Prueba de Caja Blanca

“SISTEMA DE INVENTARIO DE FRUTOS SECOS”

Integrantes:

- Alvear Alexnader
- Mateo Velecela
- Antony Campoverde

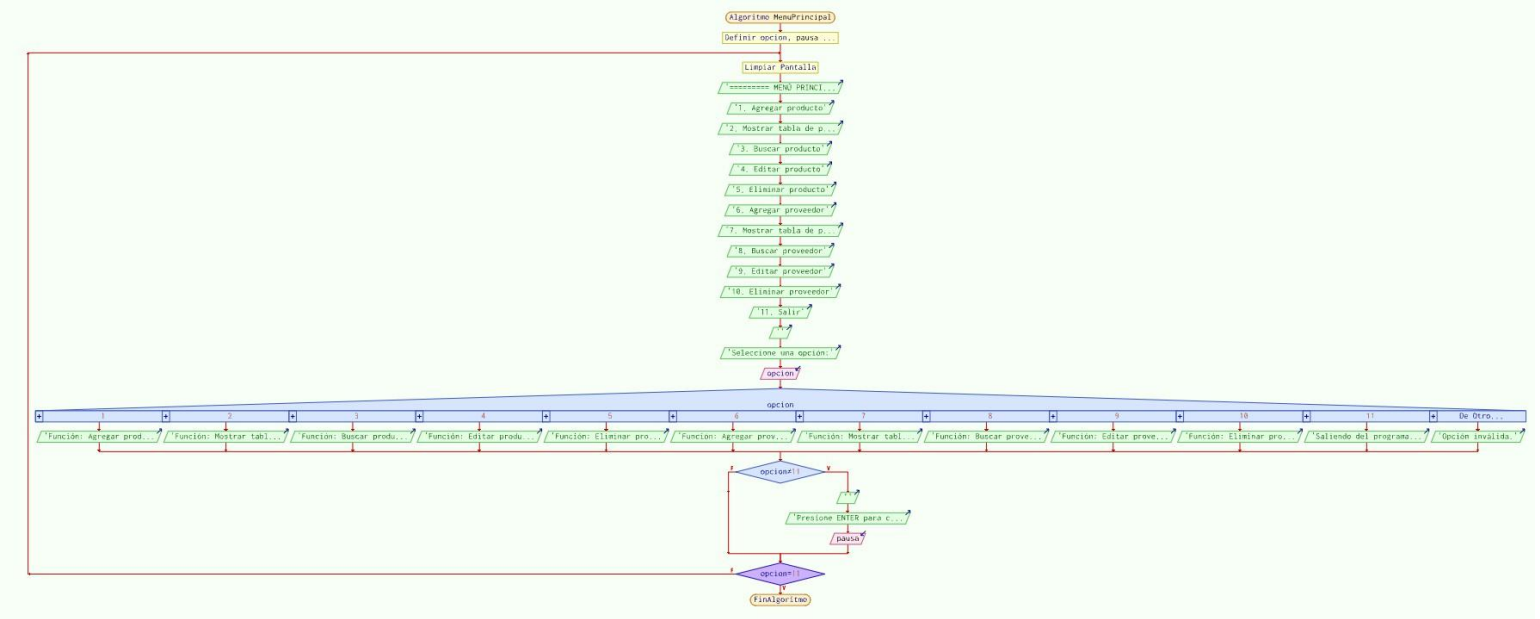
Fecha: 2025/07/24

Prueba caja blanca : El sistema deberá iniciar con un menú principal

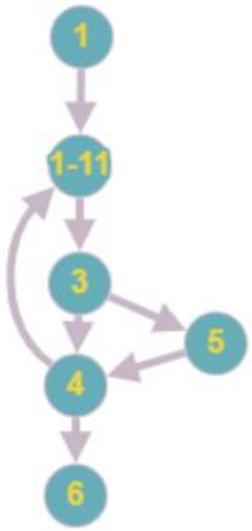
1. CÓDIGO FUENTE

```
669         do {
670             system(CLEAR);
671             imprimirMenu();
672             if (scanf("%d", &opcion) != 1) {
673                 printf("Entrada invalida.\n");
674                 limpiarBuffer();
675                 opcion = 0;
676                 continue;
677             }
678             limpiarBuffer();
679 
680             switch (opcion) {
681                 case 1: agregarProducto(); break;
682                 case 2: mostrarTablaProductos(); break;
683                 case 3: buscarProducto(); break;
684                 case 4: editarProducto(); break;
685                 case 5: eliminarProducto(); break;
686                 case 6: agregarProveedor(); break;
687                 case 7: mostrarTablaProveedores(); break;
688                 case 8: buscarProveedor(); break;
689                 case 9: editarProveedor(); break;
690                 case 10: eliminarProveedor(); break;
691                 case 11: printf("Saliendo...\n"); break;
692                 default: printf("Opcion invalida.\n"); pausa(); break;
693             }
694         } while (opcion != 11);
695 
696         return 0;
697     }
```

2. DIAGRAMA DE FLUJO (DF) PSEINT



3. GRAFO DE FLUJO (GF)



4. IDENTIFIACCIÒN DE LAS RUTAS (Camino básico)

RUTAS

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

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)= 7– 6 + 2 = 3

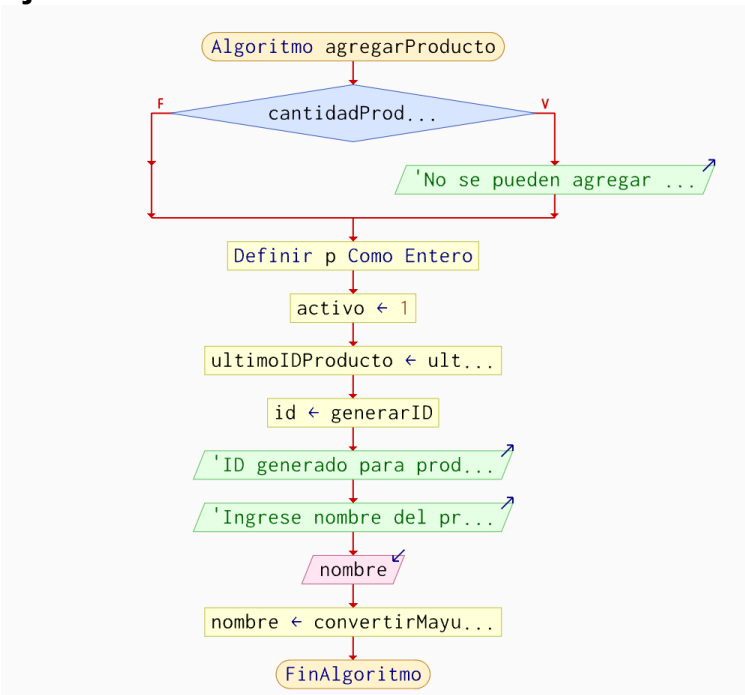
DONDE:
P: Número de nodos predicado
A: Número de aristas
N: Número de nodos

