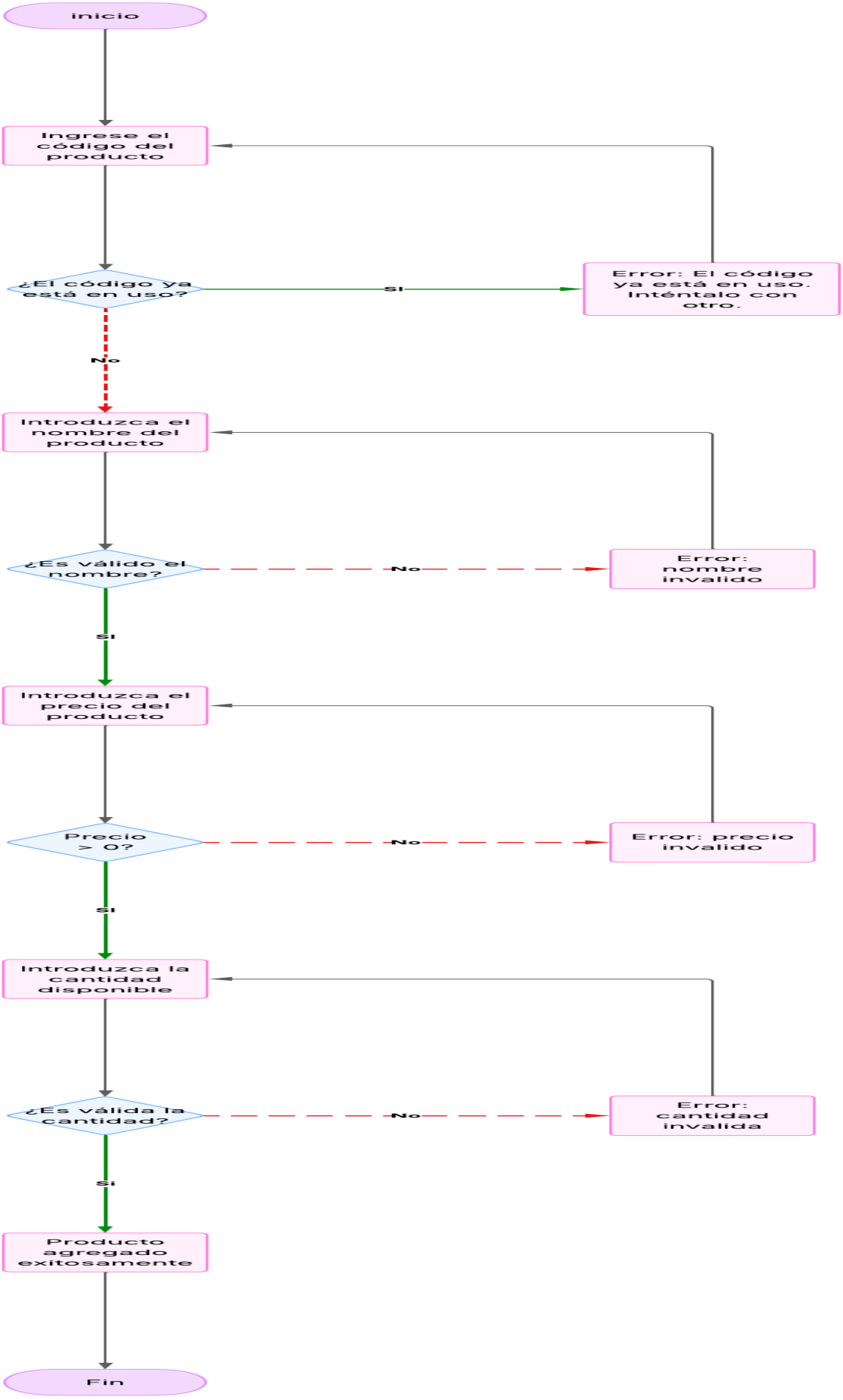
Prueba caja blanca de Requisito N° 1: El programa debe permitir agregar un nuevo producto al inventario. (nombre, precio y cantidad).

1. Código fuente

```
void crearProducto(Producto *p, Producto inventario[], int cantidadActual) {
    system("cls");
    while (1) {
        printf("Ingrese el codigo del producto: ");
        leerCadena(p->codigo, 20);
        if (buscarProducto(inventario, cantidadActual, p->codigo) != -1) {
            printf("Error: El codigo '%s' ya esta en uso. Intente con otro.\n", p->codigo);
        } else {
            break;
        }
    }
    printf("Ingrese el nombre del producto: ");
    leerCadena(p->nombre, 50);
    printf("Ingrese el precio del producto: ");
    p->precio = leerPrecio();
    printf("Ingrese la cantidad disponible: ");
    p->cantidad = leerCantidad();
    printf("Producto agregado correctamente. Presione ENTER para continuar...");
    getchar();
    system("cls");
     }
```

2. Diagrama de flujo



4. IDENTIFICACIÓN DE LAS RUTAS (Camino básico) RUTAS

- **R1:** 1-2-3-4-5-9
- **R2:** 1-2-2-3-4-5-9 *(reintento por código repetido)*
- **R3:** 1-2-3a-3-4-5-9 *(nombre inválido → reingresa)*
- **R4:** 1-2-3-4a-4-5-9 *(precio inválido → reingresa)*
- **R5:** 1-2-3-4-5a-5-9 *(cantidad inválida → reingresa)*

5. COMPLEJIDAD CICLOMÁTICA

Se puede calcular de las siguientes formas:

- **V(G) = número de nodos predicados(decisiones) + 1**
  V(G) = 4 + 1 = **5**
- **V(G) = A – N + 2**
  V(G) = 11 – 8 + 2 = **5**

DONDE:

