|  |
| --- |
|  |
| Prueba de Caja Blanca |
| ***“PLATAFORMA DE GESTIÓN INTEGRAL PARA EL CONTROL DE CLIENTES, SERVICIOS Y FINANZAS EN UNA ELECTROMECÁNICA”*** |
|  |
|  |
| **Integrantes:**  **Benavides Xavier, Yacelga Iván, Suasnavas Gabriel, Jami Klever**  **Fecha: 2025-01-21** |

## Prueba caja blanca de: Menú Principal

## CÓDIGO FUENTE

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define MAX\_CLIENTES 100

#define MAX\_TRANSACCIONES 100

typedef struct {

int id;

char nombre[50];

char correo[50];

} Cliente;

typedef struct {

int id;

int idCliente;

double monto;

char fecha[20];

} Transaccion;

Cliente clientes[MAX\_CLIENTES];

Transaccion transacciones[MAX\_TRANSACCIONES];

int contadorClientes = 0;

int contadorTransacciones = 0;

void registrarCliente();

void gestionarTransacciones();

void generarReportes();

void verHistorialTransacciones();

void buscarCliente();

int main() {

int opcion;

do {

printf("\n--- Sistema de Gestion de Clientes ---\n");

printf("1. Registrar Cliente\n");

printf("2. Gestionar Transacciones\n");

printf("3. Generar Reportes\n");

printf("4. Ver Historial de Transacciones\n");

printf("5. Buscar Cliente\n");

printf("6. Salir\n");

printf("Elija una opcion: ");

scanf("%d", &opcion);

if (opcion == 1) {

registrarCliente();

} else if (opcion == 2) {

gestionarTransacciones();

} else if (opcion == 3) {

generarReportes();

} else if (opcion == 4) {

verHistorialTransacciones();

} else if (opcion == 5) {

buscarCliente();

} else if (opcion == 6) {

printf("Saliendo...\n");

} else {

printf("Opcion invalida. Por favor, intente de nuevo.\n");

}

} while (opcion != 6);

return 0;

}

void registrarCliente() {

if (contadorClientes >= MAX\_CLIENTES) {

printf("Se ha alcanzado el limite de clientes.\n");

return;

}

Cliente nuevoCliente;

nuevoCliente.id = contadorClientes + 1;

printf("Ingrese el nombre del cliente: ");

scanf("%s", nuevoCliente.nombre);

printf("Ingrese el correo del cliente: ");

scanf("%s", nuevoCliente.correo);

clientes[contadorClientes++] = nuevoCliente;

printf("Cliente registrado exitosamente con ID: %d\n", nuevoCliente.id);

return;

}

void gestionarTransacciones() {

if (contadorTransacciones >= MAX\_TRANSACCIONES) {

printf("Se ha alcanzado el limite de transacciones.\n");

return;

}

Transaccion nuevaTransaccion;

printf("Ingrese el ID del cliente: ");

scanf("%d", &nuevaTransaccion.idCliente);

int clienteExiste = 0;

for (int i = 0; i < contadorClientes; i++) {

if (clientes[i].id == nuevaTransaccion.idCliente) {

clienteExiste = 1;

break;

}

}

if (!clienteExiste) {

printf("ID de cliente no encontrado.\n");

return;

}

nuevaTransaccion.id = contadorTransacciones + 1;

printf("Ingrese el monto de la transaccion: ");

scanf("%lf", &nuevaTransaccion.monto);

printf("Ingrese la fecha de la transaccion (AAAA-MM-DD): ");

scanf("%s", nuevaTransaccion.fecha);

transacciones[contadorTransacciones++] = nuevaTransaccion;

printf("Transaccion registrada exitosamente con ID: %d\n", nuevaTransaccion.id);

return;

}

void generarReportes() {

double ingresosTotales = 0, egresosTotales = 0;

for (int i = 0; i < contadorTransacciones; i++) {

if (transacciones[i].monto > 0) {

ingresosTotales += transacciones[i].monto;

} else {

egresosTotales += transacciones[i].monto;

}

}

printf("\n--- Reporte Financiero ---\n");

printf("Ingresos Totales: $%.2lf\n", ingresosTotales);

printf("Egresos Totales: $%.2lf\n", egresosTotales);

printf("Balance Neto: $%.2lf\n", ingresosTotales + egresosTotales);

return;

}

void verHistorialTransacciones() {

int idCliente;

printf("Ingrese el ID del cliente: ");

scanf("%d", &idCliente);

printf("\n--- Historial de Transacciones para el Cliente ID: %d ---\n", idCliente);

for (int i = 0; i < contadorTransacciones; i++) {

if (transacciones[i].idCliente == idCliente) {

printf("ID de Transaccion: %d, Monto: $%.2lf, Fecha: %s\n",

transacciones[i].id, transacciones[i].monto, transacciones[i].fecha);

}

}

return;

}

void buscarCliente() {

char nombre[50];

printf("Ingrese el nombre del cliente a buscar: ");

scanf("%s", nombre);

printf("\n--- Resultados de Busqueda ---\n");

for (int i = 0; i < contadorClientes; i++) {

if (strstr(clientes[i].nombre, nombre) != NULL) {

printf("ID del Cliente: %d, Nombre: %s, Correo: %s\n",

clientes[i].id, clientes[i].nombre, clientes[i].correo);

}

}

return;

}

**DIAGRAMA DE FLUJO (DF)**

Diagrama de flujo generado para el menú principa

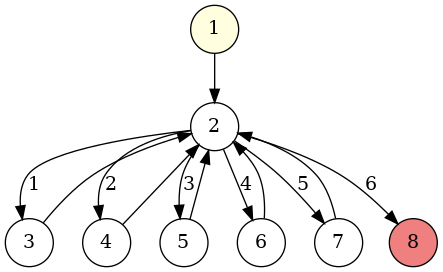
Diagrama, Dibujo de ingeniería, Esquemático

Descripción generada automáticamente

l.

## GRAFO DE FLUJO (GF)

Grafo de flujo generado a partir del diagrama de flujo.



## IDENTIFICACIÓN DE LAS RUTAS

R1: Inicio -> Menú Principal -> Opción 1 -> Volver al Menú

R2: Inicio -> Menú Principal -> Opción 2 -> Volver al Menú

R3: Inicio -> Menú Principal -> Opción 3 -> Volver al Menú

R4: Inicio -> Menú Principal -> Opción 4 -> Volver al Menú

R5: Inicio -> Menú Principal -> Opción 5 -> Volver al Menú

R6: Inicio -> Menú Principal -> Opción 6 (Salir)

## COMPLEJIDAD CICLOMÁTICA

Se calcula de las siguientes formas:

1. V(G) = número de nodos predicados + 1

V(G) = 6 (opciones de menú) + 1 = 7

2. V(G) = A – N + 2

V(G) = 13 (aristas) – 7 (nodos) + 2 = 8