Prueba de Caja Blanca

"Validacion de contraseña"

Integrantes:

Marco Chuquisala
Lendro Safla

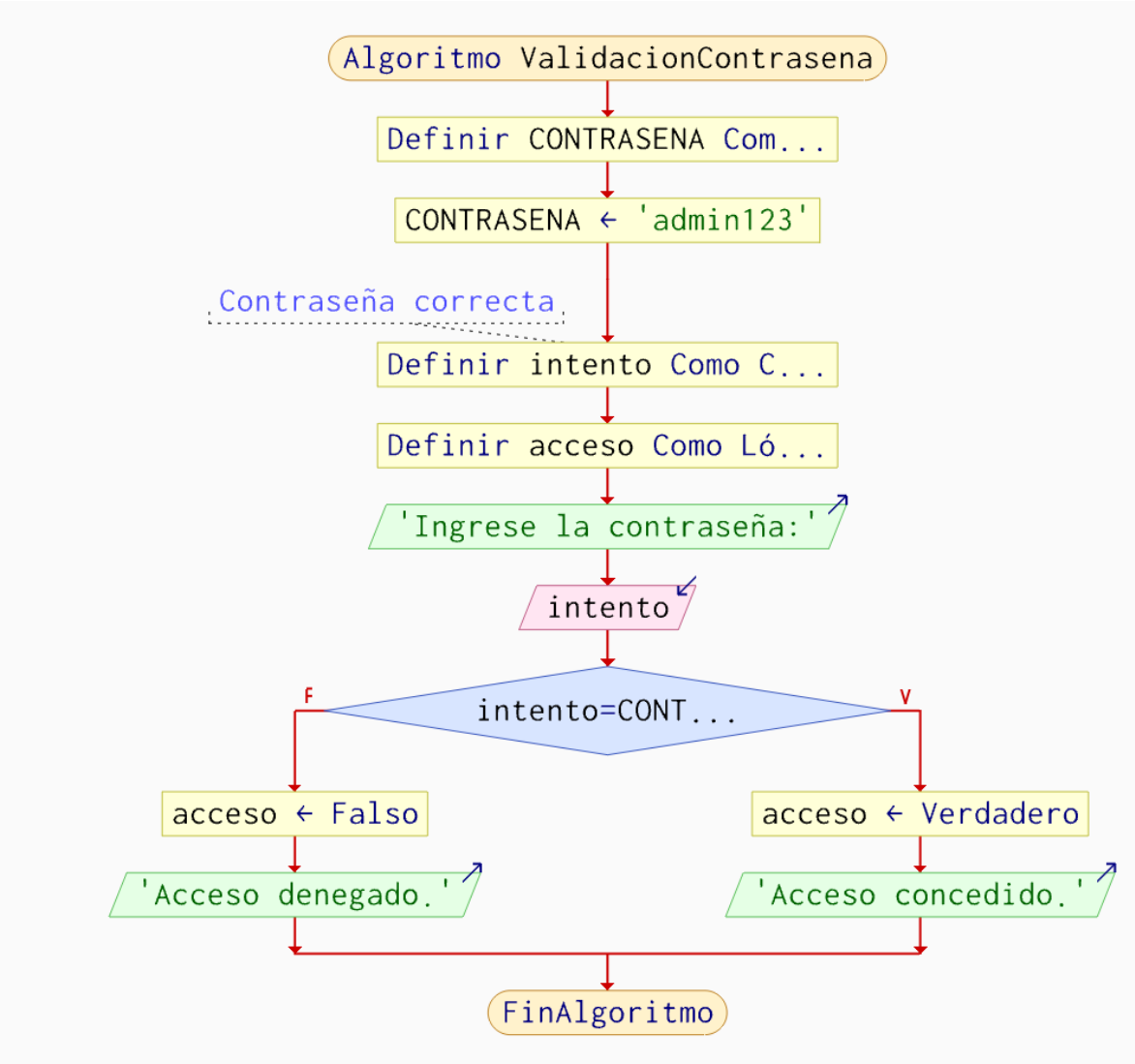
Fecha: 2025/07/23

Prueba caja blanca de Requisito N° 1: El código debe pedir una contraseña al iniciar el programa., el usuario debe ingresar la contraseña correcta (admin123) para acceder al menú.