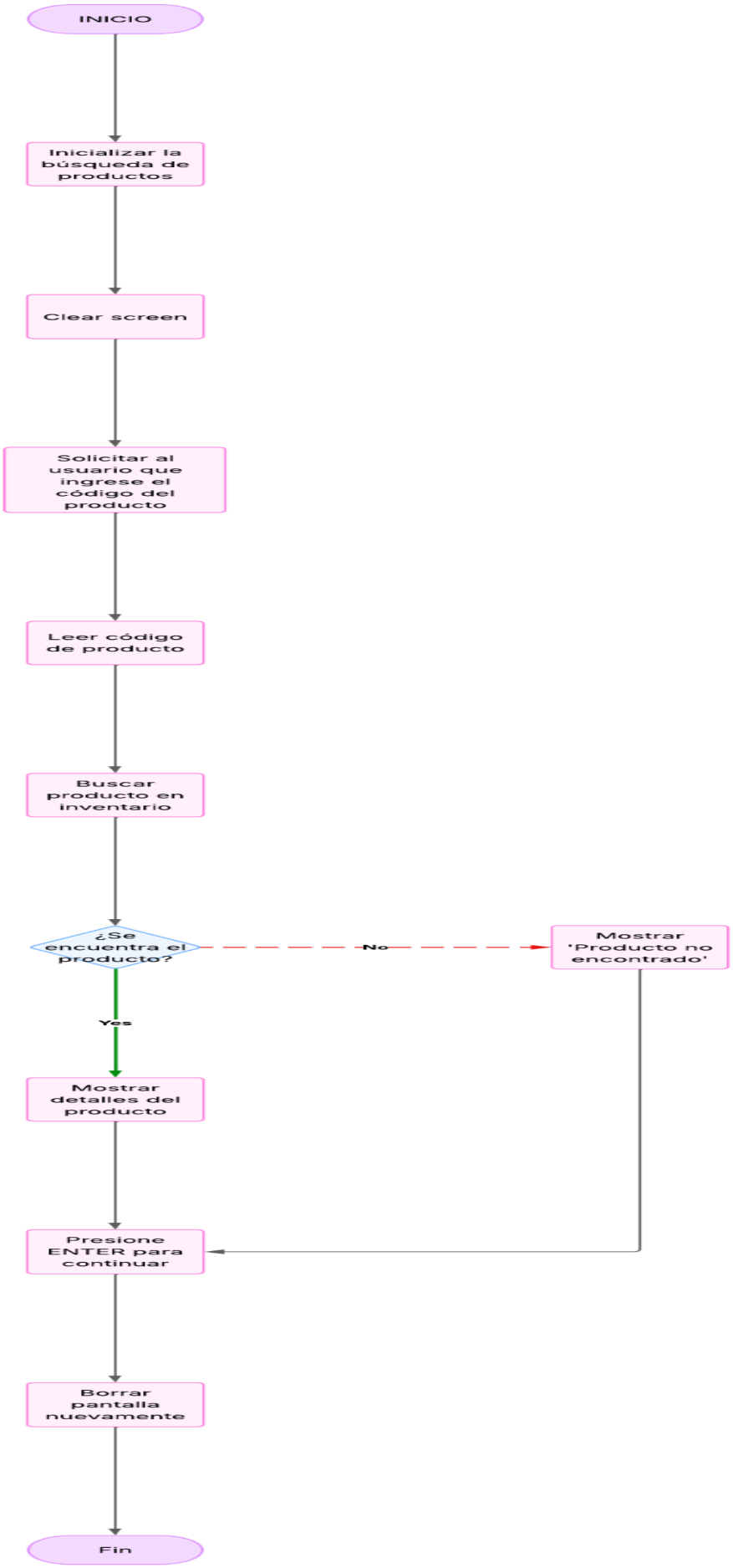
- **P**: Número de nodos predicado = 4
- **A**: Número de aristas = 11
- **N**: Número de nodos = 8

REQUISITO FUNCIONAL N.º 2: El programa debe permitir buscar un producto por código.

1. CODIGO FUENTE

```
void buscarProductoPorCodigo(Producto inventario[], int cantidad) {
    system("cls");
    char codigo[20];
    printf("Ingrese el codigo del producto a buscar: ");
    leerCadena(codigo, 20);

    int index = buscarProducto(inventario, cantidad, codigo);
    if (index == -1) {
        printf("Producto no encontrado.\n");
    } else {
        mostrarProducto(inventario[index]);
    }

    printf("\nPresione ENTER para continuar...");
    getchar();
    system("cls");
}
```

2. DIAGRMA DE FLUJO



4. IDENTIFICACIÓN DE LAS RUTAS (Camino básico) RUTAS

- **R1:** 1-2-3-5 *(producto encontrado)*
- **R2:** 1-2-4-5 *(producto no encontrado)*

5. COMPLEJIDAD CICLOMÁTICA

- **V(G) = número de nodos predicados + 1**
  V(G) = 1 + 1 = **2**
- **V(G) = A – N + 2**
  V(G) = 4 – 4 + 2 = **2**

DONDE:

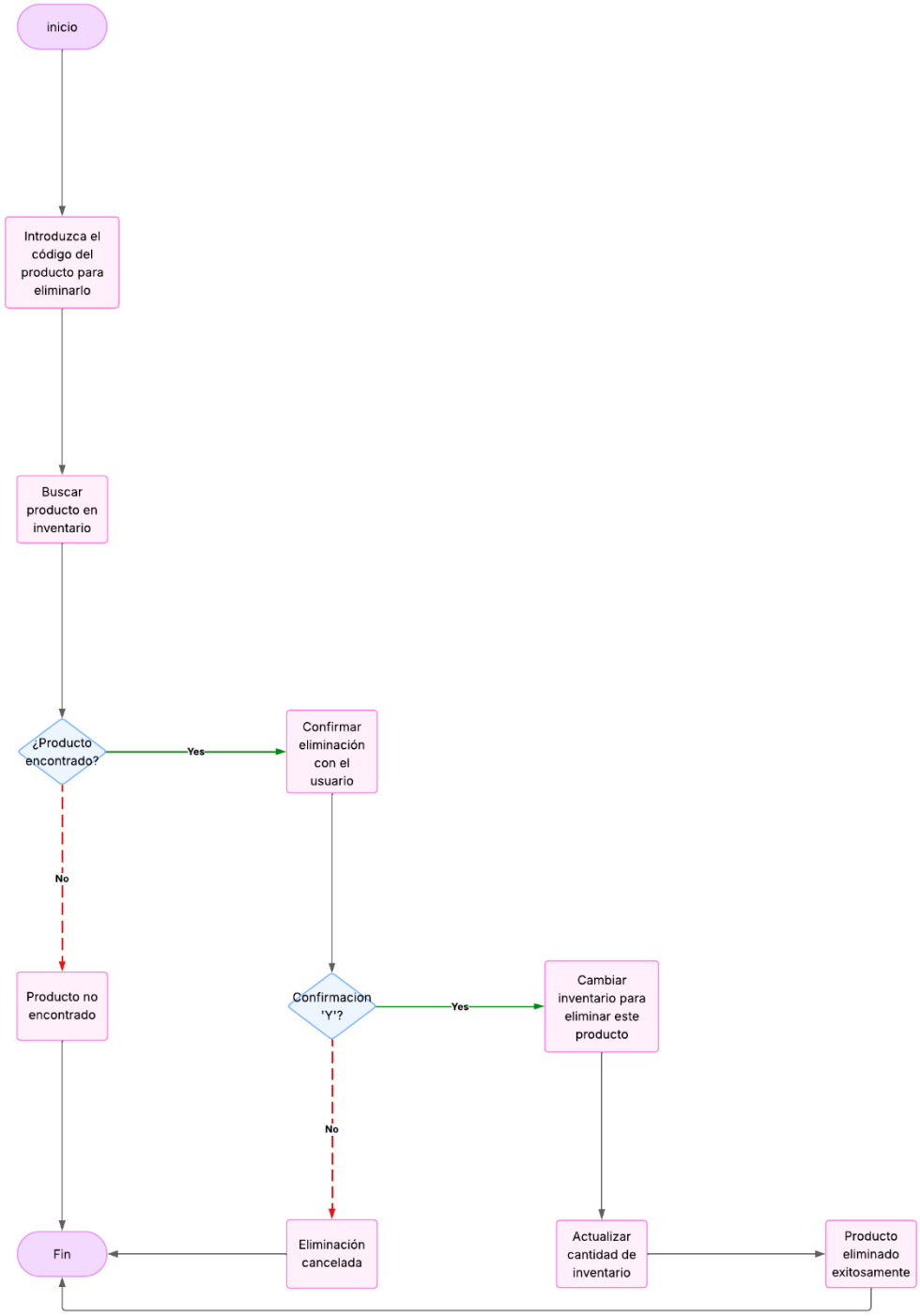
- **P**: 1
- **A**: 4
- **N**: 4

REQUISITO FUNCIONAL N.º 4: El programa debe permitir eliminar un producto existente del inventario.

1. **CODIGO FUENTE**

```
void eliminarProducto(Producto inventario[], int *cantidad) {
    system("cls");
    char codigo[20];
    printf("Ingrese el codigo del producto a eliminar: ");
    leerCadena(codigo, 20);

    int index = buscarProducto(inventario, *cantidad, codigo);
    if (index == -1) {
        printf("Producto no encontrado.\n");
        printf("Presione ENTER para continuar...");
        getchar();
        system("cls");
        return;
    }

    char confirmacion;
    printf("Seguro que desea eliminar el producto '%s'? (S/N): ", inventario[index].nombre);
    scanf(" %c", &confirmacion);
    getchar();

    if (confirmacion == 'S' || confirmacion == 's') {
        for (int i = index; i < *cantidad - 1; i++) {
            inventario[i] = inventario[i + 1];
        }
        (*cantidad)--;
        printf("Producto eliminado correctamente.\n");
    } else {
        printf("Eliminacion cancelada.\n");
    }

    printf("Presione ENTER para continuar...");
    getchar();
    system("cls");
}
```

2. DIAGRMA DE FLUJO



4. IDENTIFICACIÓN DE LAS RUTAS (Camino básico) RUTAS

- **R1: 1-2-3-7** *(producto no encontrado)*
- **R2: 1-2-4-5-7** *(producto encontrado, confirmado → eliminar)*
- **R3: 1-2-4-6-7** *(producto encontrado, no confirmado → cancelar)*

5. COMPLEJIDAD CICLOMÁTICA

- **V(G) = número de nodos predicados + 1**
  V(G) = 2 + 1 = 3
- **V(G) = A – N + 2**
  V(G) = 8 – 7 + 2 = 3

DONDE:

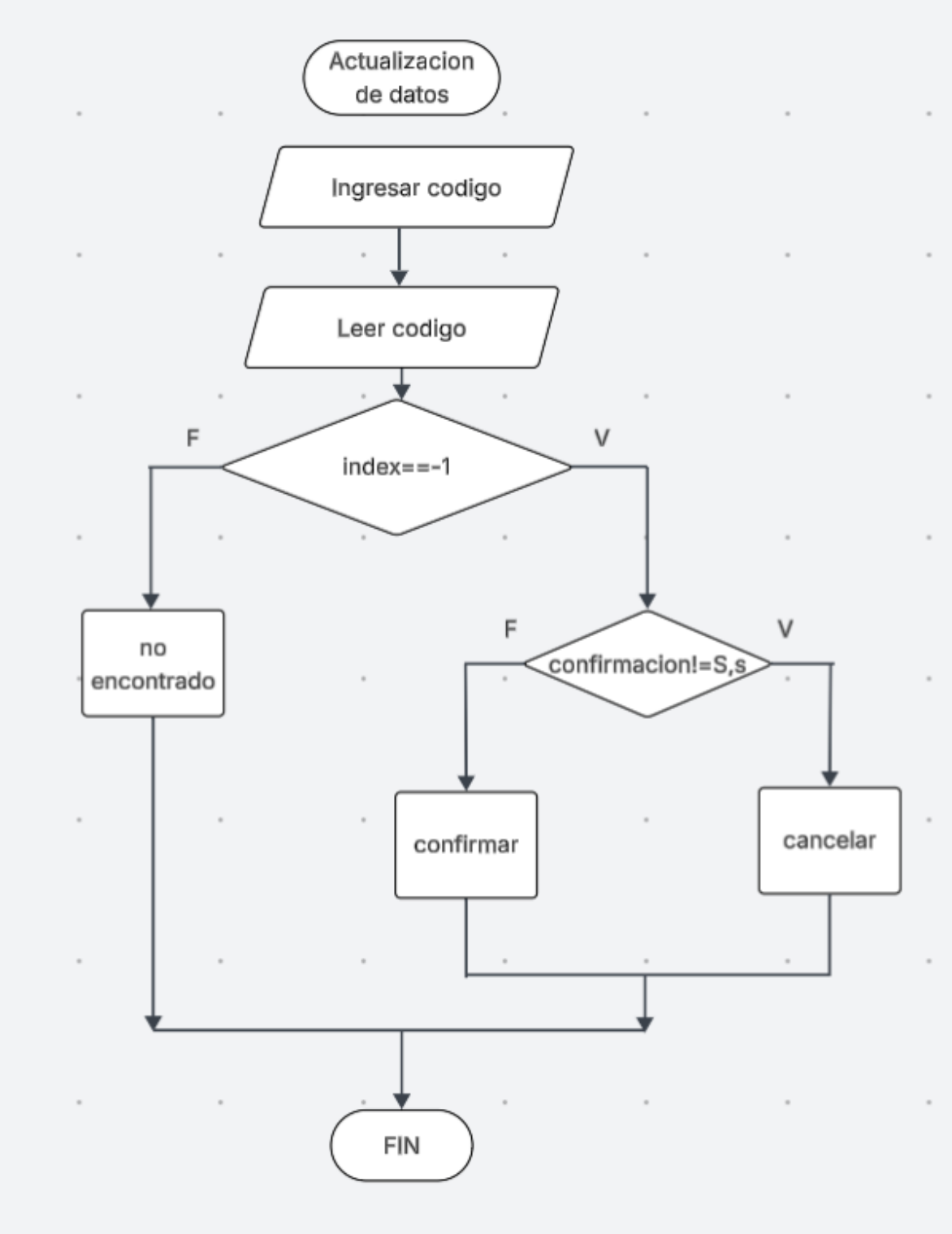
- **P: 2**
- **A: 8**
- **N: 7**

Prueba caja blanca de Requisito N° 5: El programa debe permitir actualizar la información de un producto existente (nombre, precio y cantidad).

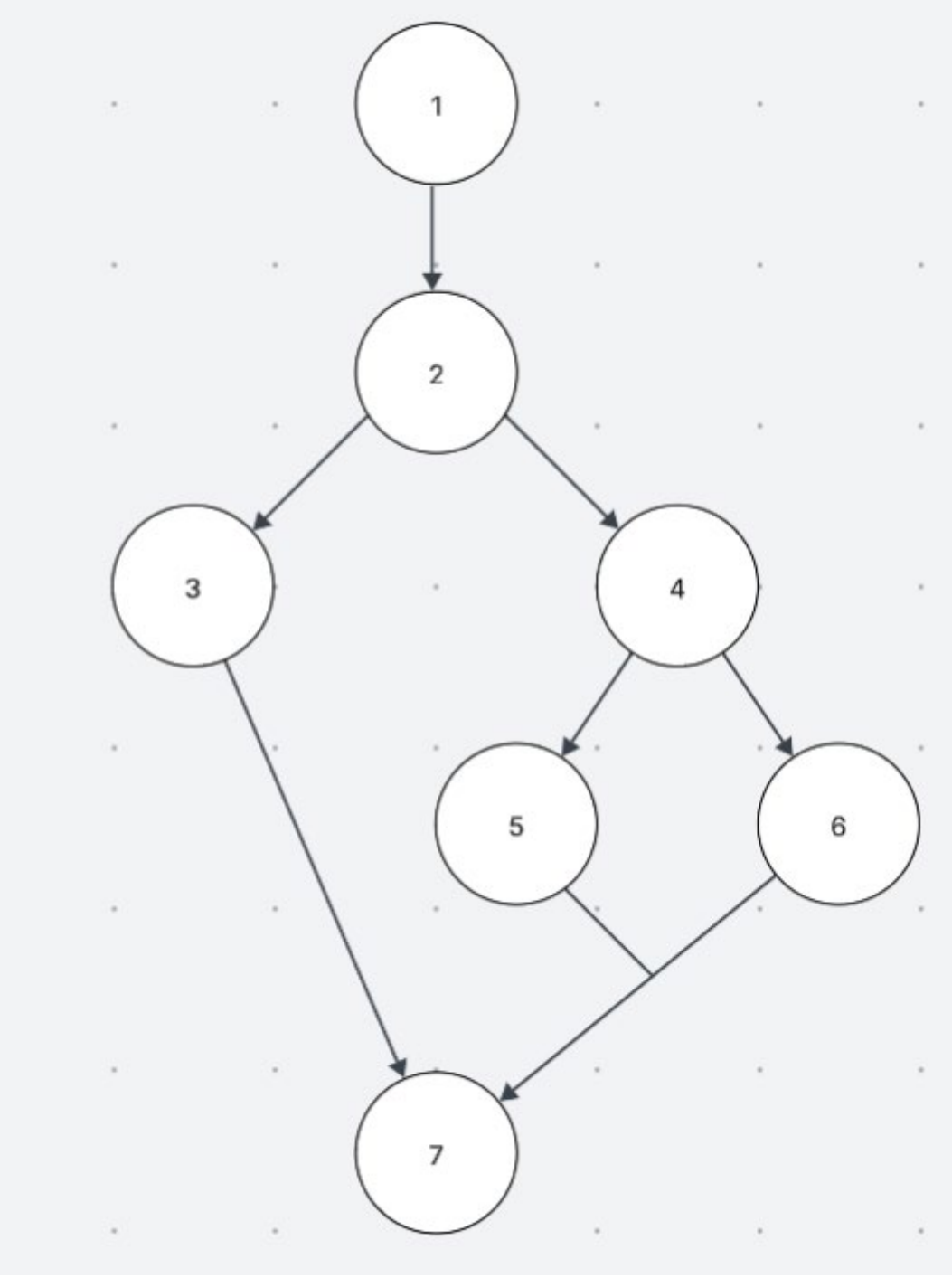
1. **CÓDIGO FUENTE**

```
int index = buscarProducto(inventario, cantidad, codigo);
if (index == -1) {
    printf("Producto no encontrado.\n");
    printf("Presione ENTER para continuar...");
    getchar();
    system("cls");
    return;
}

char confirmacion;
printf("Seguro que desea actualizar el producto '%s'? (S/N): ", inventario[index].nombre);
scanf(" %c", &confirmacion);
getchar();

if (confirmacion != 'S' && confirmacion != 's') {
    printf("Actualizacion cancelada.\n");
    printf("Presione ENTER para continuar...");
    getchar();
    system("cls");
    return;
}
```

