o

Prueba de Caja Blanca

"Caso venta dulces"

Integrantes:
COLLAGUAZO PABLO
RODRIGUEZ JEICOL
VILAÑA ANTHONY

Fecha 16/1/2025

Prueba caja blanca de REGISTRO DE PRODUCTOS

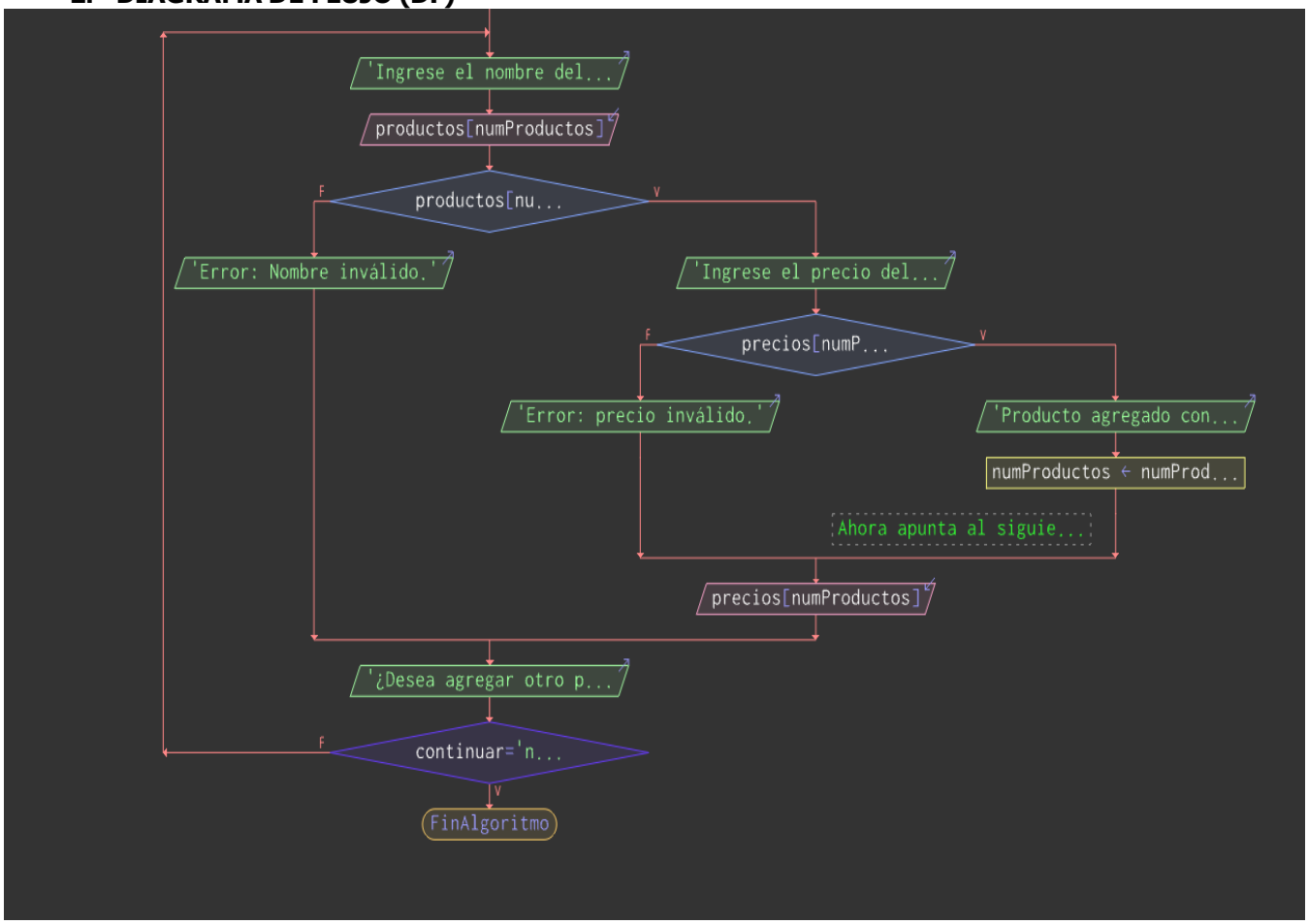
CASO DE PRUEBA 1

1. **CÓDIGO FUENTE**

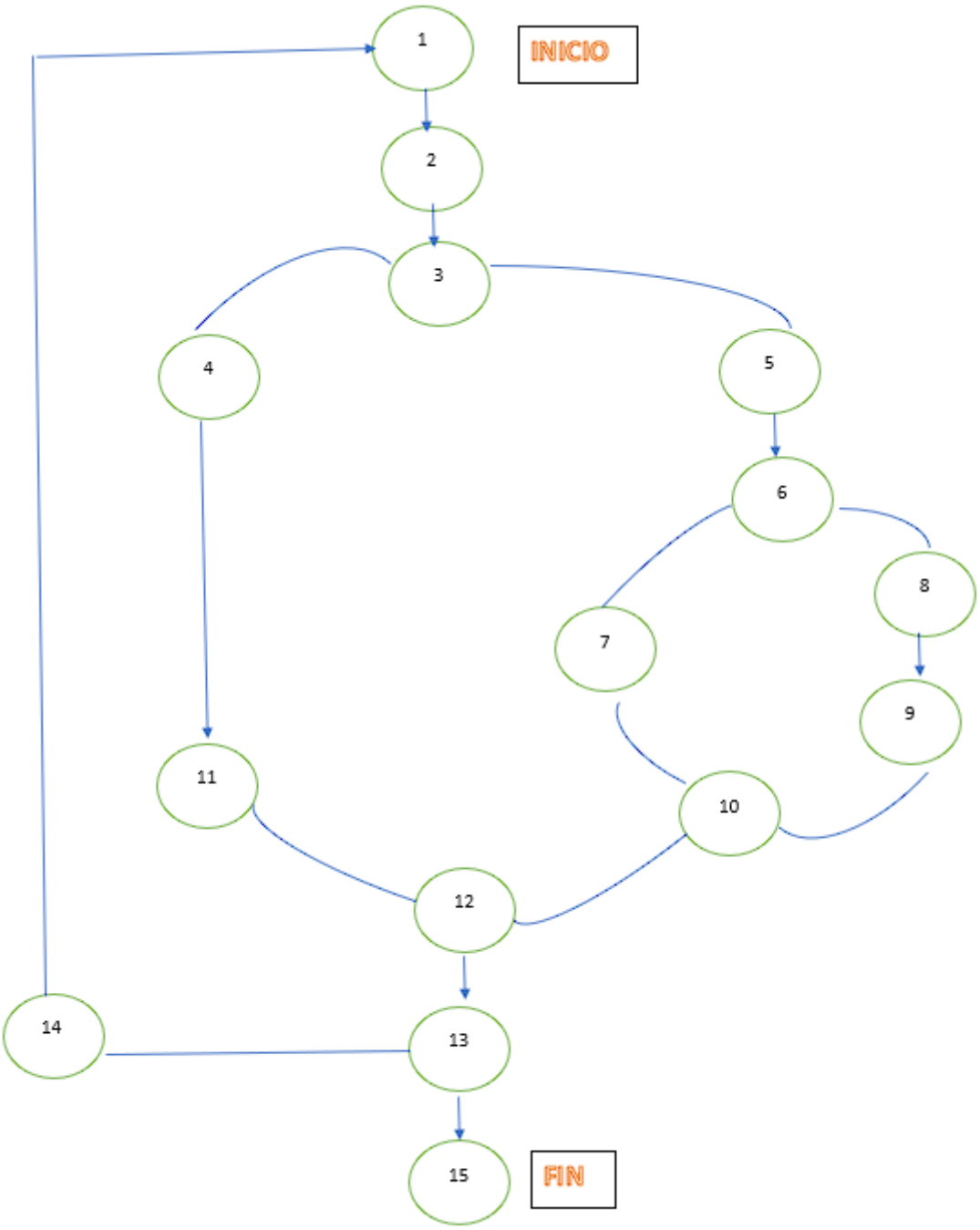
Ingreso de productos

```
// Agregar productos sin salir del menú
void agregarProducto() {
    char continuar;
    do {
        Producto p;
        string nombreProducto;
        cin.ignore();

        do {
            cout << "Ingrese el nombre del producto: ";
            getline(cin, nombreProducto);
            if (!esNombreValido(nombreProducto)) {
                cout << "Error: Nombre inválido." << endl;
            }
        } while (!esNombreValido(nombreProducto));

        p.nombre = nombreProducto;
        p.precio = ingresarPrecio();
        productos.push_back(p);
        cout << "Producto agregado con éxito!" << endl;

        cout << "¿Desea agregar otro producto? (s/n): ";
        cin >> continuar;
    } while (continuar == 's' || continuar == 'S');
}
```

2. DIAGRAMA DE FLUJO (DF)



3. GRAFO DE FLUJO (GF)



4. IDENTIFIACCIÒN DE LAS RUTAS (Camino basico)

RUTAS

R1: 1-2-3-4-11-12-13-15
R2: 1-2-3-5-6-8-9-10-12-13-15
R3: 1-2-3-5-6-7-10-12-13-15
R4: 1-2-3-4-11-12-13-14-15

5. COMPLEJIDAD CICLOMÁTICA

Se puede calcular de las siguientes formas:

- V(G) = número de nodos predicados(decisiones)+1
  V(G)= 3+1=4

- V(G) = A – N + 2
  V(G)= 17-15+2=4

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

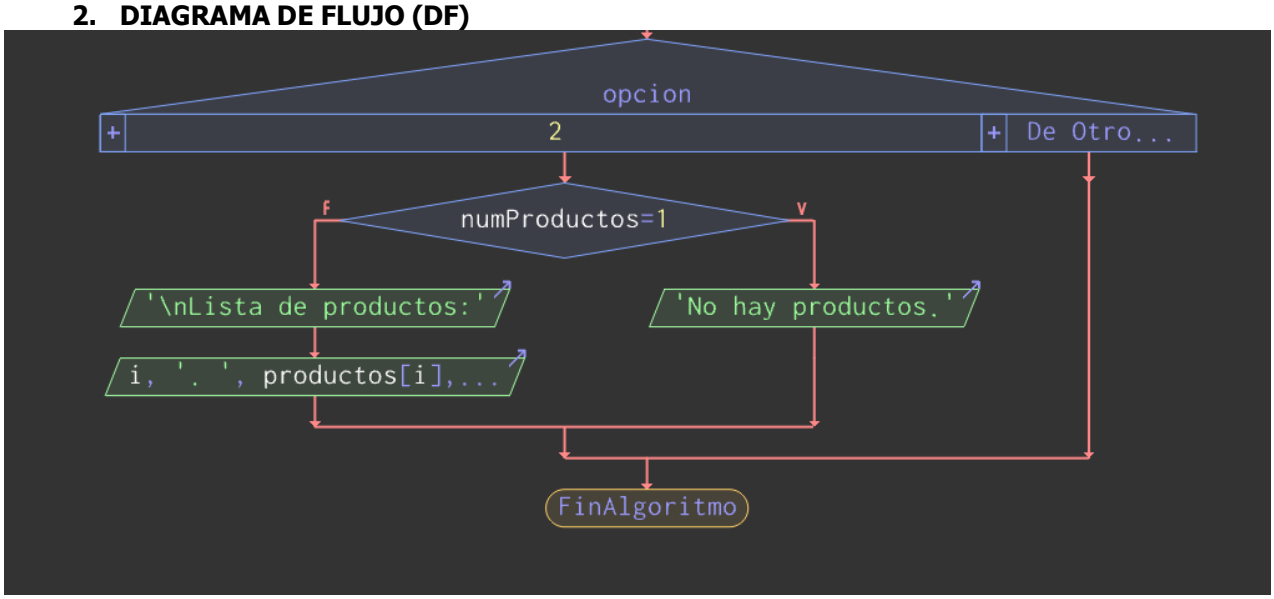
Prueba caja blanca de Lista de productos

1. **CÓDIGO FUENTE**

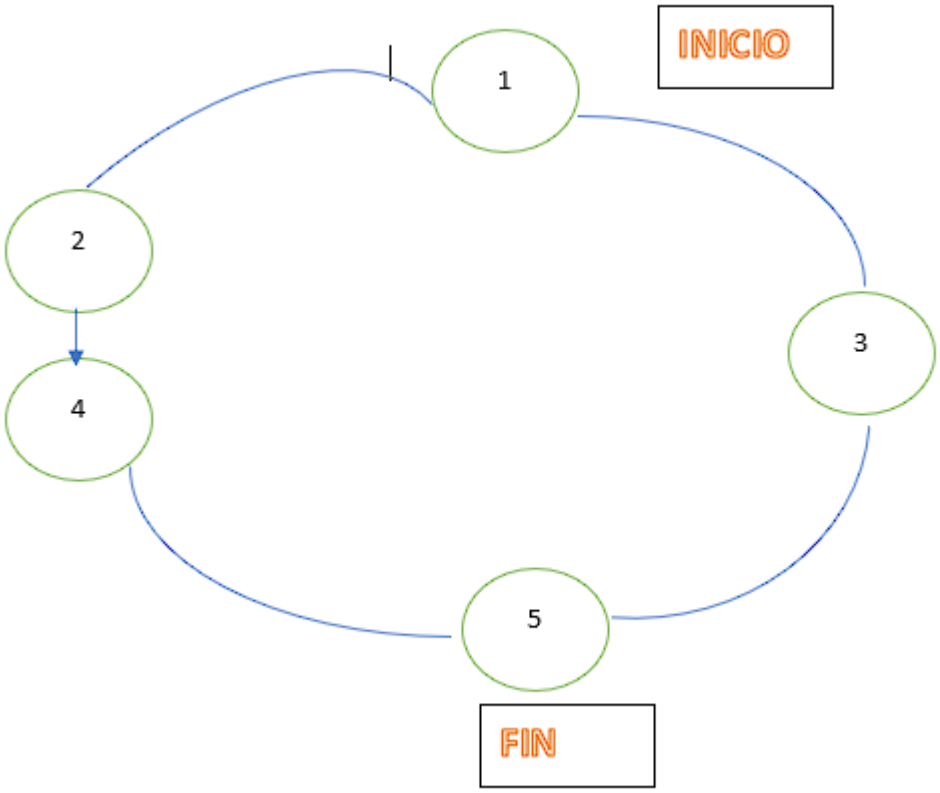
LISTA DE PRODUCTOS

```
// Mostrar productos disponibles
void mostrarProductos(const vector<Producto>& lista) {
    if (lista.empty()) {
        cout << "No hay productos." << endl;
        return;
    }
    cout << "\nLista de productos:" << endl;
    for (size_t i = 0; i < lista.size(); i++) {
        cout << i + 1 << ". " << lista[i].nombre << " - $" << fixed << setprecision(2) << lista[i].precio << endl;
    }
}
```

