Prueba de Caja Blanca

"Proyecto Proformas Serviglas"

Integrantes: Mateo Neppas, Morrison Quillupangui, Mateo Arellano y Freddy Fuentes

Fecha: 2025-02-11

Contenido

Historia de Revisión

Fecha	Versión	Descripción	Autores
09/01/2025	1	Creación del documento de cajas blanca	Mateo Neppas, Morrison Quillupangui, Mateo Arellano y Freddy Fuentes
15/01/2025	2	Modificación de Diagramas y Complejidad Ciclomatica	Mateo Neppas, Morrison Quillupangui, Mateo Arellano y Freddy Fuentes
20/01/2025	3	Modificación de Diagramas y Complejidad Ciclomatica	Mateo Neppas, Morrison Quillupangui, Mateo Arellano y Freddy Fuentes
12/02/2025	4	Modificación de Diagramas y Complejidad Ciclomatica	Mateo Neppas, Morrison Quillupangui, Mateo Arellano y Freddy Fuentes

Prueba caja blanca de describa el requisito funcional

1 REQOO1 REGISTRO DE USUARIO

El requisito 1 al no tener nodos predicados no es apto para hacer la complejidad ciclomatica.

2 REQ002 INICIO DE SESION

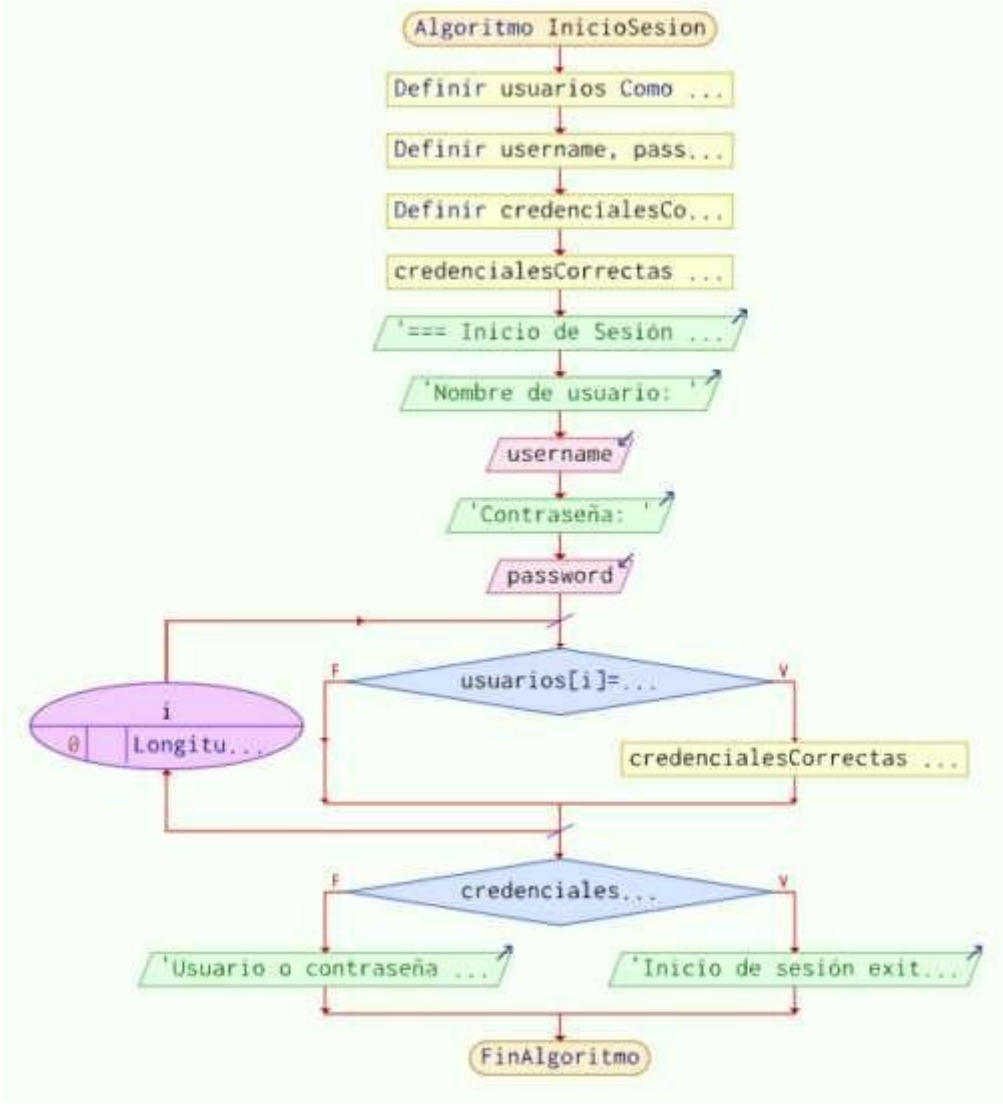
2.1 CÓDIGO FUENTE

```
bool loginUser() {
    string username, password;
    cout << "\n=== Inicio de sesion ===\n";
    cout << "Nombre de usuario: ";
    cin >> username;
    cout << "Contrasena: ";
    password = getPassword();
    for (const auto& user : users) {
        if (user.username == username && user.password == password) {
            cout << "\nInicio de sesion exitoso!\n";
            return true;
        }
    }
    cout << "\nUsuario o contrasena incorrectos.\n";
    return false;
}
```