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



3. GRAFO DE FLUJO (GF)



4. IDENTIFIACCIÒN DE LAS RUTAS (Camino básico)

RUTAS

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

5. 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)=8-7+2=3

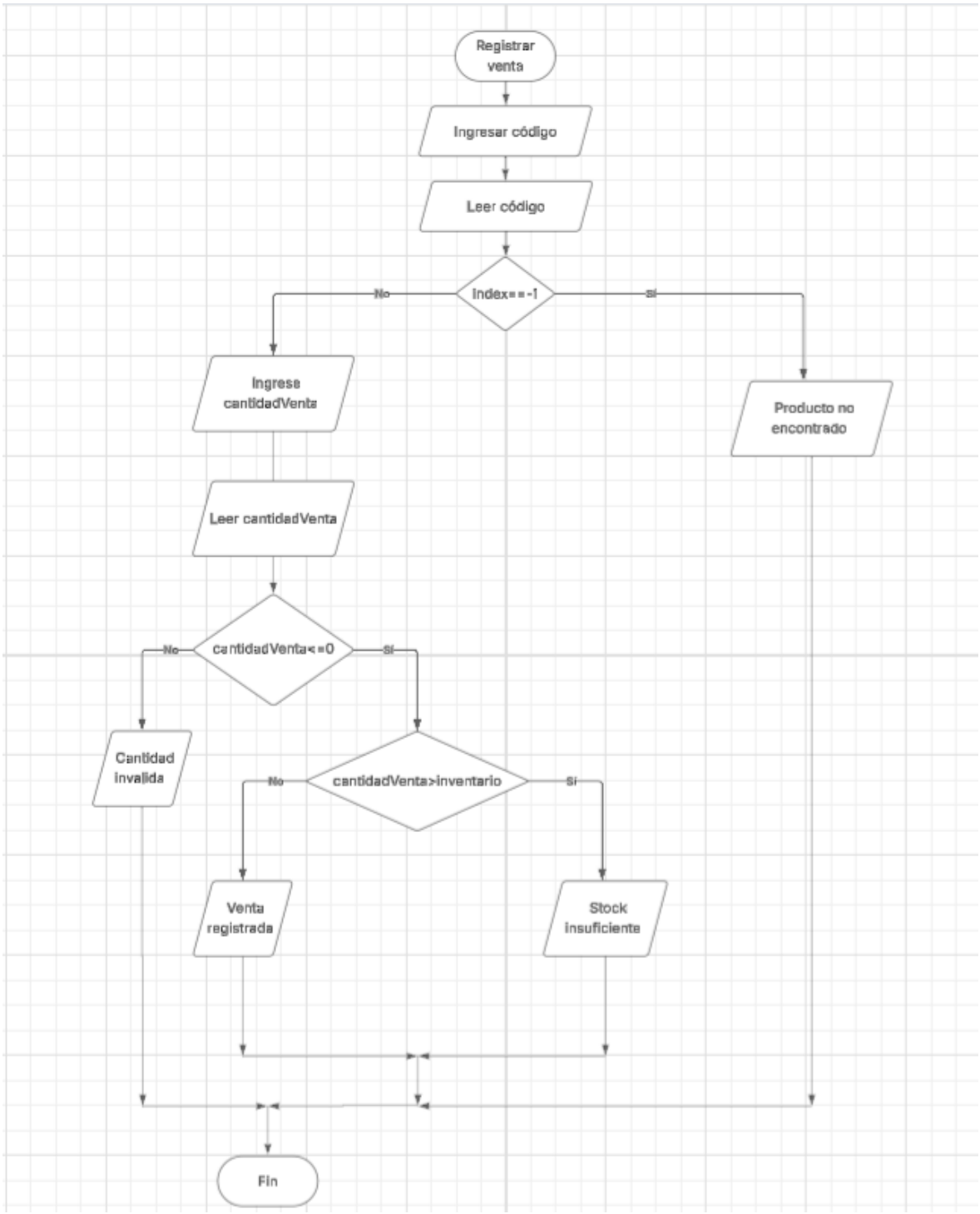
DONDE:
P: Número de nodos predicado
A: Número de aristas
N: Número de nodos

Prueba caja blanca de Requisito N° 6: El programa debe permitir realizar ventas de los productos existentes, actualizando la cantidad disponible al final de la transacción.

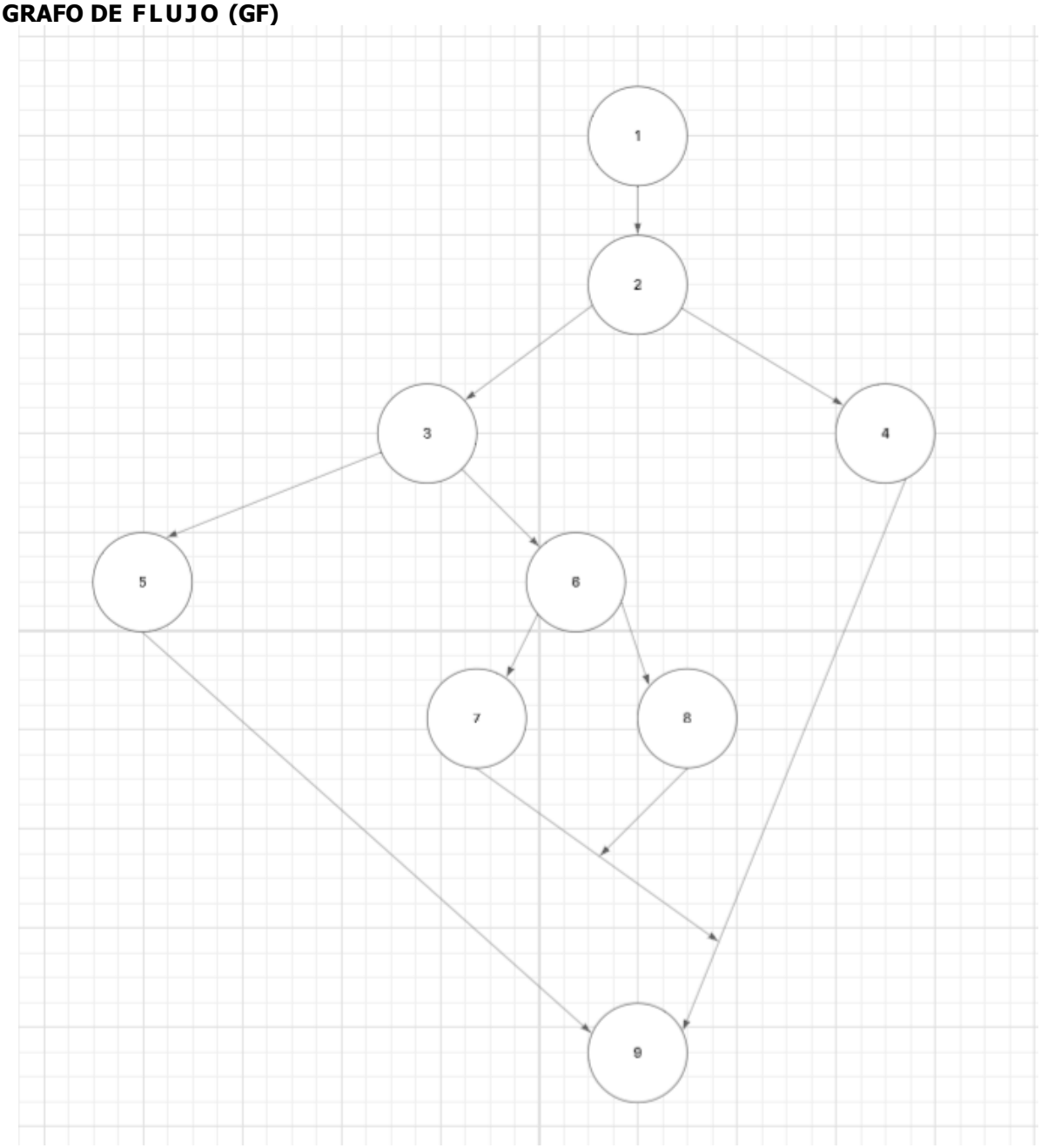
1. CÓDIGO FUENTE

```
int index = buscarProducto(inventario, *cantidad, codigo);
if (index == -1) {
    printf("Producto no encontrado.\n");
    printf("Presione ENTER para continuar...");
    getchar();
    system("cls");
    return;
}

// Mostrar informacion del producto encontrado
printf("\n--- INFORMACION DEL PRODUCTO ---\n");
printf("Codigo: %s\n", inventario[index].codigo);
printf("Nombre: %s\n", inventario[index].nombre);
printf("Precio Unitario: $%.2f\n", inventario[index].precio);
printf("Cantidad Disponible: %d\n\n", inventario[index].cantidad);

// Solicitar cantidad a vender
while (1) {
    printf("Ingrese la cantidad a vender: ");
    cantidadVenta = leerCantidad();

    if (cantidadVenta <= 0) {
        printf("Error: La cantidad debe ser mayor que cero.\n");
    } else if (cantidadVenta > inventario[index].cantidad) {
        printf("Error: No hay suficiente stock disponible.\n");
    } else {
        break;
    }
}
```