1. **CÓDIGO FUENTE**

```
int verificarContrasena() {

   char ingreso[20];

   printf("Ingrese la contrasena para acceder al sistema: ");

   fgets(ingreso, 20, stdin);

   ingreso[strcspn(ingreso, "\n")] = '\0';



   if (strcmp(ingreso, CONTRASENA) == 0) {

      return 1; // Contraseña correcta

   } else {

      return 0; // Contraseña incorrecta

   }

}
int main() {
   if (!verificarContrasena()) {
      printf("Contrasena incorrecta. Acceso denegado.\n");
      return 1;
   }
```

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



3. GRAFO DE FLUJO (GF)

1. • Leer ingreso
2. • Limpiar salto de línea
3. • Comparación `strcmp(...) == 0`
4. • Return 1 (contraseña correcta)
5. • Return 0 (contraseña incorrecta)

4. IDENTIFIACCIÒN DE LAS RUTAS (Camino básico)

1. • 1 → 2
2. • 2 → 3
3. • 3 → 4 (si verdadero)
4. • 3 → 5 (si falso)

RUTAS

R1: Ingresar contraseña-Comparar-Correcta-Ingresar al sistema
R2: Ingresar contraseña-Comparar-Incorrecta-Denegar y finalizar

5. COMPLEJIDAD CICLOMÁTICA

V(G) = P + 1 = 1 + 1 = 2

Prueba de Caja Blanca

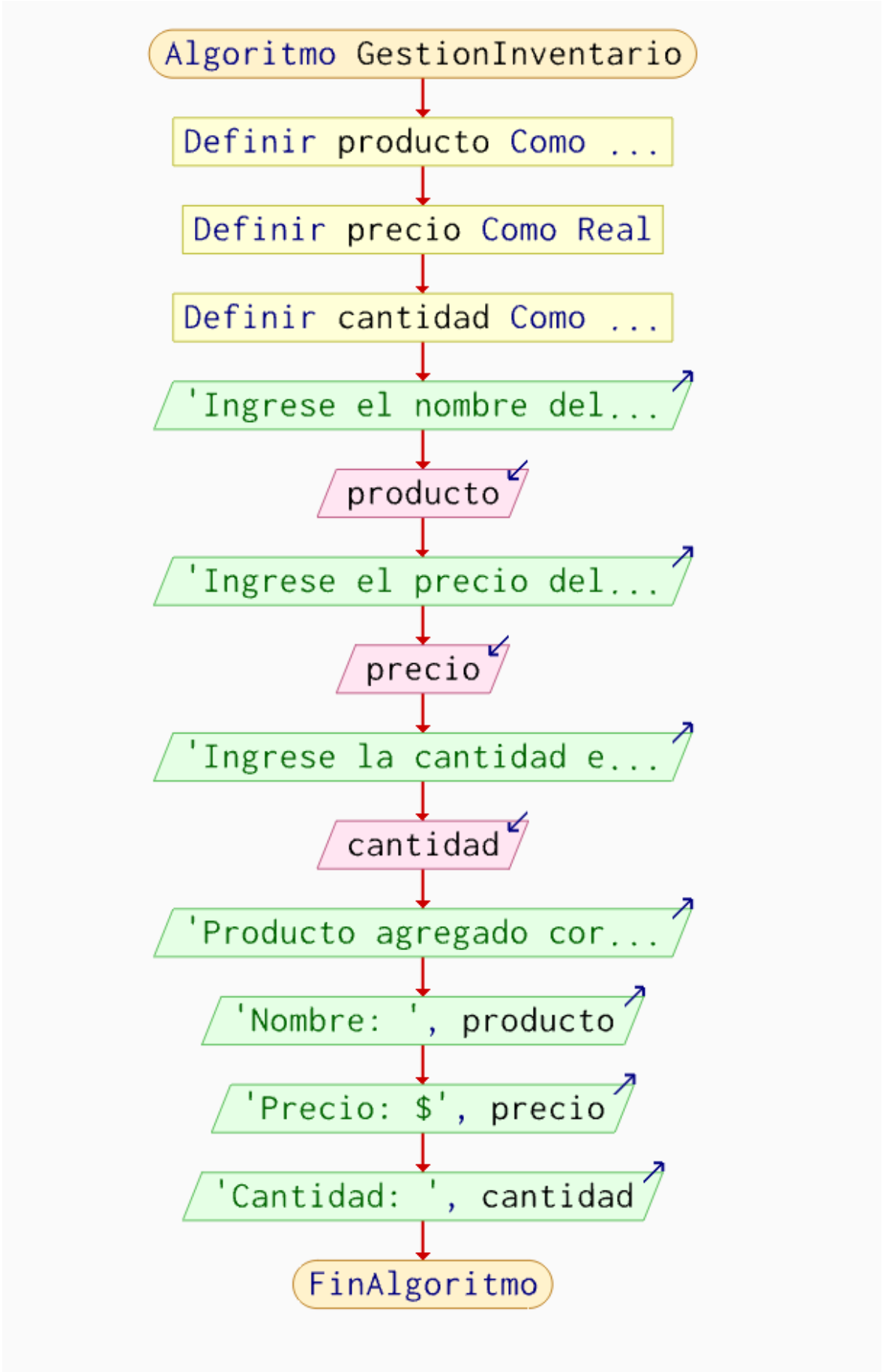
"Registro del producto"