Prueba caja blanca de Requisito N° 2: Como agregar productos

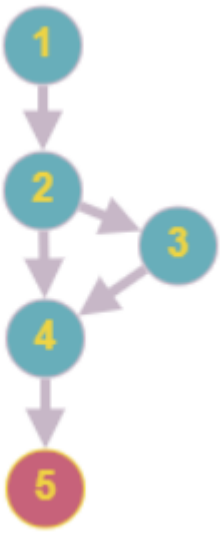
1. **CÓDIGO FUENTE**

```
158 void agregarProducto() {
159     if (cantidadProductos >= MAX_PRODUCTOS) {
160         printf("No se pueden agregar mas productos.\n");
161         pausa();
162         return;
163     }
164     Producto p;
165     p.activo = 1;
166
167     ultimoIDProducto++;
168     strcpy(p.id, generarID(ultimoIDProducto - 1)); // Generar nuevo ID
169     printf("ID generado para producto: %s\n", p.id);
170
171     printf("Ingrese nombre del producto: ");
172     fgets(p.nombre, sizeof(p.nombre), stdin);
173     p.nombre[strcspn(p.nombre, "\n")] = '\0';
174     convertirMayusculas(p.nombre);
```

2. **Diagrama de flujo**



3. **Grafo de flujo (GF)**



4. **IDENTIFIACCIÒN DE LAS RUTAS (Camino básico)**

RUTAS

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

5. **COMPLEJIDAD CICLOMÁTICA**

Se puede calcular de las siguientes formas:

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

b. V(G) = A – N + 2
V(G)= 4 – 4 +2 = 2

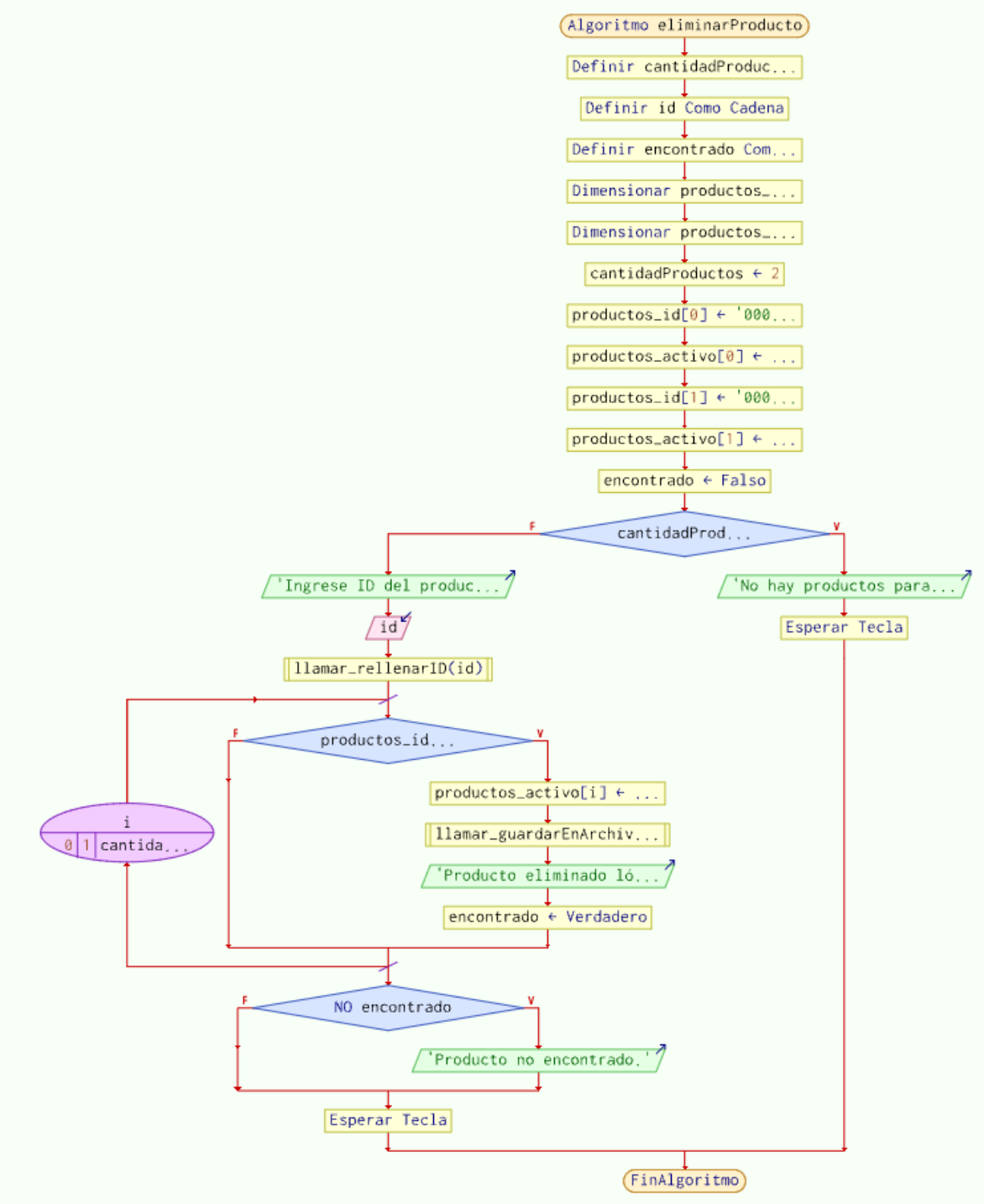
DONDE:
P: Número de nodos predicado
A: Número de aristas
N: Número de nodos

Prueba caja blanca de Requisito N° 3: Eliminar producto

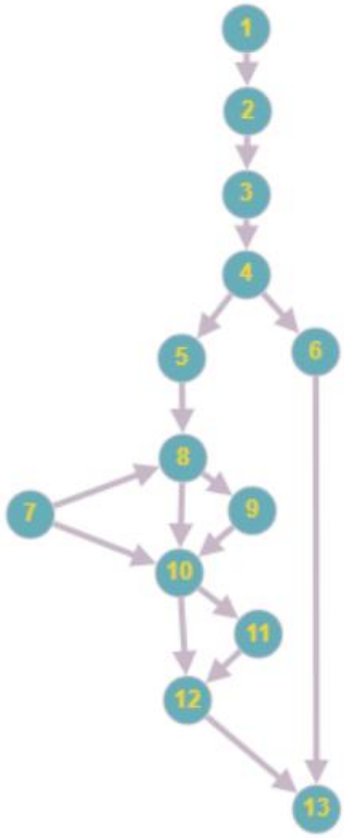
1. **CÓDIGO FUENTE**

```
360
361 void eliminarProducto() {
362     if (cantidadProductos == 0) {
363         printf("No hay productos para eliminar.\n");
364         pausa();
365         return;
366     }
367     char id[20];
368     printf("Ingrese ID del producto a eliminar (numeros, se rellenara a 10 digitos): ");
369     fgets(id, sizeof(id), stdin);
370     id[strcspn(id, "\n")] = '\0';
371     rellenarID(id);
372
373     for (int i = 0; i < cantidadProductos; i++) {
374         if (strcmp(productos[i].id, id) == 0 && productos[i].activo) {
375             productos[i].activo = 0;
376             guardarEnArchivos();
377             printf("Producto eliminado lógicamente.\n");
378             pausa();
379             return;
380         }
381     }
382     printf("Producto no encontrado.\n");
383     pausa();
384 }
```

