Prueba de Caja Blanca

"Título proyecto sistema de automatización de mensajes e ingreso de datos para fechas importantes"

Integrantes:

Alejandro De La Cruz
Santiago Nogales
Ian Escobar

Fecha 2025-06-16

Prueba caja blanca de describa el requisito funcional

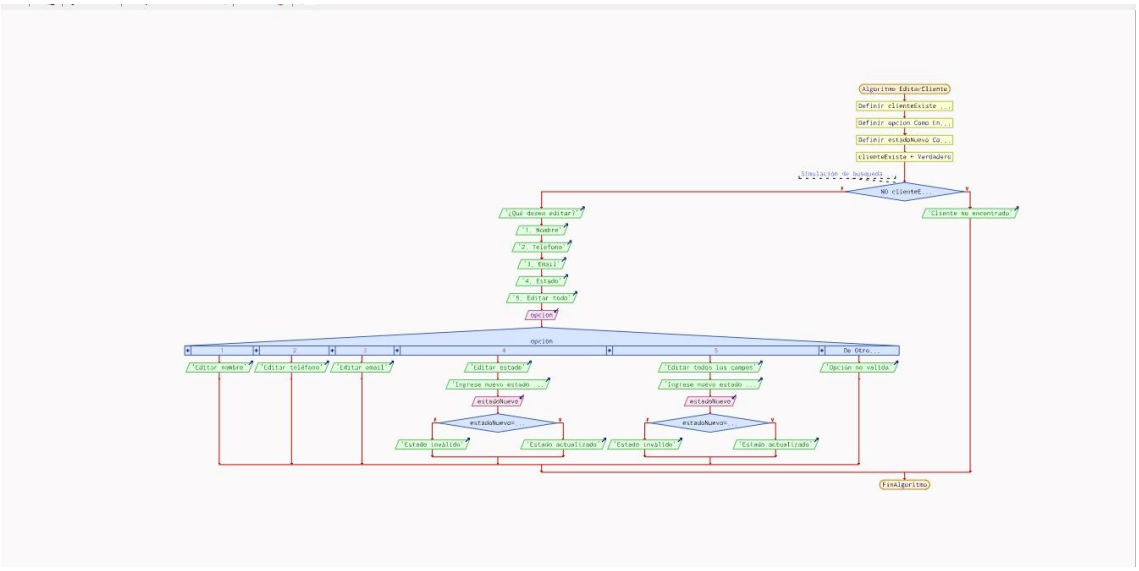
1. CÓDIGO FUENTE

Pegar el trozo de código fuente que se requiere para el caso de prueba

```
void editarCliente() {
    if (clientes.empty()) {
        std::cout << "No hay clientes registrados.\n";
        return;
    }

    listarClientes();
    int clienteId;
    std::cout << "\nIngrese el ID del cliente a editar: ";
    std::cin >> clienteId;

    auto clienteIt = std::find_if(clientes.begin(), clientes.end(),
        [clienteId](const Cliente& c) { return c.id == clienteId; });

    if (clienteIt == clientes.end()) {
        std::cout << "Cliente no encontrado.\n";
        return;
    }

    std::cout << "\n=== EDITAR CLIENTE ===\n";
    std::cout << "Cliente actual:\n";
    std::cout << "Nombre: " << clienteIt->nombre << "\nTeléfono: " << clienteIt->telefono
              << "\nEmail: " << clienteIt->email << "\nEstado: " << clienteIt->estado << std::endl;

    int opcion;
    std::cout << "\n¿Qué desea editar?\n1. Nombre\n2. Teléfono\n3. Email\n4. Estado\n5. Editar todo\nSeleccione una opción: ";
    std::cin >> opcion;
    std::string nuevoValor;
    std::cin.ignore();

    switch (opcion) {
        case 1:
            std::cout << "Nuevo nombre: ";
            std::getline(std::cin, nuevoValor);
            clienteIt->nombre = nuevoValor;
            break;
        case 2:
            std::cout << "Nuevo teléfono: ";
            std::getline(std::cin, nuevoValor);
            clienteIt->telefono = nuevoValor;
            break;
        case 3:
            std::cout << "Nuevo email: ";
            std::getline(std::cin, nuevoValor);
            clienteIt->email = nuevoValor;
            break;
        case 4:
            std::cout << "Nuevo estado (activo/inactivo): ";
            std::getline(std::cin, nuevoValor);
            if (nuevoValor == "activo" || nuevoValor == "inactivo") {
                clienteIt->estado = nuevoValor;
            } else {
                std::cout << "Estado inválido. Debe ser 'activo' o 'inactivo'.\n";
            }
            break;
        case 5:
            std::cout << "Nuevo nombre: ";
            std::getline(std::cin, clienteIt->nombre);
            std::cout << "Nuevo teléfono: ";
            std::getline(std::cin, clienteIt->telefono);
            std::cout << "Nuevo email: ";
            std::getline(std::cin, clienteIt->email);
            std::cout << "Nuevo estado (activo/inactivo): ";
            std::getline(std::cin, nuevoValor);
            if (nuevoValor == "activo" || nuevoValor == "inactivo") {
                clienteIt->estado = nuevoValor;
            }
            break;
        default:
            std::cout << "Opción no válida.\n";
            return;
    }

    std::cout << "Datos del cliente actualizados exitosamente.\n";
}
```