Prueba caja blanca de Requisito N° 2: El código debe permitir al usuario registrar un nuevo producto esto incluye, código único del producto, nombre del producto, precio, cantidad disponible.

1. **CÓDIGO FUENTE**

```
void crearProducto(Producto *p) {
    printf("Ingrese el codigo del producto: ");
    fgets(p->codigo, 20, stdin);
    p->codigo[strcspn(p->codigo, "\n")] = '\0';

    printf("Ingrese el nombre del producto: ");
    fgets(p->nombre, 50, stdin);
    p->nombre[strcspn(p->nombre, "\n")] = '\0';

    printf("Ingrese el precio del producto: ");
    scanf("%f", &p->precio);

    printf("Ingrese la cantidad disponible: ");
    scanf("%d", &p->cantidad);

    getchar(); // Limpia el buffer del teclado
}
```

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



3. GRAFO DE FLUJO (GF)

1. • Leer código del producto
2. • Quitar salto de línea del código
3. • Leer nombre del producto
4. • Quitar salto de línea del nombre
5. • Leer precio
6. • Leer cantidad
7. • Limpiar buffer

4. IDENTIFIACCIÒN DE LAS RUTAS (Camino básico)

RUTAS

R1: Leer código → quitar salto → leer nombre → quitar salto → leer precio → leer cantidad → limpiar buffer