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



3. GRAFO DE FLUJO (GF)



4. IDENTIFIACCIÒN DE LAS RUTAS (Camino básico)

RUTAS

R1: 1-

2-3-4-

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

5. COMPLEJIDAD CICLOMÁTICA

Se puede calcular de las siguientes formas:

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

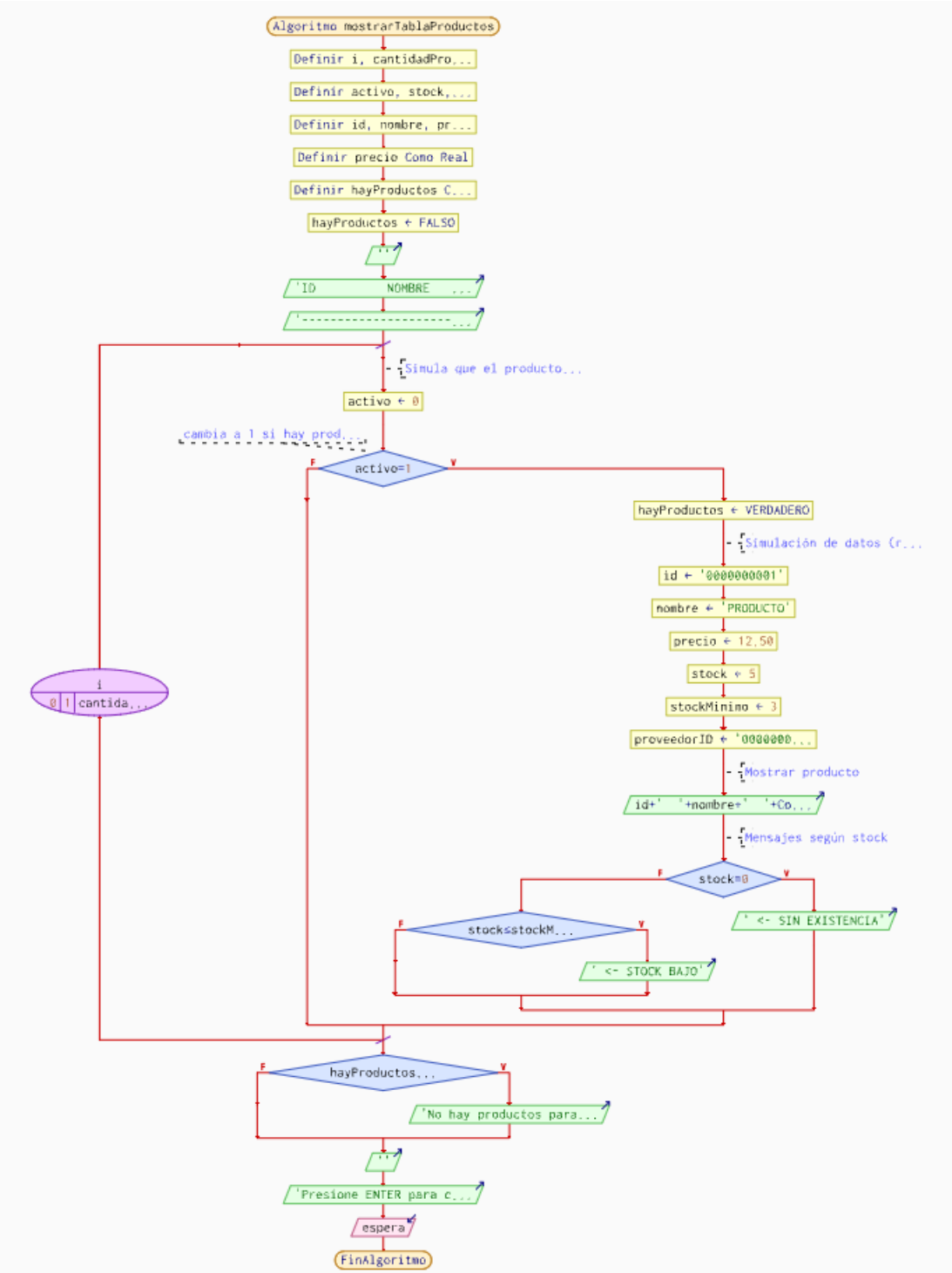
V(G)=

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

Prueba caja blanca de Requisito N° 4: Cómo veo la lista de los productos

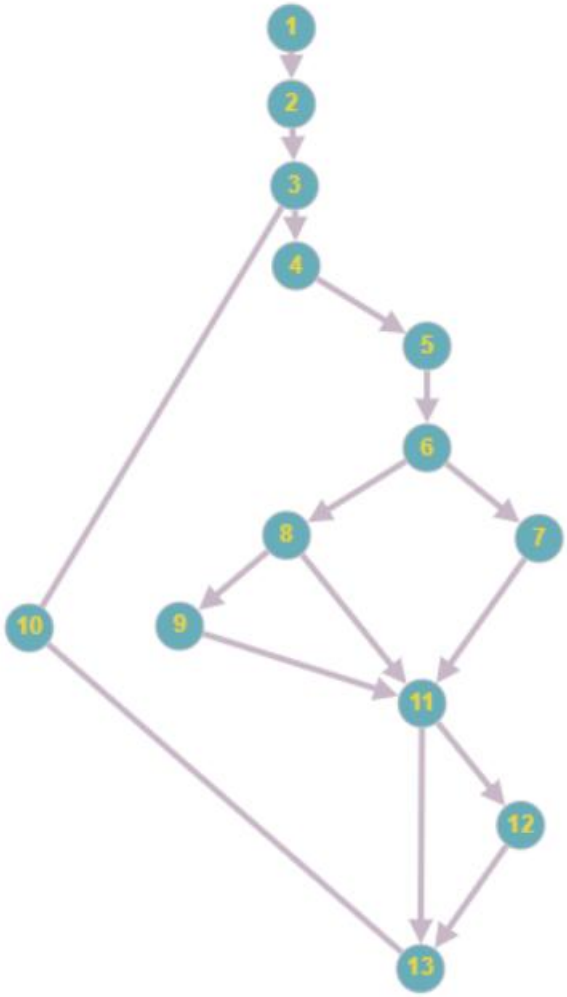
1. **CÓDIGO FUENTE**

```
121 void mostrarTablaProductos() {
122     printf("\n%-12s%-20s%-10s%-8s%-14s%-12s\n", "ID", "NOMBRE", "PRECIO", "STOCK", "STOCK MINIMO", "PROVEEDOR");
123     printf("-------------------------------------------------------------------------------\n");
124     for (int i = 0; i < cantidadProductos; i++) {
125         if (productos[i].activo) {
126             printf("%-12s%-20s%10.2f%-8d%-14d%-12s",
127                    productos[i].id,
128                    productos[i].nombre,
129                    productos[i].precio,
130                    productos[i].stock,
131                    productos[i].stockMinimo,
132                    productos[i].proveedorID);
133             if (productos[i].stock == 0)
134                 printf(" <- SIN EXISTENCIA");
135             else if (productos[i].stock <= productos[i].stockMinimo)
136                 printf(" <- STOCK BAJO");
137             printf("\n");
138         }
139     }
140     pausa();
141 }
```