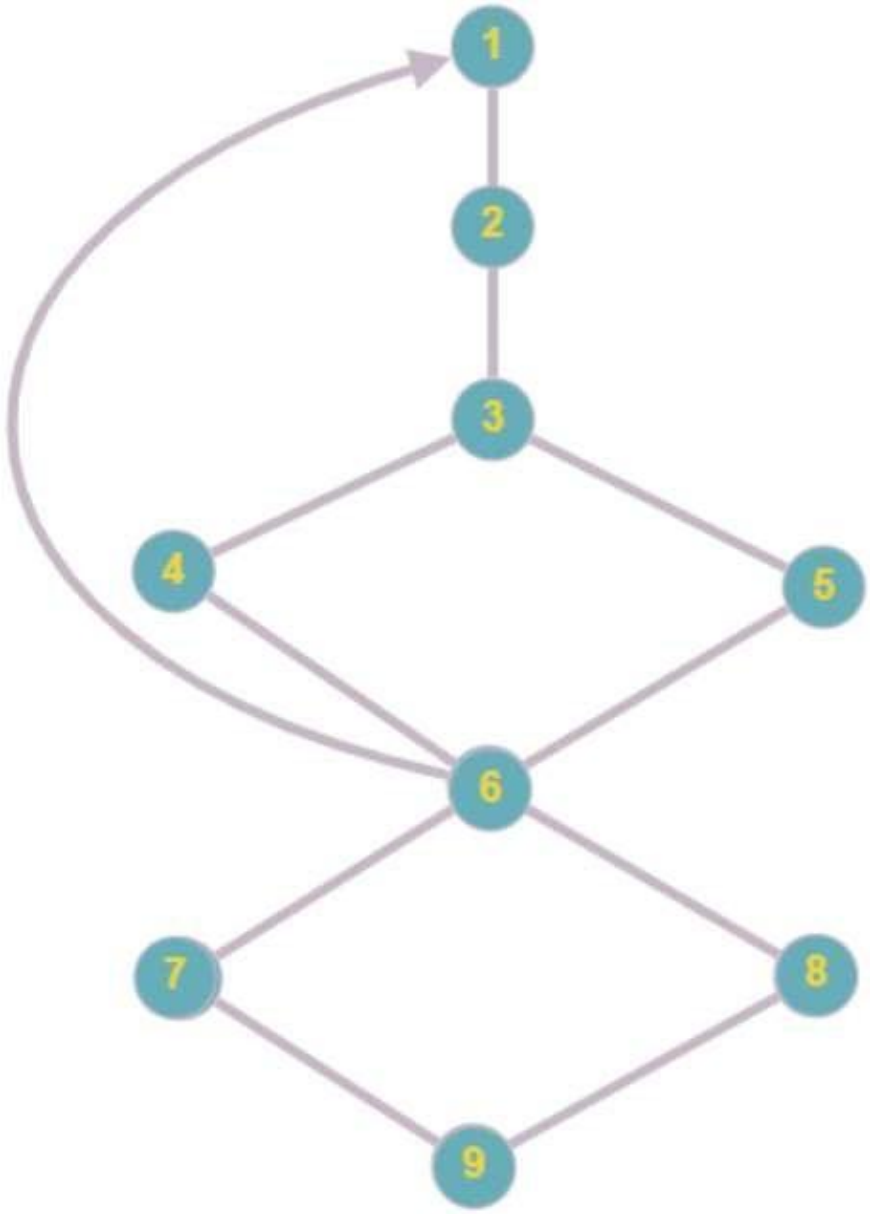
PSEUDOCODIGO

```
Proceso InicioSesion
    Definir usuarios Como Arreglo de Cadena;
    Definir username, password Como Cadena;
    Definir credencialesCorrectas Como Logico;
    credencialesCorrectas ← Falso;

    Escribir "=== Inicio de Sesión ===";
    Escribir "Nombre de usuario: ";
    Leer username;
    Escribir "Contraseña: ";
    Leer password;

    Para i ← 0 Hasta Longitud(usuarios) - 1 Hacer
        Si usuarios[i] = username + "," + password Entonces
            credencialesCorrectas ← Verdadero;
        FinSi
    FinPara

    Si credencialesCorrectas Entonces
        Escribir "Inicio de sesión exitoso!";
    Sino
        Escribir "Usuario o contraseña incorrectos.";
    FinSi
FinProceso
```