2. **DIAGRAMA DE FLUJO (DF)**



3. GRAFO DE FLUJO (GF)



4. IDENTIF IACCIÒN DE LAS RUTAS (Camino basico)

Determinar en base al GF del numeral 4
RUTAS

R1: 1-2-4-5
R2: 1-3-5

5. COMPLEJIDAD CICLOMÁTICA

Se puede calcular de las siguientes formas:

- V(G) = número de nodos predicados(decisiones)+1
  V(G)=1+1=2

- V(G) = A – N + 2
  V(G)= 5- 5+2=2

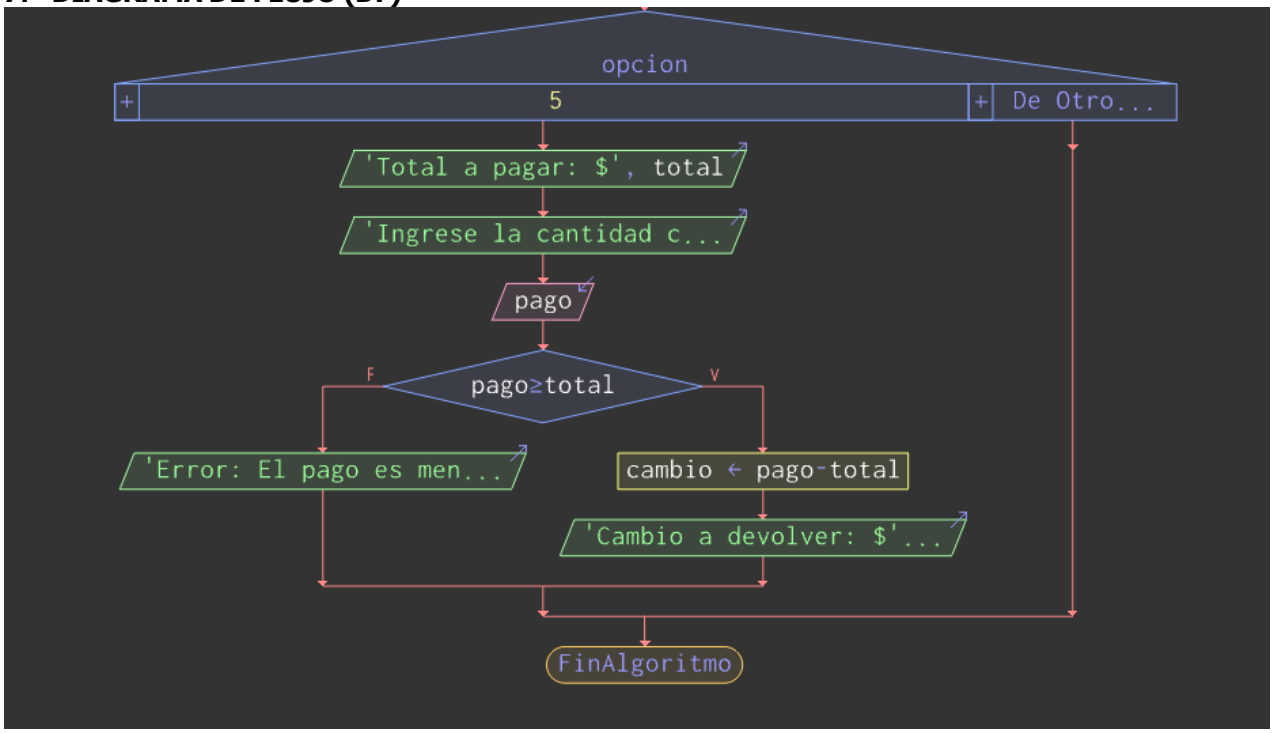
DONDE:
P: Número de nodos predicado
A: Número de aristas
N: Número de nodos