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

3. GRAFO DE FLUJO (GF)



4. IDENTIFIACCIÒN DE LAS RUTAS

(Camino básico) RUTAS

R1: 1-2-3-4-5-6-7-11-12-13
R2: 1-2-3-4-5-6-8-11-13
R3: 1-2-3-4-5-6-7-11-13
R4: 1-2-3-4-5-6-8-9-11-12-13
R5: 1-2-3-4-5-6-8-9-11-13

5. COMPLEJIDAD CICLOMÁTICA

Se puede calcular de las siguientes formas:

- V(G) = número de nodos predicados(decisiones)
  V(G)=4+1=5

- V(G) = A – N + 2
- V(G)= 15-12+2=5

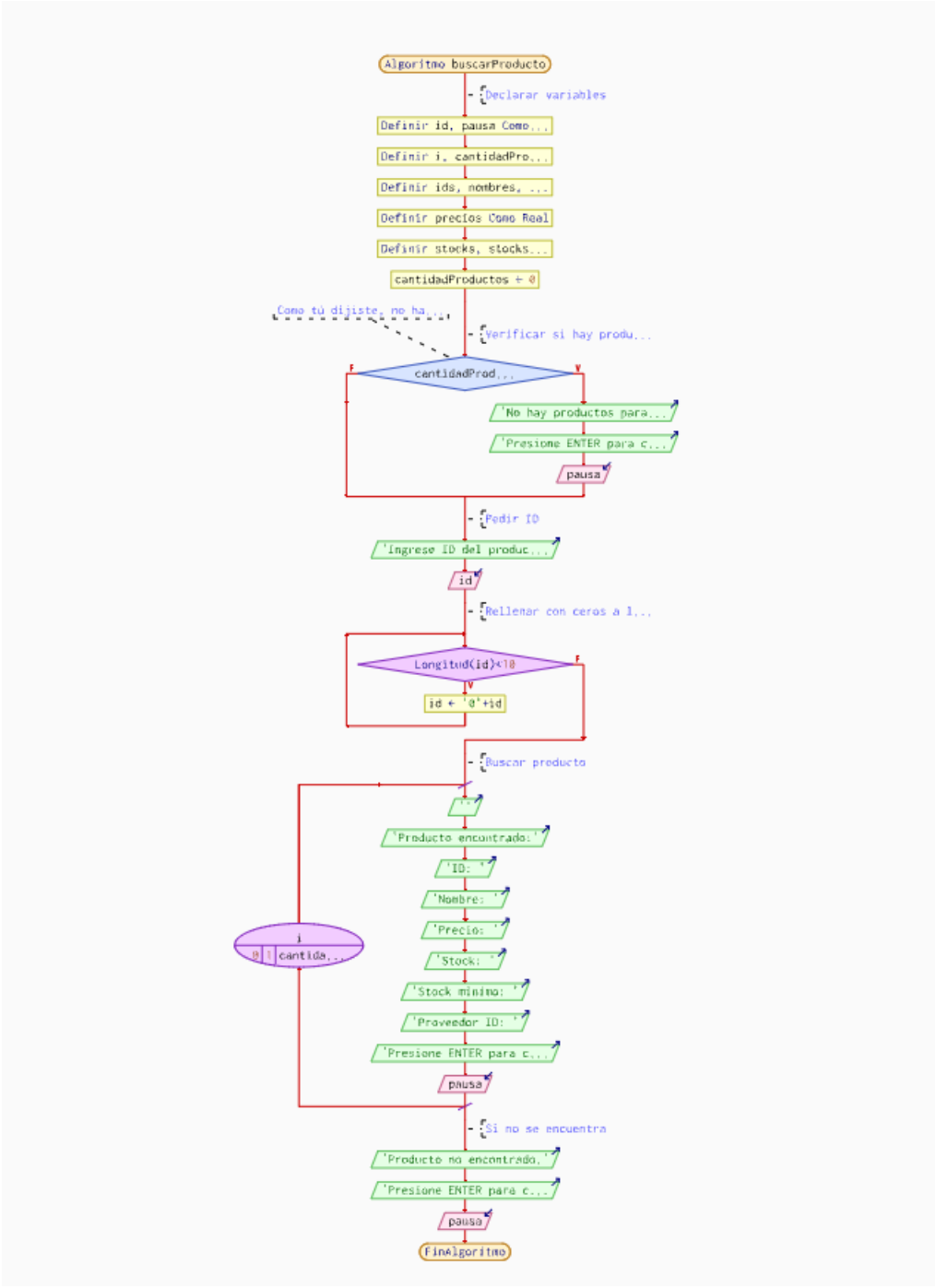
DONDE:
P: Número de nodos predicado
A: Número de aristas
N: Número de nodos

Prueba caja blanca de Requisito N° 5: Cómo busco productos

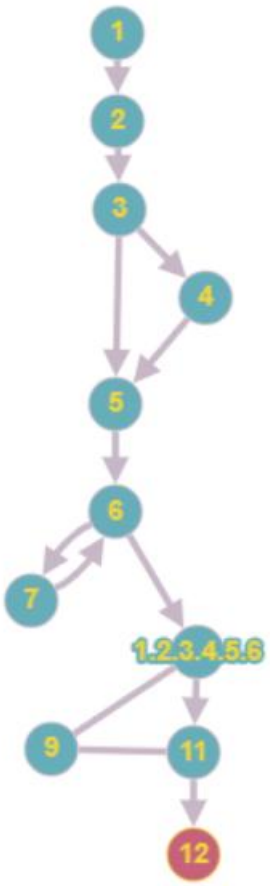
1. **CÓDIGO FUENTE**

```
496  void buscarProducto() {
497      if (cantidadProductos == 0) {
498          printf("No hay productos para buscar.\n");
499          pausa();
500          return;
501      }
502      char id[20];
503      printf("Ingrese ID del producto a buscar (numeros, se rellenara a 10 digitos): ");
504      fgets(id, sizeof(id), stdin);
505      id[strcspn(id, "\n")] = '\0';
506      rellenarID(id);
507
508      for (int i = 0; i < cantidadProductos; i++) {
509          if (strcmp(productos[i].id, id) == 0 && productos[i].activo) {
510              printf("\nProducto encontrado:\n");
511              printf("ID: %s\n", productos[i].id);
512              printf("Nombre: %s\n", productos[i].nombre);
513              printf("Precio: %.2f\n", productos[i].precio);
514              printf("Stock: %d\n", productos[i].stock);
515              printf("Stock minimo: %d\n", productos[i].stockMinimo);
516              printf("Proveedor ID: %s\n", productos[i].proveedorID);
517              pausa();
518              return;
519          }
520      }
521      printf("Producto no encontrado.\n");
522      pausa();
523  }
```