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



3. GRAFO DE FLUJO (GF)



4. IDENTIFIACCIÒN DE LAS RUTAS (Camino básico)

RUTAS

R1:1-2-3-5-9
R2:1-2-3-6-7-9
R3:1-2-3-6-8-9
R4:1-2-4-9

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)=11-9+2=4

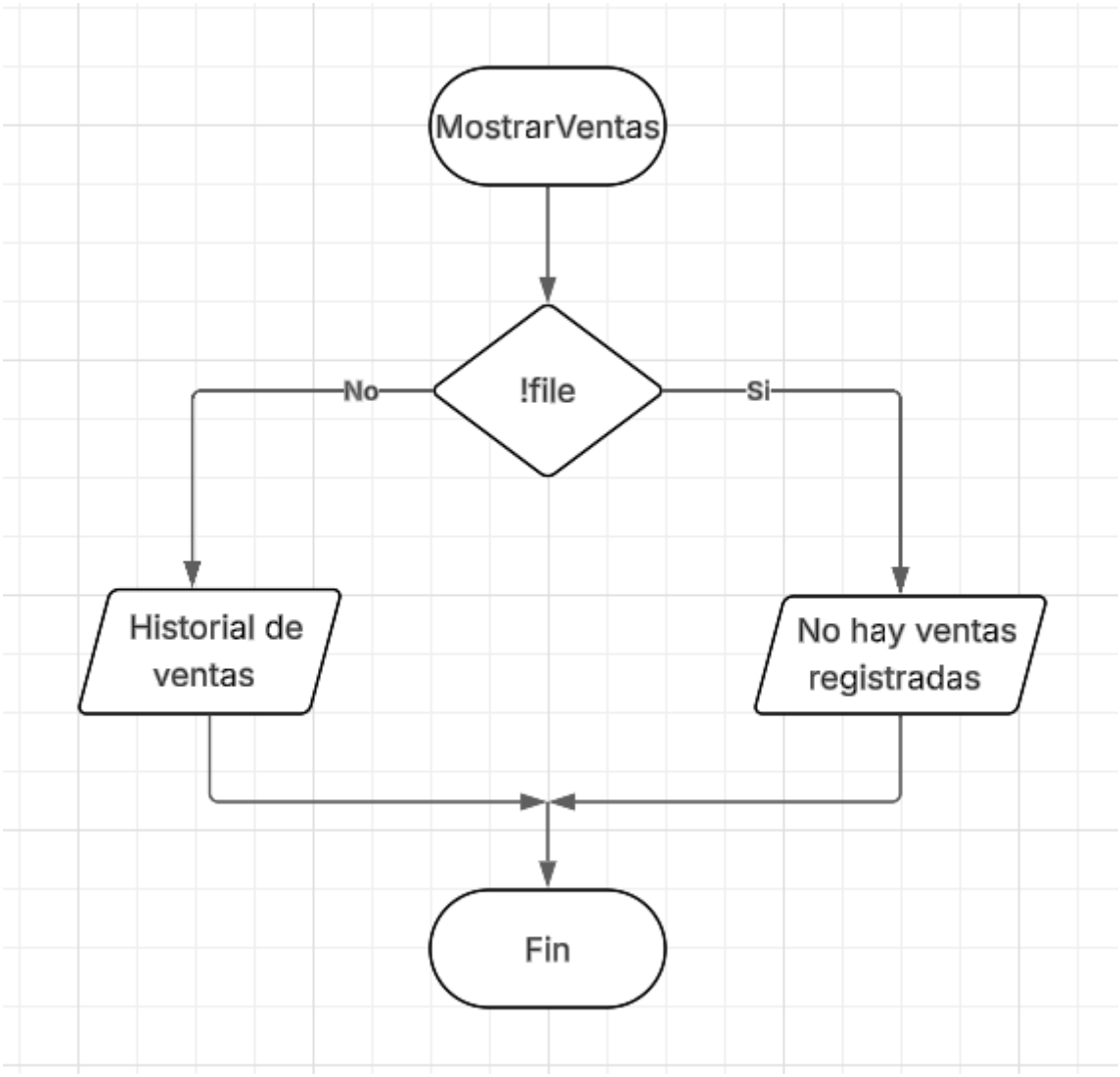
DONDE:
P: Número de nodos predicado
A: Número de aristas
N: Número de nodos