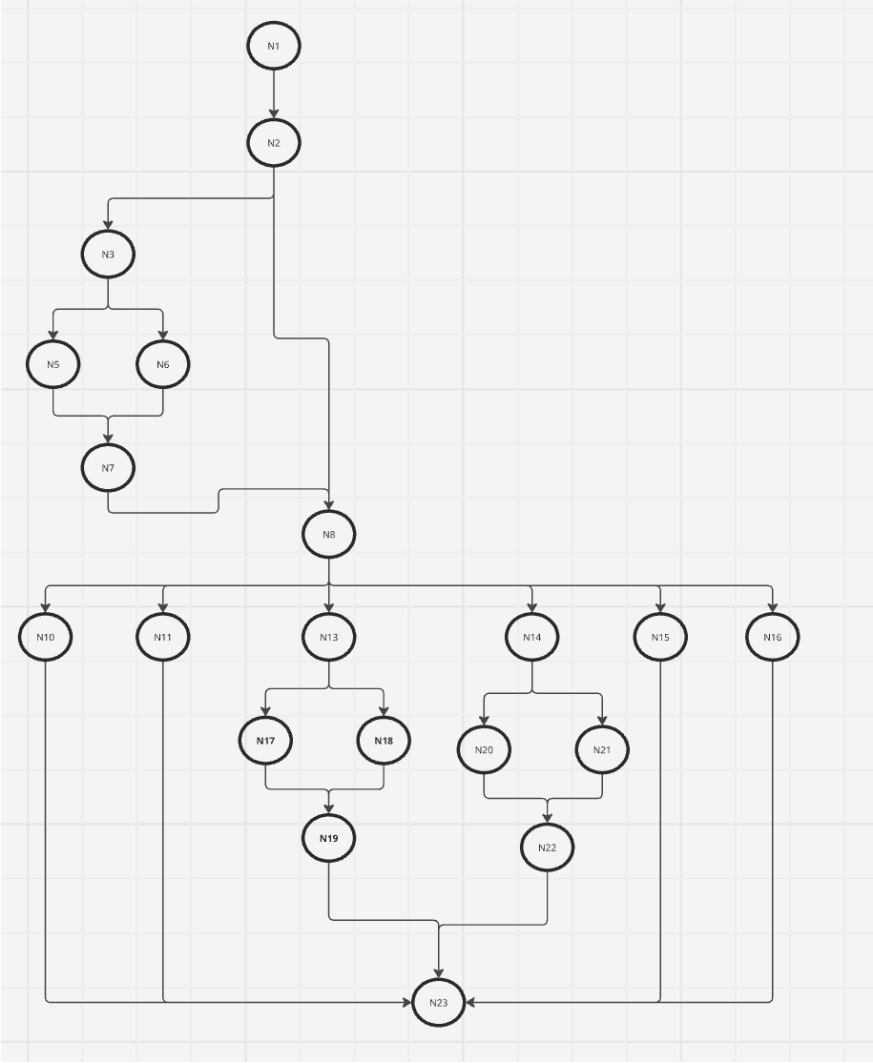
2. DIAGRAMA DE FLUJO (DF)

Realizar un DF del código fuente del numeral 1



3. GRAFO DE FLUJO (GF)

Realizar un GF en base al DF del numeral 2



4. IDENTIFIACCIÒN DE LAS RUTAS (Camino basico)

Determinar en base al GF del numeral 4
RUTAS

1. **R1:** N5 → N14
   - *Camino*: Seleccionar opción 0 (Salir) → Terminar programa.

2. **R2:** N5 → N15
   - *Camino*: Ingresar opción inválida → Mostrar error → Volver al menú.

3. **R3:** N5 → N6 → N16 → N17 → N5
   - *Camino*: Opción 1 (Agregar cliente) → Ejecutar función → Pausa → Volver al menú.
4. **R4:** N5 → N7 → N18 → N5
   - *Camino*: Opción 2 (Listar clientes) → Ejecutar → Volver al menú.
5. **R5:** N5 → N9 → N20 → N21 → N5
   - *Camino*: Opción 4 (Mensaje individual) → Enviar → Pausa → Volver al menú.

Se puede calcular de las siguientes formas:

A= 21
N= 18

- V(G) = número de nodos predicados(decisiones)+1
  V(G)= P = 8+1= 9

- V(G) = A – N + 2
  V(G)= 21−18+2 =5.

DONDE:
P: Número de nodos predicado
A: Número de aristas
N: Número de nodos