Prueba caja blanca de Operaciones

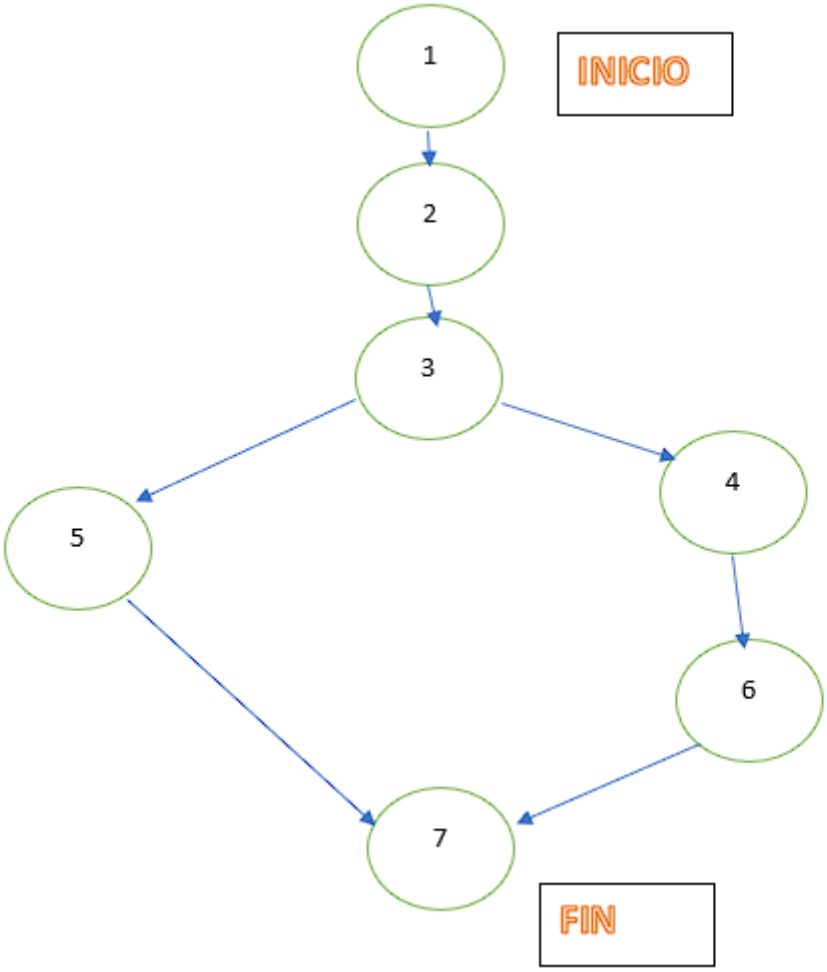
6. CÓDIGO FUENTE

```
// Ingresar pago y calcular cambio
void realizarPago() {
    float total = calcularTotal();
    if (total == 0) {
        cout << "No hay productos en el carrito." << endl;
        return;
    }

    float pago;
    cout << "Total a pagar: $" << fixed << setprecision(2) << total << endl;

    do {
        cout << "Ingrese la cantidad con la que paga: ";
        cin >> pago;
        if (pago < total) cout << "Error: El pago es menor al total." << endl;
    } while (pago < total);

    calcularCambio(pago);
    carrito.clear();
}
```

7. DIAGRAMA DE FLUJO (DF)



8. GRAFO DE FLUJO (GF)



9. IDENTIF IACCIÒN DE LAS RUTAS (Camino basico)

Determinar en base al GF del numeral 4
RUTAS

R1: 1-2-3-4-6-7
R2: 1-2-3-5-7

10. COMPLEJIDAD CICLOMÁTICA

Se puede calcular de las siguientes formas:

- V(G) = número de nodos predicados(decisiones)+1
  V(G)=1+1=2

- V(G) = A – N + 2
  V(G)= 7- 7+2=2

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

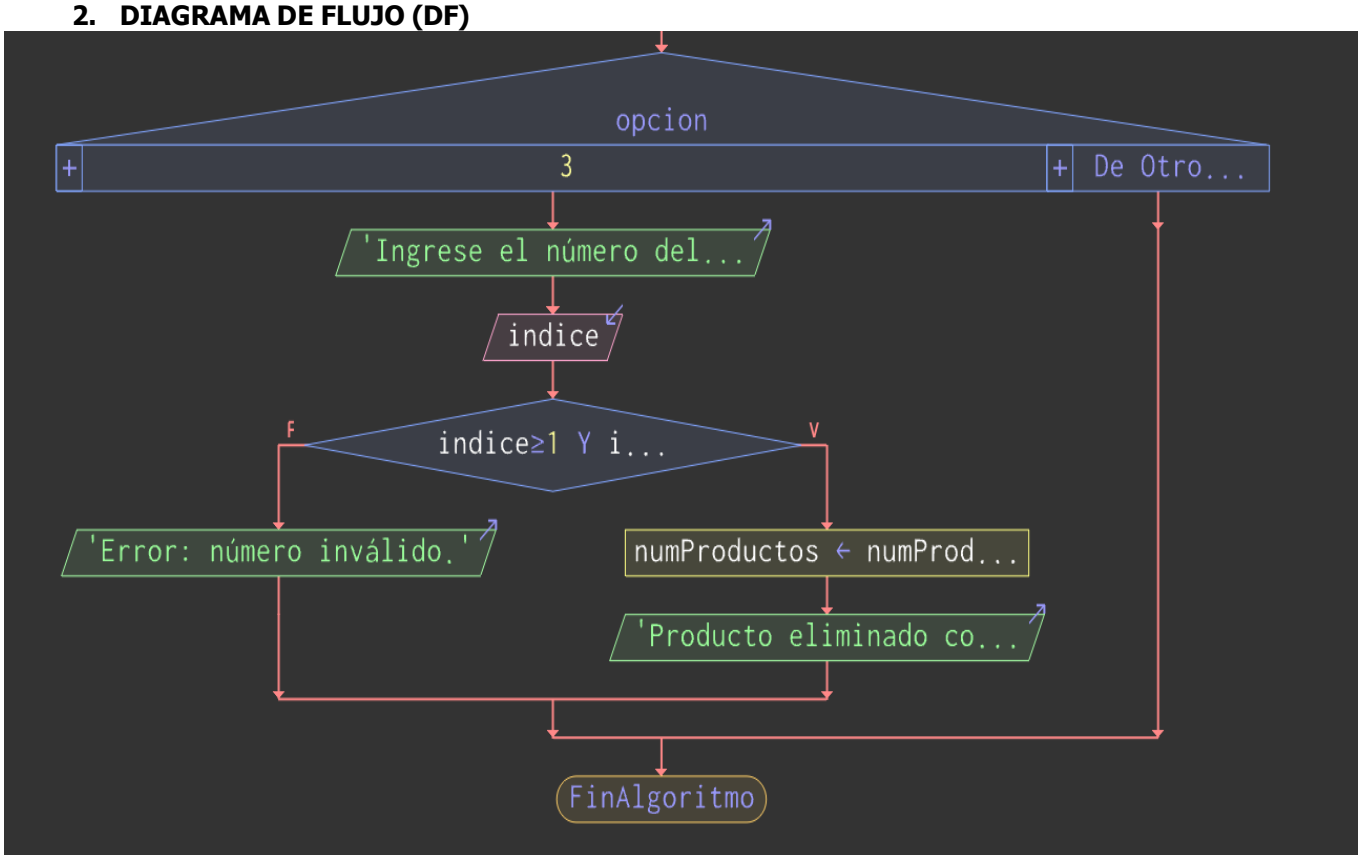
Prueba caja blanca de Modificación

1. **CÓDIGO FUENTE**

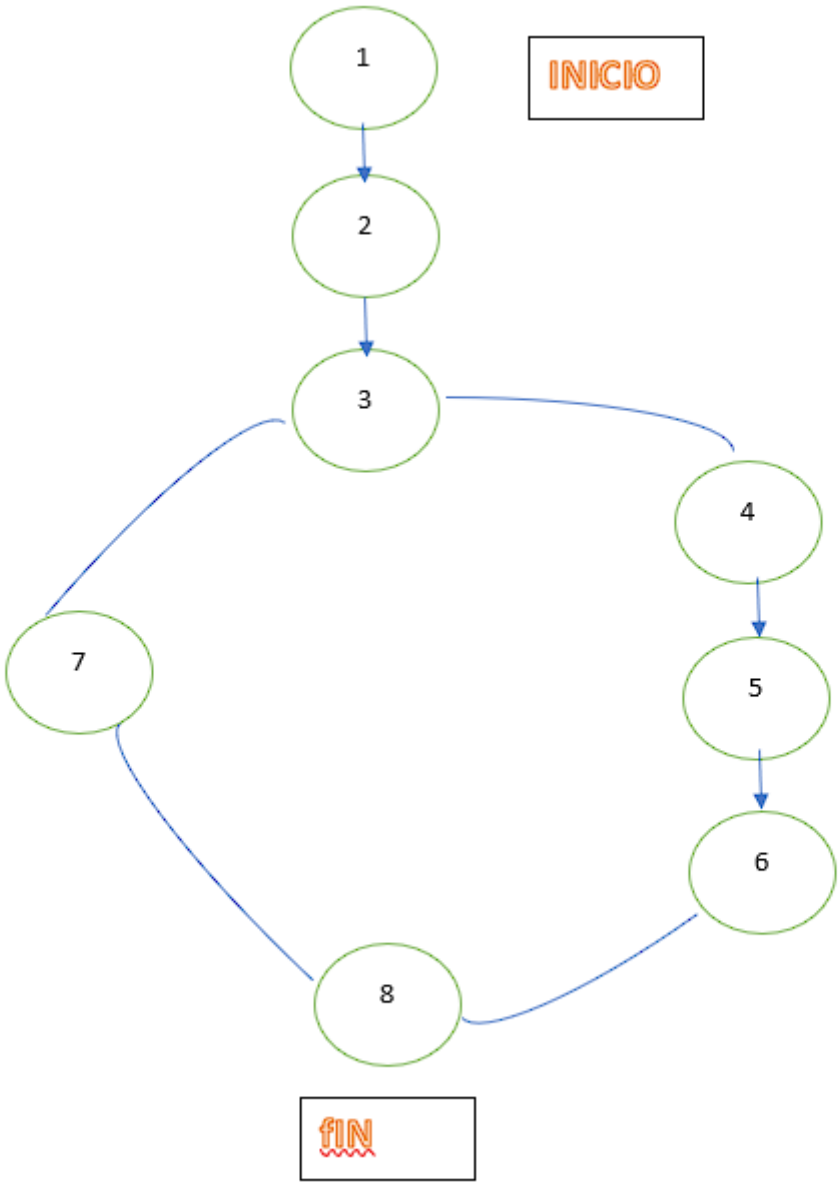
Eliminar productos

```
// Eliminar un producto
void eliminarProducto() {
    mostrarProductos(productos);
    if (productos.empty()) return;

    int indice;
    cout << "Ingrese el número del producto a eliminar: ";
    cin >> indice;
    if (indice < 1 || indice > productos.size()) {
        cout << "Error: número inválido." << endl;
        return;
    }
    productos.erase(productos.begin() + (indice - 1));
    cout << "Producto eliminado correctamente!" << endl;
}
```