2.2 DIAGRAMA DE FLUJO



2.3 GRAFO DE FLUJO



1* Iniciar función
2* Obtener username y password
3* Verificar username o password vacíos
4* Vuelve a pedir username y password
5* Credenciales validas
6* Comprobar largo
7* Analizar Credenciales
8* Mensaje de Error
9* Inicio Valido
10* Fin

2.4 4. IDENTIFIACCIÒN DE LAS RUTAS

RUTAS

Ruta 1: N1 → N2 → N3 → N5 → N6 → N7 → N9 →N10

Ruta 2: N1 → N2 → N3 → N5 → N6 → N7 → N8 →N10

Ruta 3: N1 → N2 → N4 → N5 → N6 → N7 → N8 →N10

2.5 COMPLEJIDAD CICLOMÁTICA

Se puede calcular de las siguientes formas:

3. V(G) = número de nodos predicados(decisiones)+1
   V(G)= 0
   2+1=3

4. V(G) = A – N + 2 V(G)= 10 - 9 +2 =3

Por lo tanto, la complejidad ciclomática del código es 1 y no hay nodos predicados

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

3 REQ003 CREACION DE PROFORMAS

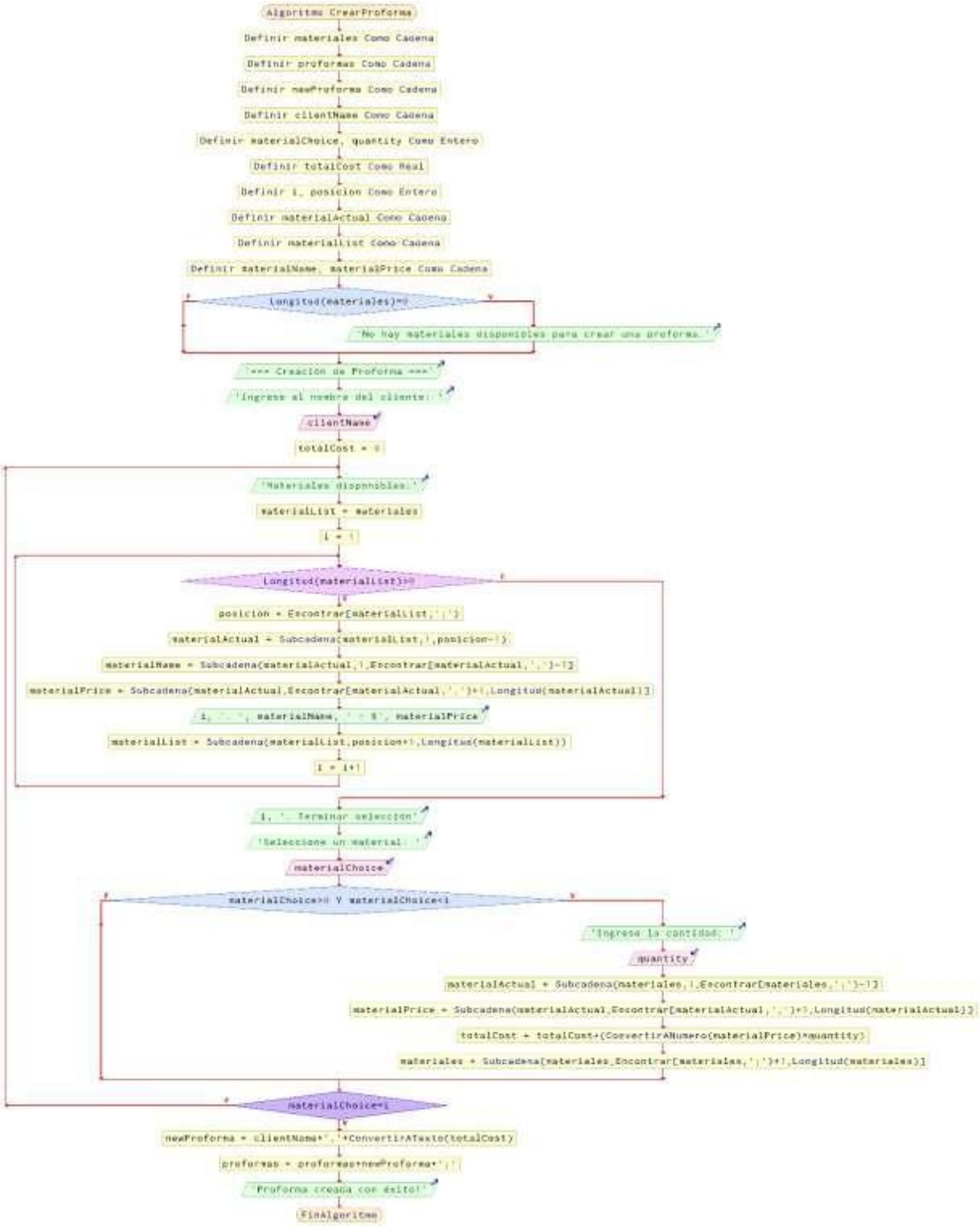
3.1 CÓDIGO FUENTE

```
void createProforma() {
    if (materials.empty()) {
        cout << "\nNo hay materiales disponibles para crear una proforma.\n";
        return;
    }

    Proforma newProforma;
    cout << "\n=== Creacion de Proforma ===\n";
    cout << "Ingrese el nombre del cliente: ";
    cin.ignore(); // Limpiar el buffer de entrada
    getline(cin, newProforma.clientName);

    float totalCost = 0;
    int materialChoice;
    do {
        cout << "\nMateriales disponibles:\n";
        for (size_t i = 0; i < materials.size(); i++) {
            cout << i + 1 << ". " << materials[i].name << " - $" << materials[i].price << "\n";
        }
        cout << materials.size() + 1 << ". Terminar seleccion\n";
        materialChoice = getValidOption(1, materials.size() + 1);
        if (materialChoice > 0 && materialChoice <= materials.size()) {
            int quantity;
            cout << "Ingrese la cantidad: ";
            quantity = getValidOption(1, 1000, false); // No mostrar "Seleccione una opcion"
            float cost = materials[materialChoice - 1].price * quantity;
            newProforma.items.push_back({materials[materialChoice - 1].name, {quantity, cost}});
            totalCost += cost;
        }
    } while (materialChoice != materials.size() + 1);

    newProforma.totalCost = totalCost;
    proformas.push_back(newProforma);
    cout << "Proforma creada con exito!\n";
}
```