5. COMPLEJIDAD CICLOMÁTICA

V(G) = 0 + 1 = 1

Prueba de Caja Blanca

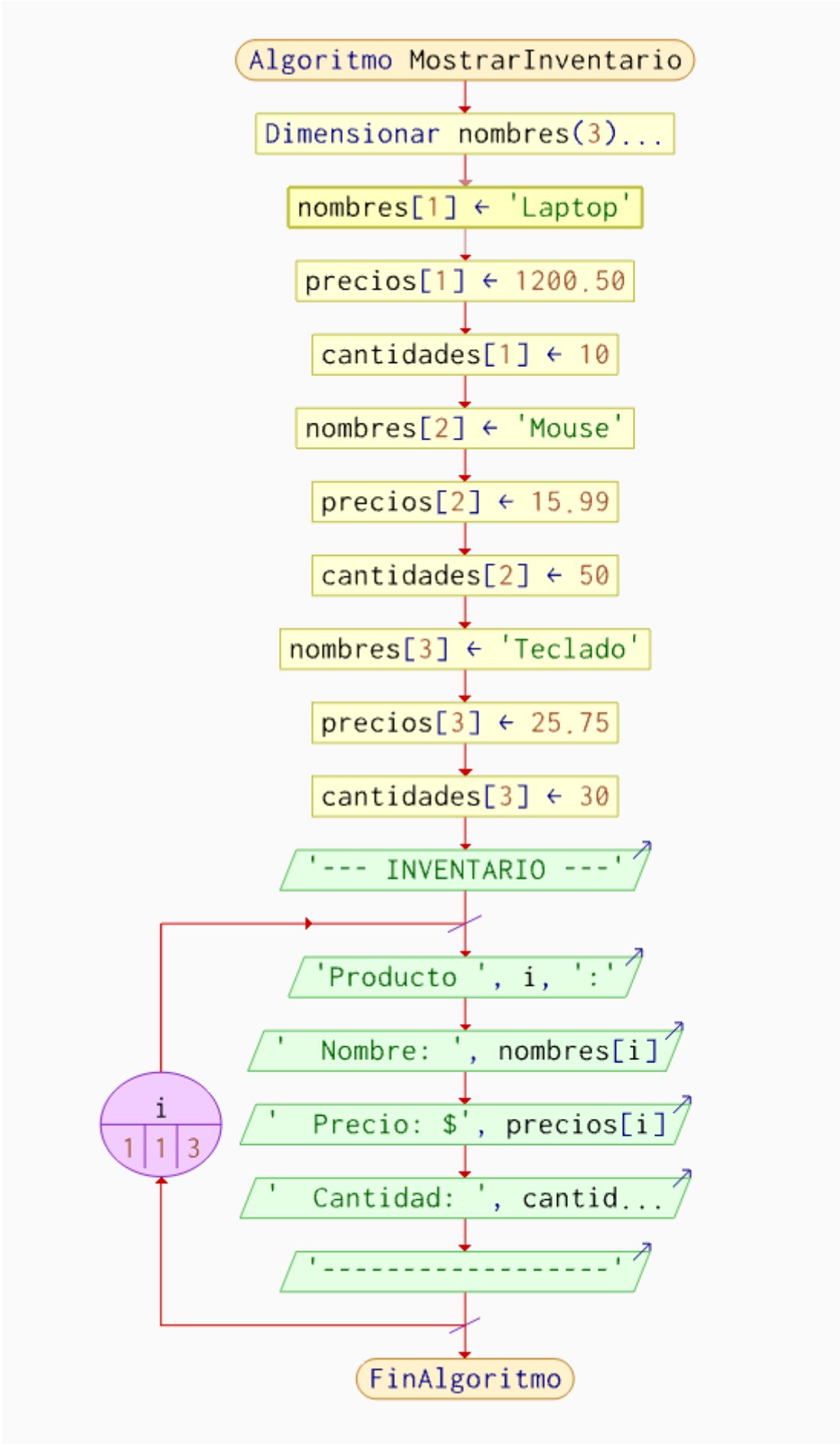
"Visualización de productos"

Prueba caja blanca de Requisito N° 3: El código debe permitir al usuario ver todos los productos registrados en el inventario

1. **CÓDIGO FUENTE**

```
case 2:

    if (cantidad == 0) {

        printf("No hay productos registrados aun.\n");

    } else {

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

            mostrarProducto(inventario[i]);

        }

    }

    break;
void mostrarProducto(Producto p) {
    printf("\nCodigo: %s\n", p.codigo);
    printf("Nombre: %s\n", p.nombre);
    printf("Precio: $%.2f\n", p.precio);
    printf("Cantidad disponible: %d\n", p.cantidad);
}
```

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



3. GRAFO DE FLUJO (GF)

Nodos:

1. Verificar si cantidad == 0
2. Mostrar "No hay productos..."
3. Inicializar i = 0
4. Comparar i < cantidad
5. Llamar a mostrarProducto()
6. i++
7. Fin