Prueba caja blanca de Requisito N7 El programa debe permitir visualizar las ventas registradas incluyendo la cantidad vendida, precio unitario y precio total

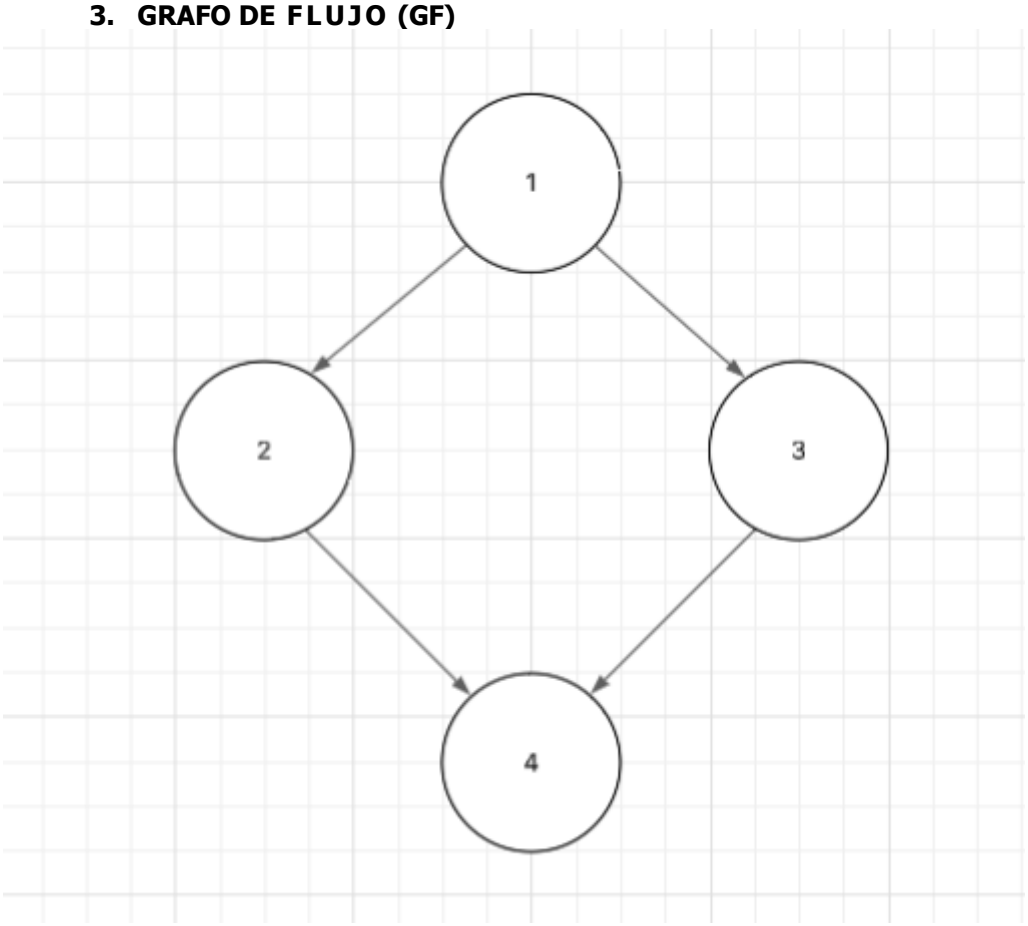
1. **CÓDIGO FUENTE**

```
void mostrarHistorialVentas() {
  system("cls");
  FILE *file = fopen(ARCHIVO_VENTAS, "r");
   if (!file) {
   printf("No hay ventas registradas aun.\n");
   printf("Presione ENTER para continuar...");
  getchar();
  system("cls");
  return;
  }
```

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



3. GRAFO DE FLUJO (GF)



4. IDENTIFIACCIÒN DE LAS RUTAS (Camino básico)

Rutas

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

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)=4-4+2=2

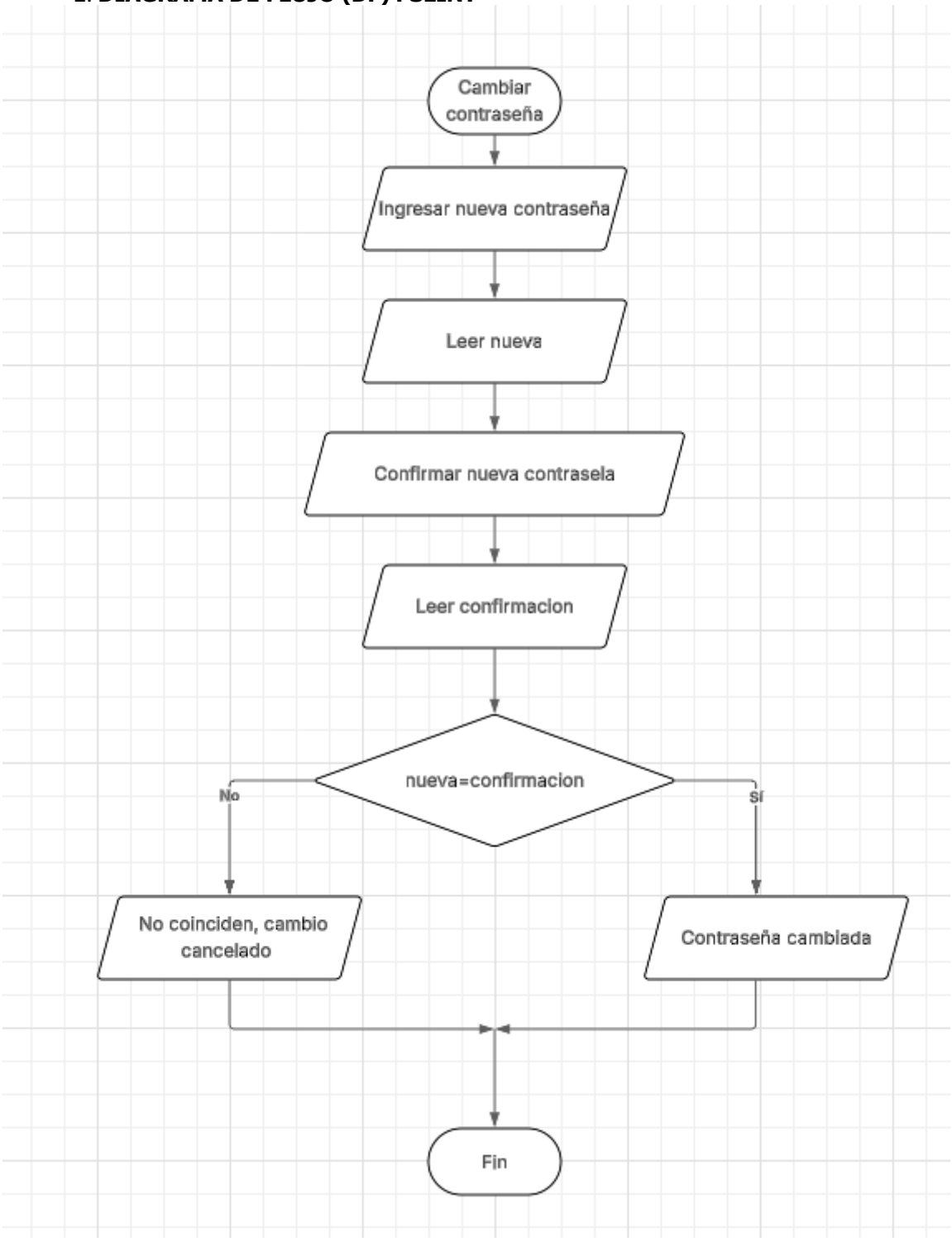
DONDE:
P: Número de nodos predicado
A: Número de aristas
N: Número de nodos