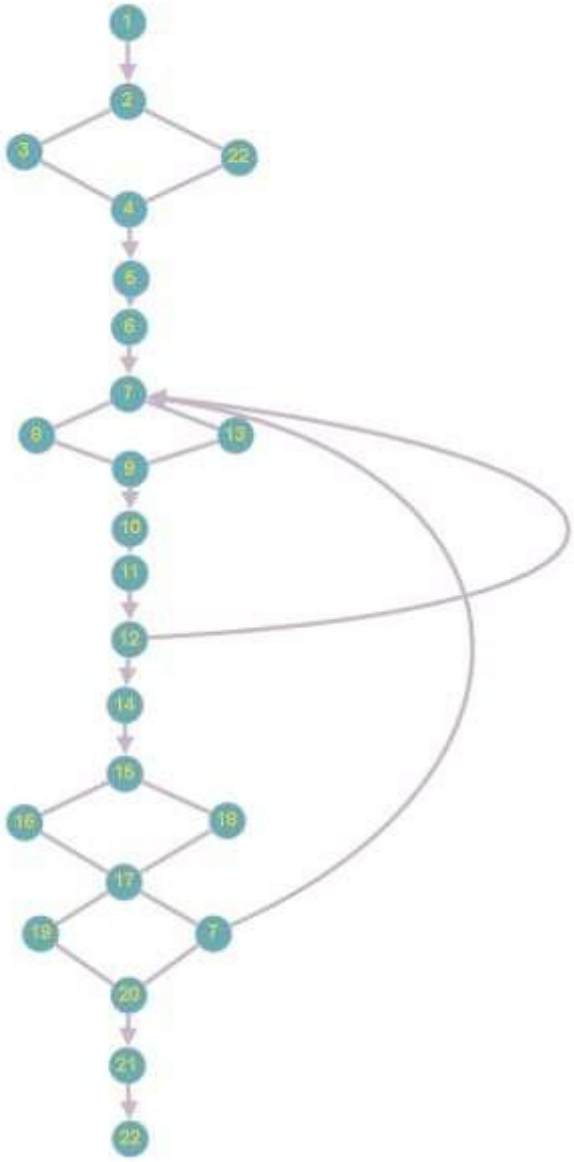
3.2 PSEUDOCODIGO

```
Algoritmo CrearProforma
    // Definir variables
    Definir materiales Como Cadena
    Definir proformas Como Cadena // Simulamos un arreglo con una cadena
    Definir newProforma Como Cadena // Simulamos un arreglo con una cadena
    Definir clientName Como Cadena
    Definir materialChoice, quantity Como Entero
    Definir totalCost Como Real
    Definir i, posicion Como Entero
    Definir materialActual Como Cadena
    Definir materialList Como Cadena
    Definir materialName, materialPrice Como Cadena

    // Verificar si hay materiales disponibles
    Si Longitud(materiales) = 0 Entonces
        Escribir 'No hay materiales disponibles para crear una proforma.'
    FinSi

    // Solicitar nombre del cliente
    Escribir '=== Creación de Proforma ==='
    Escribir 'Ingrese el nombre del cliente: '
    Leer clientName

    // Inicializar el costo total
    totalCost = 0

    // Mostrar materiales disponibles
    Repetir
        Escribir 'Materiales disponibles:'
        materialList = materiales
        i = 1
        Mientras Longitud(materialList) > 0 Hacer
            posicion = Encontrar(materialList, ';')
            materialActual = Subcadena(materialList, 1, posicion - 1)
            materialName = Subcadena(materialActual, 1, Encontrar(materialActual, ',') - 1)
            materialPrice = Subcadena(materialActual, Encontrar(materialActual, ',') + 1, Longitud(materialActual))
            Escribir i, '. ', materialName, ' - $', materialPrice
            materialList = Subcadena(materialList, posicion + 1, Longitud(materialList))
            i = i + 1
        FinMientras
        Escribir i, '. Terminar selección'

        // Solicitar selección de material
        Escribir 'Seleccione un material: '
        Leer materialChoice

        // Procesar la selección
        Si materialChoice > 0 Y materialChoice < i Entonces
            Escribir 'Ingrese la cantidad: '
            Leer quantity
            materialActual = Subcadena(materiales, 1, Encontrar(materiales, ';') - 1)
            materialPrice = Subcadena(materialActual, Encontrar(materialActual, ',') + 1, Longitud(materialActual))
            totalCost = totalCost + (ConvertirANumero(materialPrice) * quantity)
            materiales = Subcadena(materiales, Encontrar(materiales, ';') + 1, Longitud(materiales))
        FinSi
    Hasta Que materialChoice = i

    // Guardar la proforma
    newProforma = clientName + ',' + ConvertirATexto(totalCost)
    proformas = proformas + newProforma + ';'
    Escribir 'Proforma creada con éxito!'
FinAlgoritmo
```

