4. **Consultar Saldo**
5. **Cambiar Clave**
6. **Salir**

Seleccione una opción ingresando el número correspondiente y presione Enter.

3. Opciones del Menú

3.1 Registrar Cliente

1. Ingrese el nombre del cliente cuando se le solicite.
2. Ingrese el producto asociado al cliente (por ejemplo, "Cuenta de Ahorros").
3. El sistema asignará automáticamente un ID único al cliente y lo registrará con un saldo inicial de 0.00.
4. El sistema confirmará el registro y guardará los datos.

3.2 Depositar

1. Ingrese el ID del cliente al que desea depositar.
2. Ingrese el monto del depósito.
3. El sistema actualizará el saldo del cliente y confirmará la transacción.
4. Los detalles de la transacción se guardarán en un archivo externo para referencia.

3.3 Retirar

1. Ingrese el ID del cliente del cual desea retirar fondos.
2. Ingrese el monto a retirar.
3. Si el cliente tiene saldo suficiente, el sistema descontará el monto e informará el nuevo saldo.
4. En caso de saldo insuficiente, el sistema mostrará un mensaje de error y no realizará cambios.

3.4 Consultar Saldo

1. Ingrese el ID del cliente cuyo saldo desea consultar.
2. El sistema mostrará el nombre del cliente y su saldo actual.

3.5 Cambiar Clave

1. Ingrese la nueva clave cuando el sistema lo solicite.
2. La clave será actualizada y guardada en un archivo externo.
3. Use la nueva clave para iniciar sesión en el futuro.

3.6 Salir

Seleccione esta opción para cerrar el programa de forma segura

5. Mensajes de Error Comunes

- **"Cliente no encontrado":** Asegúrese de ingresar un ID válido.
- **"Saldo insuficiente":** Verifique que el cliente tenga fondos suficientes para realizar el retiro.
- **"Clave incorrecta":** Introduzca la clave correcta o cambie la clave si es necesario.

CONCLUSIONES

El sistema cumple con los objetivos planteados, permitiendo registrar clientes, realizar depósitos y retiros, consultar saldos y cambiar la clave. Todo funciona correctamente y los datos se guardan sin problemas.

Para mejorar, considero que sería útil agregar una interfaz gráfica, más validaciones como monto mínimo para operaciones y la capacidad de generar reportes. También se podría migrar el sistema a una base de datos para manejar más clientes, permitir transferencias entre ellos y agregar notificaciones de operaciones por correo. Además, una mayor seguridad con autenticación multifactorial sería ideal.