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


Algoritmo buscarProducto
Declarar variables
Definir id, pausa Como...
Definir i, cantidadPro...
Definir ids, nombres, ...
Definir precios Como Real
Definir stocks, stocks...
cantidadProductos ← 0
Como tú dijiste, no ha...
Verificar si hay produ...
cantidadProd...
F
V
'No hay productos para...
'Presione ENTER para c...
pausa
Pedir ID
'Ingrese ID del produc...
id
Rellenar con ceros a l...
Longitud(id)<10
F
V
id ← '0'+id
Buscar producto
i
'Producto encontrado:'
'ID: '
'Nombre: '
'Precio: '
'Stock: '
'Stock mínimo: '
'Proveedor ID: '
'Presione ENTER para c...
pausa
i
0
1
cantida...
Si no se encuentra
'Producto no encontrado.'
'Presione ENTER para c...
pausa
FinAlgoritmo

3. GRAFO DE FLUJO (GF)



4. IDENTIFIACCIÒN DE LAS RUTAS

(Camino básico) RUTAS

R1: 235235235235235
R2: w234235235235
R3:

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)= 11 – 10 +2= 3

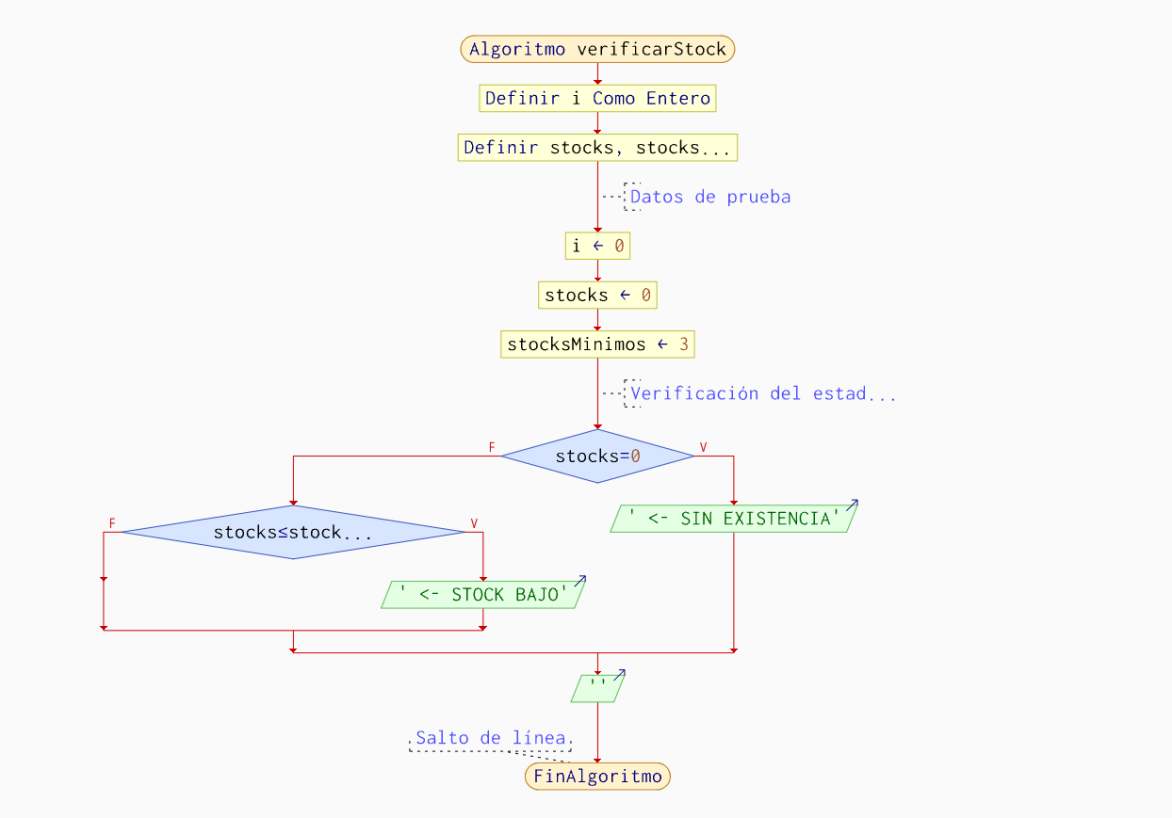
DONDE:
P: Número de nodos predicado
A: Número de aristas
N: Número de nodos

Prueba caja blanca de Requisito N° 6: Cómo alertar sobre stock (comprobado en el R.F4)

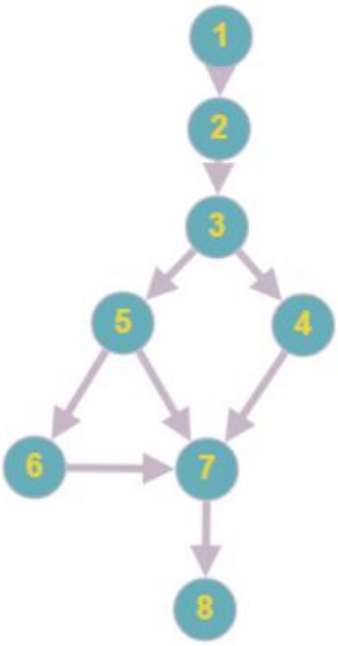
1. **CÓDIGO FUENTE**

```
133             if (productos[i].stock == 0)
134                 printf(" <- SIN EXISTENCIA");
135             else if (productos[i].stock <= productos[i].stockMinimo)
136                 printf(" <- STOCK BAJO");
137             printf("\n");
138         }
139     }
```

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



3. GRAFO DE FLUJO (GF)



4. IDENTIFIACCIÒN DE LAS RUTAS

(Camino básico) RUTAS

R1:
R2:

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 – + 2 V(G)= 9-8+2=3

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

Prueba caja blanca de Requisito N° 7: Cómo validar entrada de datos

1. **CÓDIGO FUENTE**

Validacion de datos para el ingreso del precio:

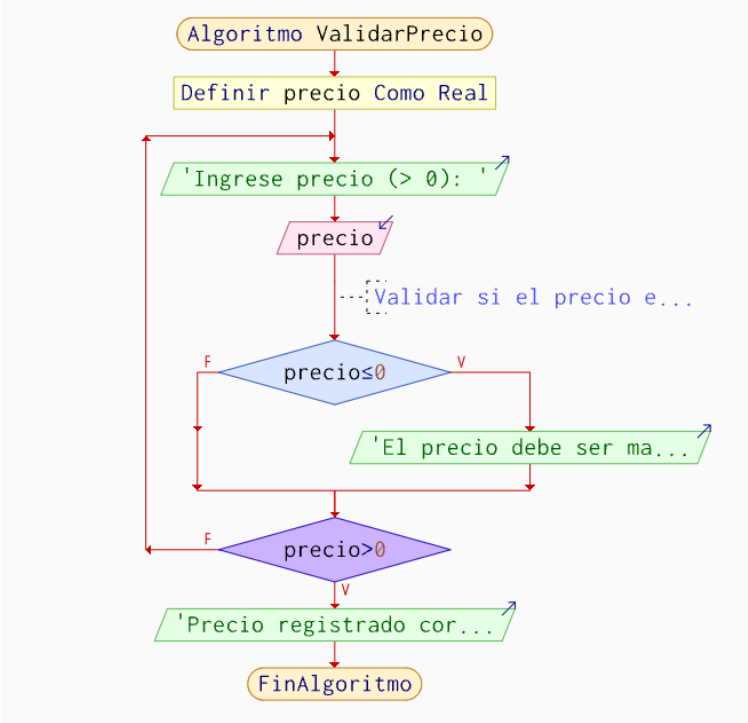
```
177     do {
178         printf("Ingrese precio (> 0): ");
179         if (scanf("%f", &p.precio) != 1) {
180             printf("Entrada invalida. Intente de nuevo.\n");
181             limpiarBuffer();
182             p.precio = 0;
183             continue;
184         }
185         if (p.precio <= 0) {
186             printf("El precio debe ser mayor a cero.\n");
187         }
188         limpiarBuffer();
189     } while (p.precio <= 0);
190
```

Validacion de datos para el ingreso del stock:

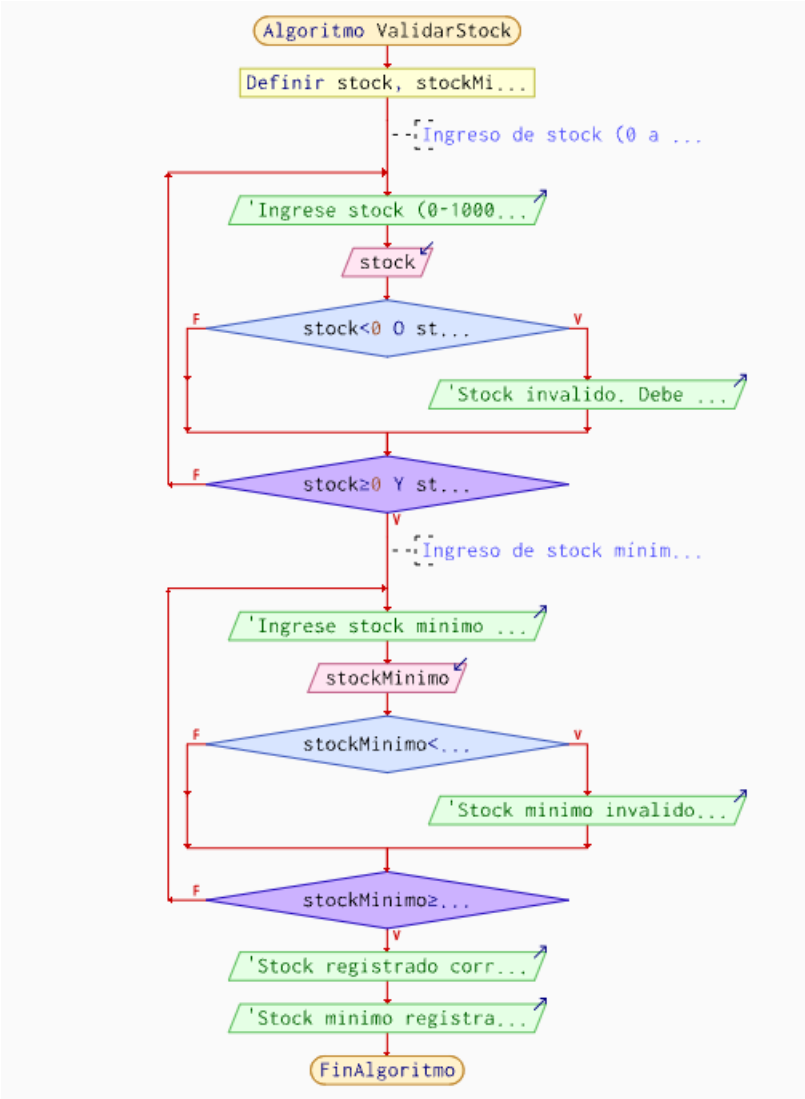
```
192     do {
193         printf("Ingrese stock (0-1000): ");
194         if (scanf("%d", &p.stock) != 1) {
195             printf("Entrada invalida. Intente de nuevo.\n");
196             limpiarBuffer();
197             p.stock = -1;
198             continue;
199         }
200         if (p.stock < 0 || p.stock > 1000) {
201             printf("Stock invalido. Debe estar entre 0 y 1000.\n");
202         }
203         limpiarBuffer();
204     } while (p.stock < 0 || p.stock > 1000);
205
206     // Stock minimo >= 0 y <= stock
207     do {
208         printf("Ingrese stock minimo (0-%d): ", p.stock);
209         if (scanf("%d", &p.stockMinimo) != 1) {
210             printf("Entrada invalida. Intente de nuevo.\n");
211             limpiarBuffer();
212             p.stockMinimo = -1;
213             continue;
214         }
215         if (p.stockMinimo < 0 || p.stockMinimo > p.stock) {
216             printf("Stock minimo invalido. Debe estar entre 0 y stock actual.\n");
217         }
218         limpiarBuffer();
219     } while (p.stockMinimo < 0 || p.stockMinimo > p.stock);
```

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

Validación de datos para el precio:



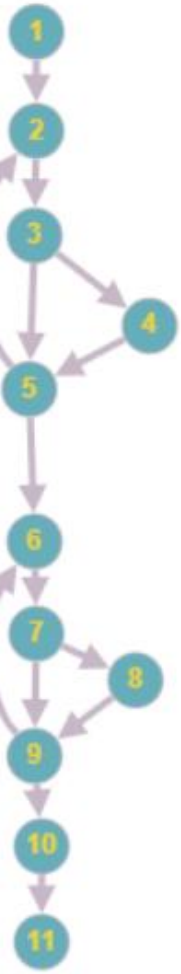
Validacion de datos para el stock:



3. GRAFO DE FLUJO (GF)

Validacion de datos para el ingreso de precio:

Validacion de datos para el stock:



4. IDENTIFIACCIÒN DE LAS RUTAS (Camino básico)

Rutas de la validación de datos para el ingreso del precio:
R1: 1-2-3-5-6-7
R2: 1-2-3-4-5-6-7
R3: 1-2-3-5-2-3-4-5-6-7

Ruta de validación de datos para el ingreso del stock:
R1: 1-2-3-5-6-7-9-10-11
R2: 1-2-3-4-5-6-7-8-9-10-11
R3: 1-2-3-5-2-3-4-5-6-7-8-9-10-11
R4:1-2-3-4-5-6-7-9-6-7-8-9-10-11
R5: 1-2-3-5-2-3-4-5-6-7-9-6-7-8-9-10-11

5. COMPLEJIDAD CICLOMÁTICA

Validacion de datos para el ingreso del precio
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

Validacion de datos para el ingreso de stock
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)=
  14-11+2 = 5

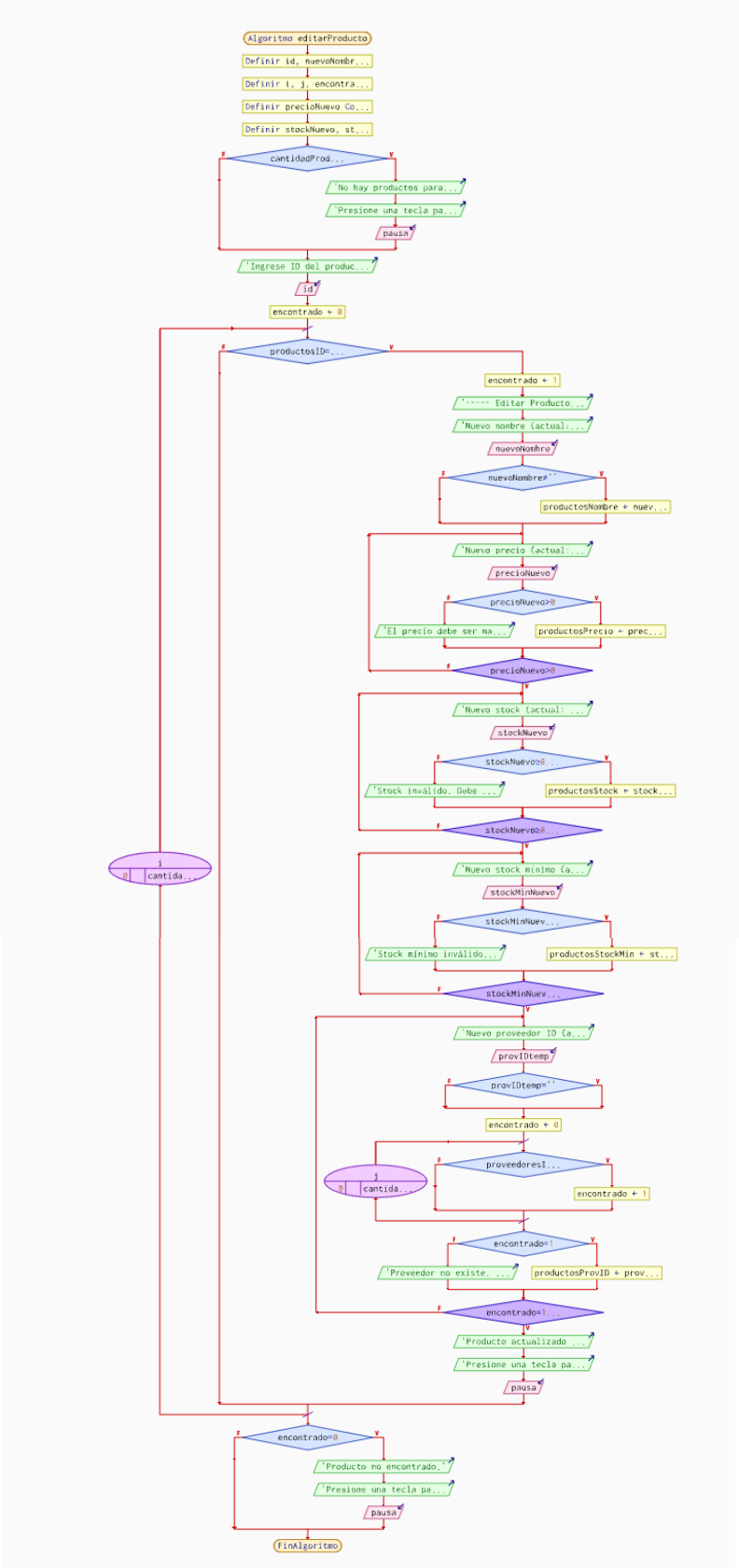
DONDE:
P: Número de nodos predicado
A: Número de aristas
N: Número de nodos