3.3 DIAGRAMA DE FLUJO



3.4 GRAFO DE FLUJO



Nodo 1: Iniciar proceso.

Nodo 2: Verificar si Longitud(materiales) = 0.

Nodo 3: Solicitar nombre del cliente (clientName).

Nodo 4: Inicializar totalCost a 0.

Nodo 5: Mostrar "Materiales disponibles:".

Nodo 6: Inicializar materialList con materiales y i a 1.

Nodo 7: Verificar si Longitud(materialList) > 0.

Sí: Continuar.

No: Ir al Nodo 13.

Nodo 8: Encontrar la posición del primer ; en materialList.

Nodo 9: Extraer materialActual desde materialList.

Nodo 10: Extraer materialName y materialPrice desde materialActual.

Nodo 11: Mostrar i, materialName, ' - $', materialPrice.

Nodo 12: Actualizar materialList y i.

Nodo 13: Mostrar "1. Terminar selección".

Nodo 14: Solicitar selección de material (materialChoice).

Nodo 15: Verificar si materialChoice > 0 y materialChoice < i.

Nodo 16: Solicitar cantidad (quantity).

Nodo 17: Calcular totalCost y actualizar materiales.

Nodo 18: Verificar si materialChoice = i.

Nodo 19: Crear newProforma con clientName y totalCost.

Nodo 20: Agregar newProforma a proformas.

Nodo 21: Mostrar "Proforma creada con éxito!".

Nodo 22: Fin.

3.5 IDENTIFIACCIÒN DE LAS RUTAS

RUTAS

Ruta 1: 1→2→22

Ruta 2: 1→2→3→4→5→6→7→13→18→19→20→21→22

Ruta 3:
1→2→3→4→5→6→7→8→9→10→11→12→7→13→14→15→16→17→18→19→20→21→22

Ruta 4: 1→2→3→4→5→6→(7→8→9→10→11→12 →7→13→14→15→16→17→18→7→13→18→19→20→21→22

Ruta 5:
1→2→3→4→5→6→(7→8→9→10→11→12)n→7→13→14→15→18→19→20→21→22

3.6 COMPLEJIDAD CICLOMÁTICA

Se puede calcular de las siguientes formas:

5. V(G) = número de nodos predicados(decisiones)+1
   V(G)= 0
   2+1=3

   V(G)=25−22+2(1)