2. **DIAGRAMA DE FLUJO (DF)**



3. GRAFO DE FLUJO (GF)



4. IDENTIF IACCIÒN DE LAS RUTAS (Camino basico)

Determinar en base al GF del numeral 4
RUTAS

R1: 1-2-3-4-5-6-8
R2: 1-2-3-7-8

5. COMPLEJIDAD CICLOMÁTICA

Se puede calcular de las siguientes formas:

- V(G) = número de nodos predicados(decisiones)+1
  V(G)=1+1=2

- V(G) = A – N + 2
  V(G)= 8- 8+2=2

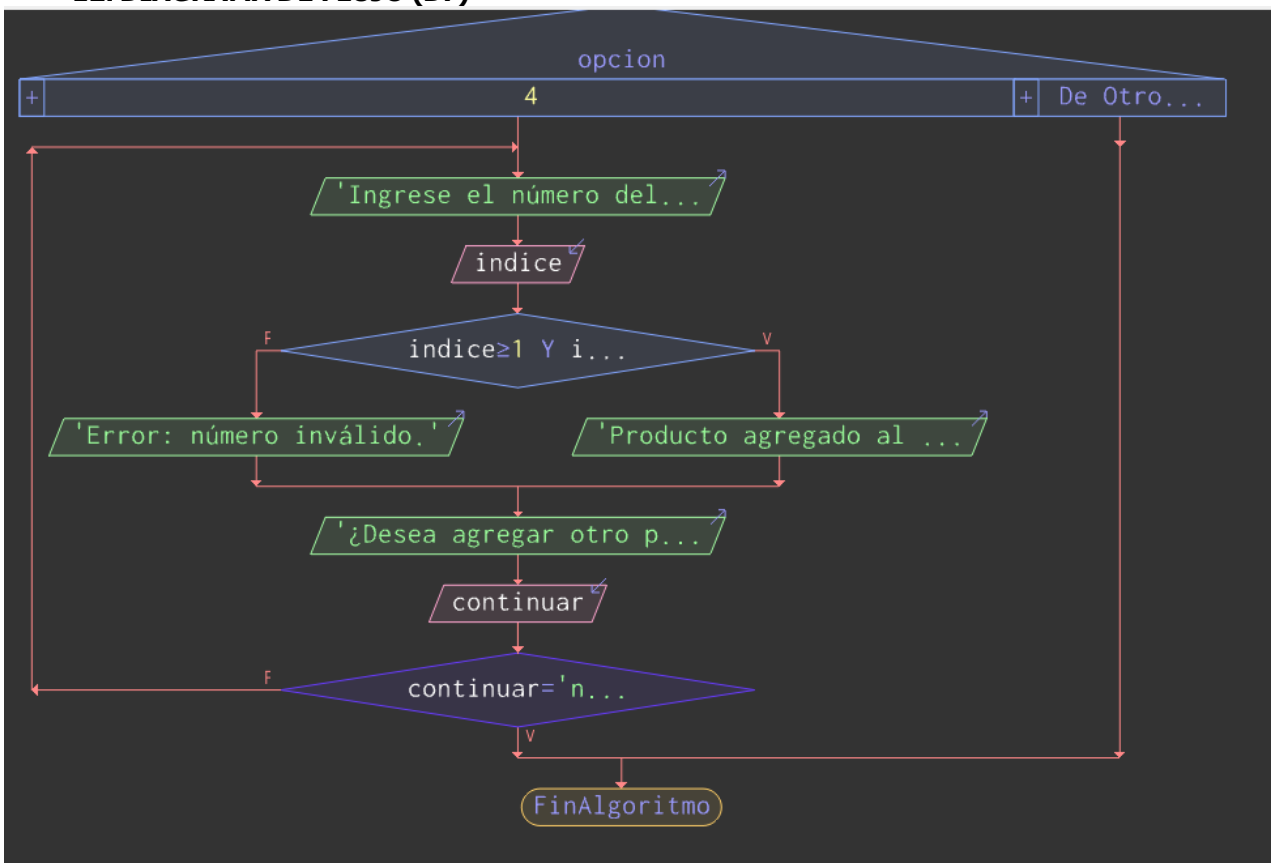
DONDE:
P: Número de nodos predicado
A: Número de aristas
N: Número de nodos