Aristas:

- 1 - 2 (si cantidad == 0)
- 1 - 3 (si cantidad > 0)
- 3 - 4
- 4 - 5 (si condici ∫n verdadera)
- 5 - 6
- 6 - 4 (bucle)
- 4 - 7 (cuando i >= cantidad)
- 2 - 7

4. IDENTIFIACCIÒN DE LAS RUTAS (Camino básico)

RUTAS

R1: cantidad == 0 → mostrar mensaje

R2: cantidad > 0 → entrar al ciclo → mostrar todos los productos → salir

5. COMPLEJIDAD CICLOMÁTICA

V(G) = 2 + 1 = 3

Prueba de Caja Blanca

"RF4, RF5, RF6: Actualizar, Eliminar, Salir del sistema de inventario"

Integrantes:

Fecha: 2025/07/23

Prueba caja blanca de Requisito N° 1: Modificar, eliminar y salir del sistema

CÓDIGO FUENTE

```
Proceso ActualizarProducto
  Escribir "Ingrese el código del producto a actualizar: "
  Leer codigo
  encontrado ← FALSO
  Para i ← 0 Hasta cantidad - 1 Con Paso 1
    Si inventario[i].codigo = codigo Entonces
      encontrado ← VERDADERO
      Escribir "Ingrese nuevo nombre: "
      Leer inventario[i].nombre
      Escribir "Ingrese nuevo precio: "
      Leer inventario[i].precio
      Escribir "Ingrese nueva cantidad: "
      Leer inventario[i].cantidad
      Escribir "Producto actualizado correctamente."
    FinSi
  FinPara
  Si encontrado = FALSO Entonces
    Escribir "Producto no encontrado."
  FinSi
FinProceso


Proceso EliminarProducto
  Escribir "Ingrese el código del producto a eliminar: "
  Leer codigo
  encontrado ← FALSO
  Para i ← 0 Hasta cantidad - 1 Con Paso 1
    Si inventario[i].codigo = codigo Entonces
```

```
        encontrado ← VERDADERO
        Para j ← i Hasta cantidad - 2 Con Paso 1
          inventario[j] ← inventario[j + 1]
        FinPara
        cantidad ← cantidad - 1
        Escribir "Producto eliminado correctamente."
        Salir
      FinSi
    FinPara
    Si encontrado = FALSO Entonces
      Escribir "Producto no encontrado."
    FinSi
FinProceso


Proceso Menu
    Repetir
      Escribir "1. Agregar producto"
      Escribir "2. Mostrar productos"
      Escribir "3. Eliminar producto"
      Escribir "4. Actualizar producto"
      Escribir "5. Salir"
      Leer opcion
      Segun opcion Hacer
        1: Llamar CrearProducto
        2: Llamar MostrarProductos
        3: Llamar EliminarProducto
        4: Llamar ActualizarProducto
        5: Escribir "Saliendo del sistema..."
      FinSegun
    Hasta Que opcion = 5
FinProceso
```

DIAGRAMA DE FLUJO (DF)

Representación gráfica sugerida:

Inicio → Verifica contraseña → Menú
→ opción 3 → eliminarProducto
→ opción 4 → actualizarProducto
→ opción 5 → salir

GRAFO DE FLUJO (GF)

Nodos:
1: Inicio

2: Validar contraseña
3: Menú
4: EliminarProducto
5: ActualizarProducto
6: Salir
7: Error opción

Aristas: conexiones entre las decisiones y acciones del menú.

IDENTIFICACIÓN DE RUTAS

R1: Contraseña correcta → Menú → Opción 3 → eliminar → guardar
R2: Contraseña correcta → Menú → Opción 4 → actualizar → guardar
R3: Contraseña correcta → Menú → Opción 5 → Fin
R4: Contraseña incorrecta → Fin

COMPLEJIDAD CICLOMÁTICA

V(G) = P + 1 = 3 + 1 = 4
V(G) = A - N + 2 = 8 - 6 + 2 = 4