   V(G)=5V(G) = 5

   V(G)=5

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

4 REQ004 INGRESO DE MATERIALES

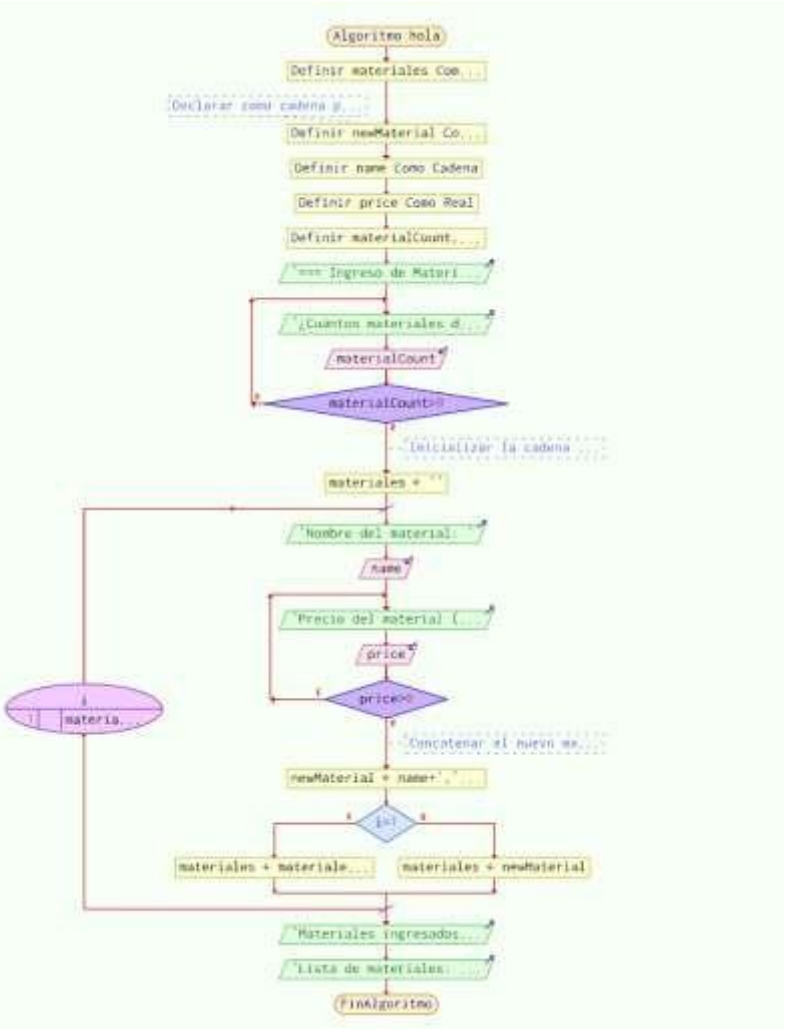
4.1 CÓDIGO FUENTE

```
void addMaterial() {
    cout << "\n=== Ingreso de Material ===\n";
    int materialCount;
    do {
        cout << "Cuantos materiales desea ingresar? ";
        materialCount = getValidOption(1, 100, false); // No mostrar "Seleccione una opcion"
    } while (materialCount <= 0);

    for (int i = 0; i < materialCount; ++i) {
        Material newMaterial;
        cout << "Nombre del material: ";
        cin >> newMaterial.name;
        do {
            cout << "Precio del material ($): ";
            cin >> newMaterial.price;
            if (cin.fail() || newMaterial.price <= 0) {
                cin.clear();
                cin.ignore(numeric_limits<streamsize>::max(), '\n');
                cout << "El precio debe ser un numero mayor que 0. Intente de nuevo.\n";
            }
        } while (cin.fail() || newMaterial.price <= 0);
        materials.push_back(newMaterial);
    }
    cout << "Materiales ingresados con exito!\n";
}
```

4.2 PSEUDOCODIGO

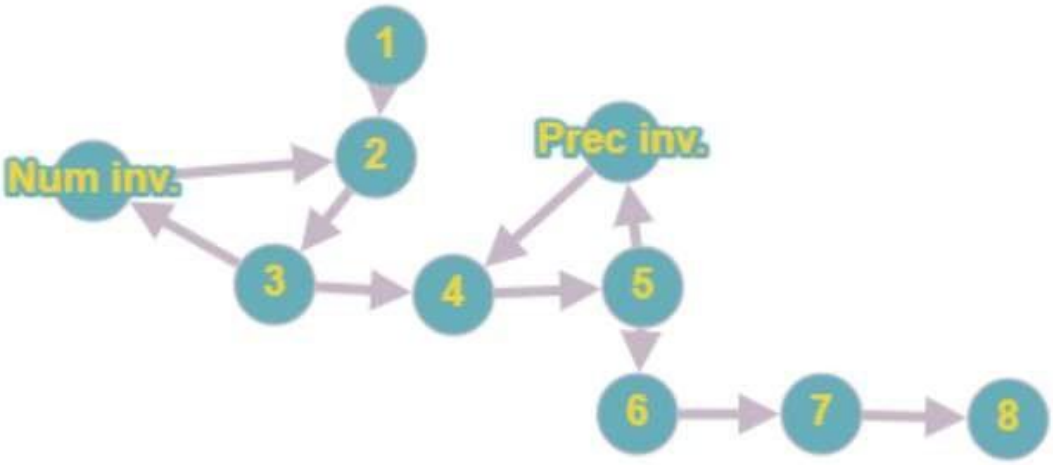
```
Algoritmo hola
    Definir materiales Como Cadena; // Declarar como cadena para almacenar múltiples valores
    Definir newMaterial Como Cadena;
    Definir name Como Cadena;
    Definir price Como Real;
    Definir materialCount, i Como Entero;

    Escribir "=== Ingreso de Material ===";
    Repetir
        Escribir "¿Cuántos materiales desea ingresar? ";
        Leer materialCount;
    Hasta Que materialCount > 0;

    // Inicializar la cadena de materiales
    materiales ← "";

    Para i ← 1 Hasta materialCount Hacer
        Escribir "Nombre del material: ";
        Leer name;
        Repetir
            Escribir "Precio del material ($): ";
            Leer price;
        Hasta Que price > 0;

        // Concatenar el nuevo material a la cadena de materiales
        newMaterial ← name + "," + ConvertirATexto(price);
        Si i = 1 Entonces
            materiales ← newMaterial;
        Sino
            materiales ← materiales + ";" + newMaterial;
        FinSi
    FinPara

    Escribir "Materiales ingresados con éxito!";
    Escribir "Lista de materiales: ", materiales;
FinAlgoritmo
```