Prueba caja blanca de Guardar Datos

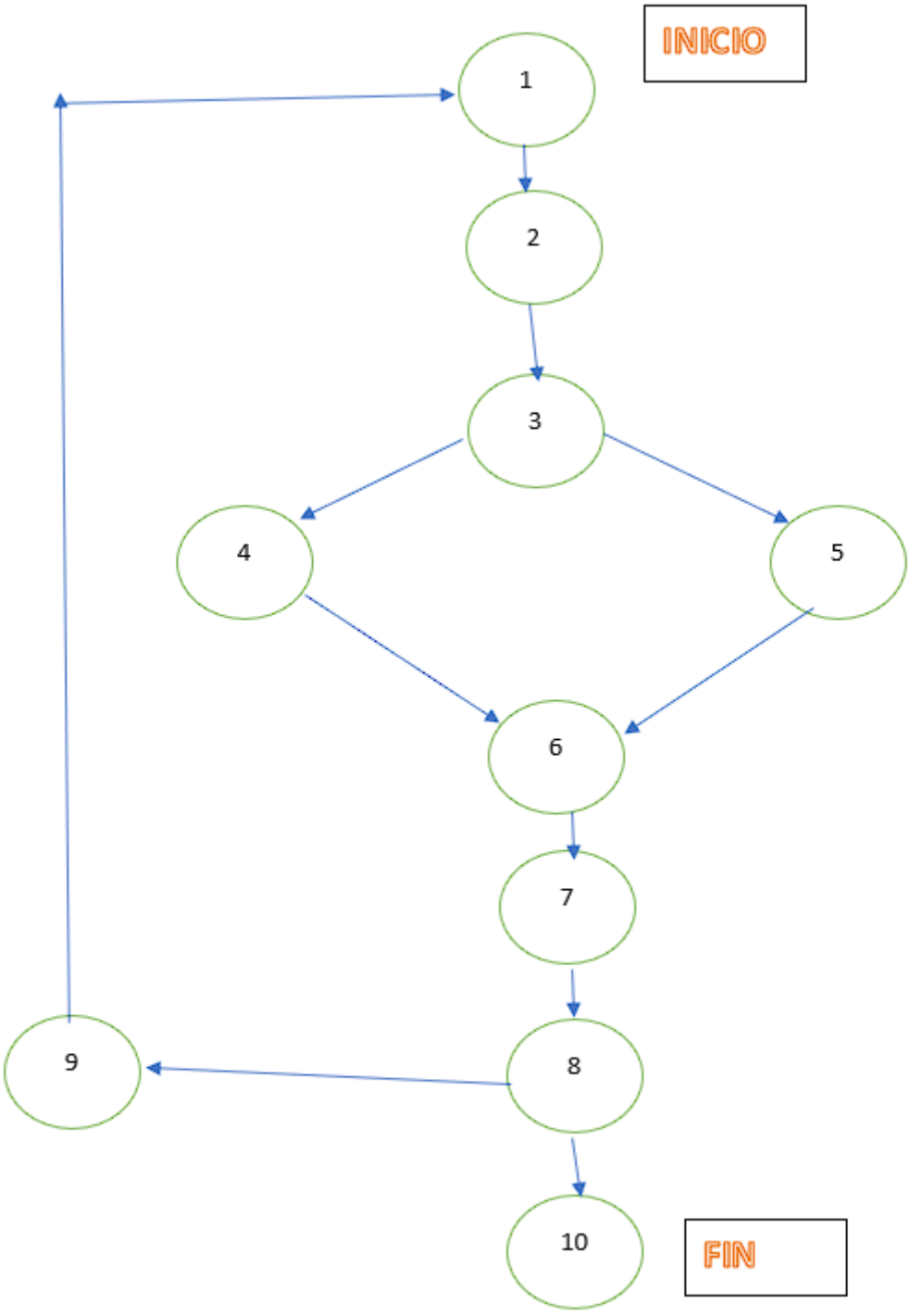
11. CÓDIGO FUENTE

```
// Seleccionar productos para la compra
void seleccionarCompra() {
    mostrarProductos(productos);
    if (productos.empty()) return;

    int indice;
    char continuar;
    do {
        cout << "Ingrese el número del producto que desea comprar: ";
        cin >> indice;
        if (indice < 1 || indice > productos.size()) {
            cout << "Error: número inválido." << endl;
        } else {
            carrito.push_back(productos[indice - 1]);
            cout << "Producto agregado al carrito." << endl;
        }
        cout << "¿Desea agregar otro producto? (s/n): ";
        cin >> continuar;
    } while (continuar == 's' || continuar == 'S');
}
```

12. DIAGRAMA DE FLUJO (DF)



13. GRAFO DE FLUJO (GF)



14. IDENTIF IACCIÒN DE LAS RUTAS (Camino basico)

Determinar en base al GF del numeral 4
RUTAS

R1: 1-2-3-4-6-7-8-10
R2: 1-2-3-5-6-7-8-10
R3: 1-2-3-4-6-7-8-9-10

15. COMPLEJIDAD CICLOMÁTICA

Se puede calcular de las siguientes formas:

- V(G) = número de nodos predicados(decisiones)+1
  V(G)=2+1=3

- V(G) = A – N + 2
  V(G)= 11- 10+2=3

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