Prueba caja blanca de Requisito N8 El programa debe permitir que el usuario cambie su contraseña a voluntad

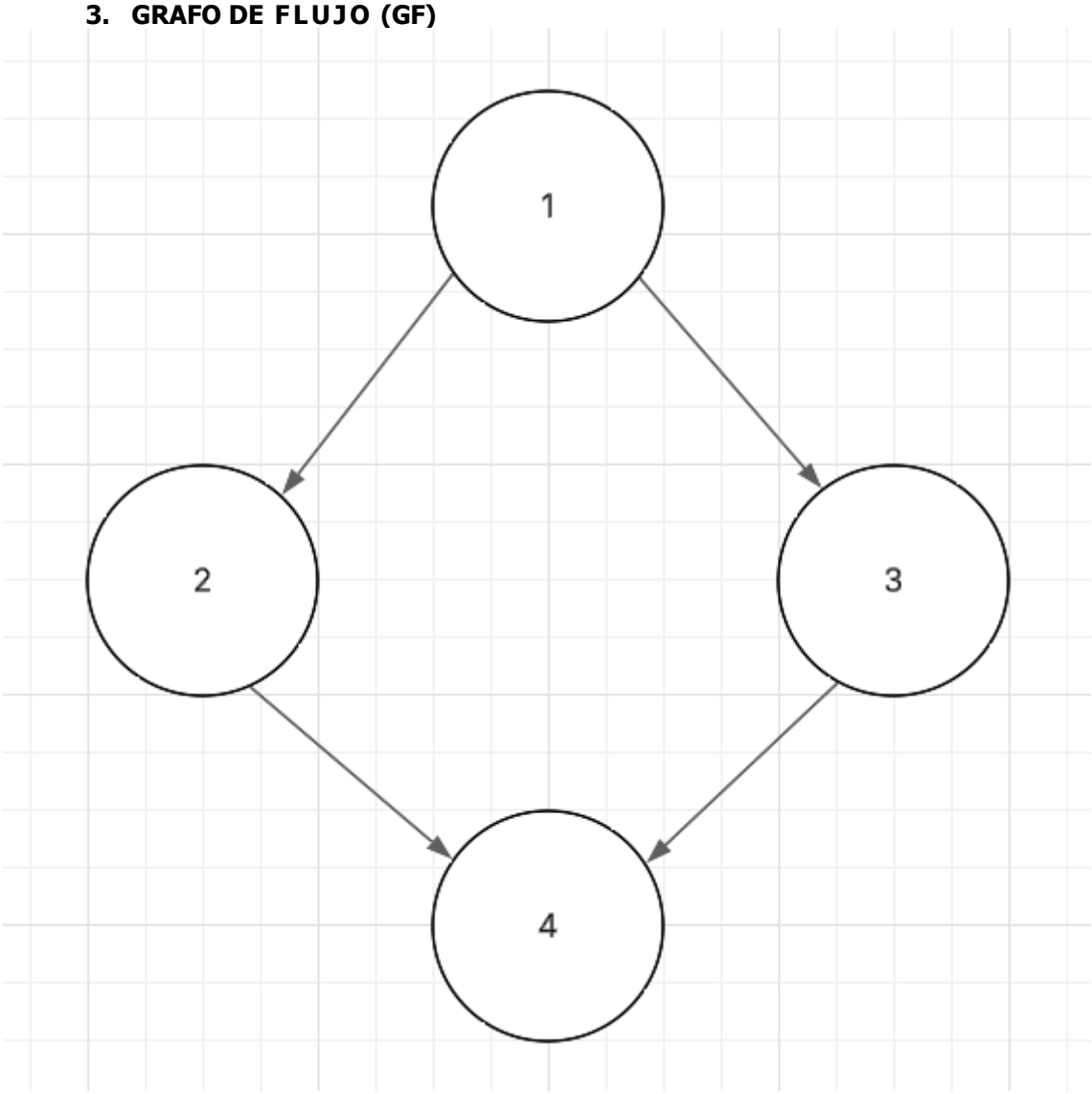
1. CÓDIGO FUENTE

```
    char nueva[50], confirmacion[50];
    printf("Ingrese la nueva contrasena: ");
    leerCadena(nueva, 50);

    printf("Confirme la nueva contrasena: ");
    leerCadena(confirmacion, 50);

    if (strcmp(nueva, confirmacion) == 0) {
        guardarContrasena(nueva);
        printf("Contrasena cambiada correctamente.\n");
    } else {
        printf("Las contrasenas no coinciden. No se hizo el cambio.\n");
    }

    printf("Presione ENTER para continuar...");
    getchar();
    system("cls");
}
```

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



3. GRAFO DE FLUJO (GF)



4. IDENTIFIACCIÒN DE LAS RUTAS (Camino básico)

Rutas

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

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)=4-4+2=2

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