4.3 DIAGRAMA DE FLUJO



4.4 4.2. GRAFO DE FLUJO

- **N1**: Inicio del proceso.
- **N2**: Solicitud de número.
- **N3**: Validación del número.
- **N4**: Solicitud de precio.
- **N5**: Validación del precio.
- **N6**: Procesamiento de datos.
- **N7**: Confirmación de ingreso.
- **N8**: Fin del proceso (ingreso exitoso).



4.5 IDENTIFICACION DE RUTAS

- **Ruta 1 (Ingreso exitoso)**: N1 → N2 → N3 → N4 → N5 → N6 → N7 → N8.
- **Ruta 2 (Número inválido)**: N1 → N2 → N3 → N2 (bucle para volver a solicitar el número).
- **Ruta 3 (Precio inválido)**: N1 → N2 → N3 → N4 → N5 → N4 (bucle para volver a solicitar el precio).

4.6 COMPLEJIDAD CICLOMÁTICA REQU 4

Se puede calcular de las siguientes formas:

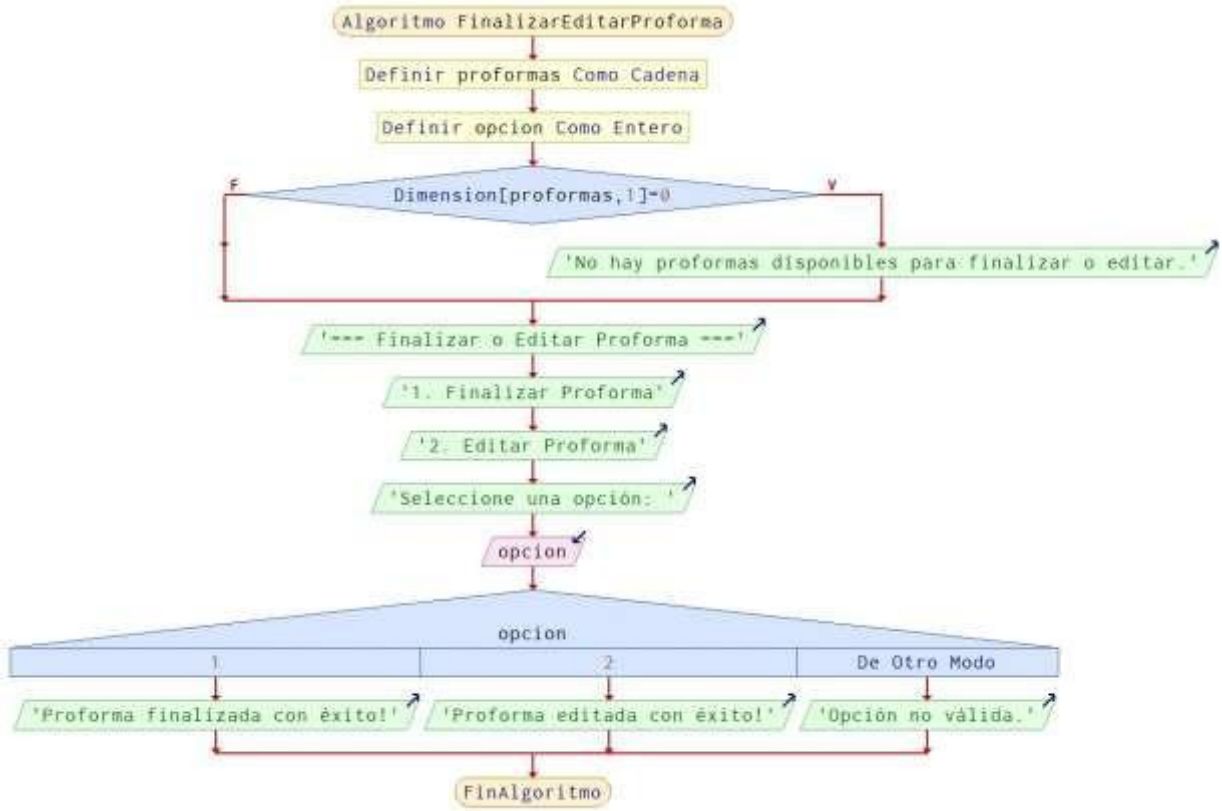
- V(G) = número de nodos predicados(decisiones)+1 V(G)= 1+1= 4
- V(G) = A – N + 2 V(G)= 8– 8 + 2 = 2

5 REQ005 MENU FINALIZACION DE PROFORMAS

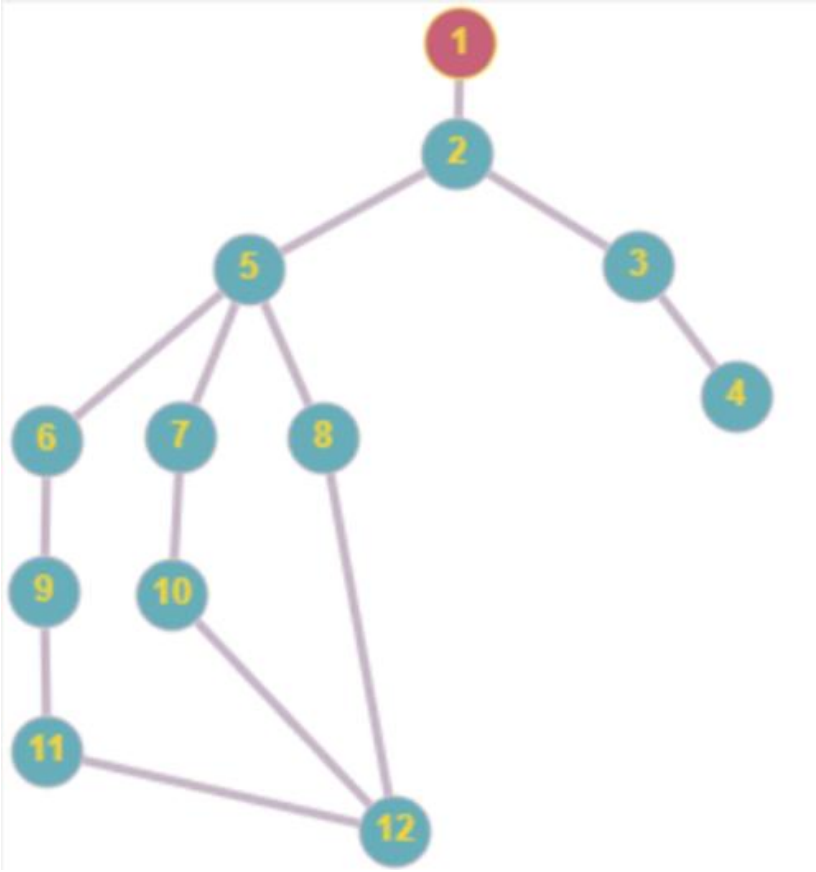
5.1 CODIGO FUENTE

```cpp

//REQ 5
void finalizarEditarProforma() {
    if (proformas.empty()) {
        cout << "\nNo hay proformas disponibles para finalizar o editar.\n";
        return;
    }

    cout << "\n=== Finalizar o Editar Proforma ===\n";
    cout << "1. Finalizar Proforma\n";
    cout << "2. Editar Proforma\n";
    cout << "Seleccione una opcion: ";
    int opcion = getValidOption(1, 2);

    switch (opcion) {
        case 1:
            cout << "Proforma finalizada con exito!\n";
            break;
        case 2:
            cout << "Proforma editada con exito!\n";
            break;
        default:
            cout << "Opcion no valida.\n";
            break;
    }
}
```

5.2 DIAGRAMA DE FLUJO



5.3 GRAFO DE FUJO



5.4 Rutas:

- **Ruta 1 (Ver Proforma):**
  N1 → N2 → N5 → N6 → N9 → N11 → N12
- **Ruta 2 (Eliminiar Proforma):**
  N1 → N2 → N5 → N7 → N10 → N12
- **Ruta 3 (Proformas vacios):**
  N1 → N2 → N3 →N4

5.5 Complejidad Ciclomática

La complejidad ciclomática (V(G)) se calcula como:

Copy V(G) = Número de arcos - Número de nodos + 2

2+1=3

V(G) = 13 - 12 + 2 = 3

Interpretación: Hay 3 caminos independientes en este código.