Prueba caja blanca de Requisito N° 8: Cómo editar productos

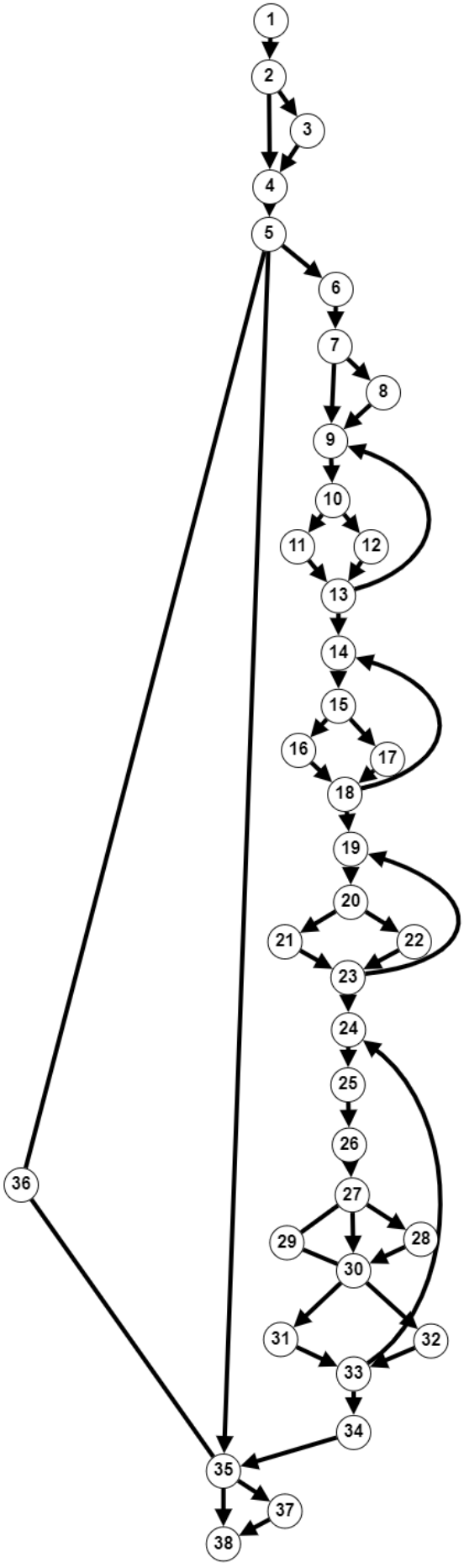
1. CÓDIGO FUENTE

```c
void editarProducto() {
    if (cantidadProductos == 0) {
        printf("No hay productos para editar.\n");
        pausa();
        return;
    }
    char id[20];
    printf("Ingrese ID del producto a editar (numeros, se rellenara a 10 digitos): ");
    fgets(id, sizeof(id), stdin);
    id[strcspn(id, "\n")] = '\0';
    rellenarID(id);

    for (int i = 0; i < cantidadProductos; i++) {
        if (strcmp(productos[i].id, id) == 0 && productos[i].activo) {
            printf("\n--- Editar Producto %s ---\n", productos[i].id);

            printf("Nuevo nombre (%s): ", productos[i].nombre);
            fgets(productos[i].nombre, sizeof(productos[i].nombre), stdin);
            productos[i].nombre[strcspn(productos[i].nombre, "\n")] = '\0';
            convertirMayusculas(productos[i].nombre);

            // Precio
            float precioNuevo;
            do {
                printf("Nuevo precio (%.2f): ", productos[i].precio);
                if (scanf("%f", &precioNuevo) != 1) {
                    printf("Entrada invalida. Intente de nuevo.\n");
                    limpiarBuffer();
                    continue;
                }
                if (precioNuevo <= 0) {
                    printf("El precio debe ser mayor a cero.\n");
                    limpiarBuffer();
                    continue;
                }
                limpiarBuffer();
                break;
            } while (1);
            productos[i].precio = precioNuevo;

            // Stock
            int stockNuevo;
            do {
                printf("Nuevo stock (%d): ", productos[i].stock);
                if (scanf("%d", &stockNuevo) != 1) {
                    printf("Entrada invalida. Intente de nuevo.\n");
                    limpiarBuffer();
                    continue;
                }
                if (stockNuevo < 0 || stockNuevo > 1000) {
                    printf("Stock invalido. Debe estar entre 0 y 1000.\n");
                    limpiarBuffer();
                    continue;
                }
                limpiarBuffer();
                break;
            } while (1);
            productos[i].stock = stockNuevo;

              do {
                  printf("Nuevo stock mínimo (%d): ", productos[i].stockMinimo);
                  if (scanf("%d", &stockMinNuevo) != 1) {
                      printf("Entrada invalida. Intente de nuevo.\n");
                      limpiarBuffer();
                      continue;
                  }
                  if (stockMinNuevo < 0 || stockMinNuevo > productos[i].stock) {
                      printf("Stock minimo invalido. Debe estar entre 0 y stock actual.\n");
                      limpiarBuffer();
                      continue;
                  }
                  limpiarBuffer();
                  break;
              } while (1);
              productos[i].stockMinimo = stockMinNuevo;

              // Proveedor ID
              char provIDtemp[20];
              do {
                  printf("Nuevo proveedor ID (%s): ", productos[i].proveedorID);
                  fgets(provIDtemp, sizeof(provIDtemp), stdin);
                  provIDtemp[strcspn(provIDtemp, "\n")] = '\0';

                  if(strlen(provIDtemp) == 0) {
                      // Si no cambia, dejar igual
                      break;
                  }

                  rellenarID(provIDtemp);

                  int encontrado = 0;
                for (int j = 0; j < cantidadProveedores; j++) {
                    if (strcmp(proveedores[j].id, provIDtemp) == 0 && proveedores[j].activo) {
                        encontrado = 1;
                        break;
                    }
                }
                if (!encontrado) {
                    printf("Proveedor no existe. Intente de nuevo.\n");
                } else {
                    strcpy(productos[i].proveedorID, provIDtemp);
                    break;
                }
            } while (1);

            guardarEnArchivos();
            printf("Producto actualizado.\n");
            pausa();
            return;
        }
    }
    printf("Producto no encontrado.\n");
    pausa();
}
```

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



3. GRAFO DE FLUJO (GF)



4. IDENTIFIACCIÒN DE LAS RUTAS (Camino básico)

RUTAS
RF1: 1-2-4-5-6-7-9-10-13-14-15-18-19-20-23-24-25-26-27-30-33-34-35-38
RF2: 1-2-4-5-6-7-8-7-9-10-13-14-15-18-19-20-23-24-25-26-27-30-33-34-35-38
RF3: 1-2-4-5-6-7-9-10-11-10-13-14-15-18-19-20-23-24-25-26-27-30-33-34-35-38
RF4: 1-2-4-5-6-7-9-10-12-10-13-14-15-18-19-20-23-24-25-26-27-30-33-34-35-38
RF5: 1-2-4-5-6-7-9-10-13-14-15-16-15-18-19-20-23-24-25-26-27-30-33-34-35-38
RF6: 1-2-4-5-6-7-9-10-13-14-15-17-15-18-19-20-23-24-25-26-27-30-33-34-35-38
RF7: 1-2-4-5-6-7-9-10-13-14-15-18-19-20-21-20-23-24-25-26-27-30-33-34-35-38
RF8: 1-2-4-5-6-7-9-10-13-14-15-18-19-20-22-20-23-24-25-26-27-30-33-34-35-38
RF9: 1-2-4-5-6-7-9-10-13-14-15-18-19-20-23-24-25-26-27-30-28-30-33-34-35-38
RF10: 1-2-4-5-6-7-9-10-13-14-15-18-19-20-23-24-25-26-27-30-31-30-33-34-35-38
RF11: 1-2-4-5-6-7-9-10-13-14-15-18-19-20-23-24-25-26-27-30-32-30-33-34-35-38
RF12: 1-2-4-5-6-7-9-10-13-14-15-18-19-20-23-24-25-26-27-30-33-34-35-37-38
RF13: 1-2-4-5-6-7-9-10-13-14-15-18-19-20-23-24-25-26-27-30-33-34-36-35-38
RF14: 1-2-3-4-5-6-7-9-10-13-14-15-18-19-20-23-24-25-26-27-30-33-34-35-38
RF15: 1-2-4-5-36-35-38

5. COMPLEJIDAD CICLOMÁTICA

Se puede calcular de las siguientes formas:

- V(G) = número de nodos predicados(decisiones)+1
  V(G)= 14+1= 15

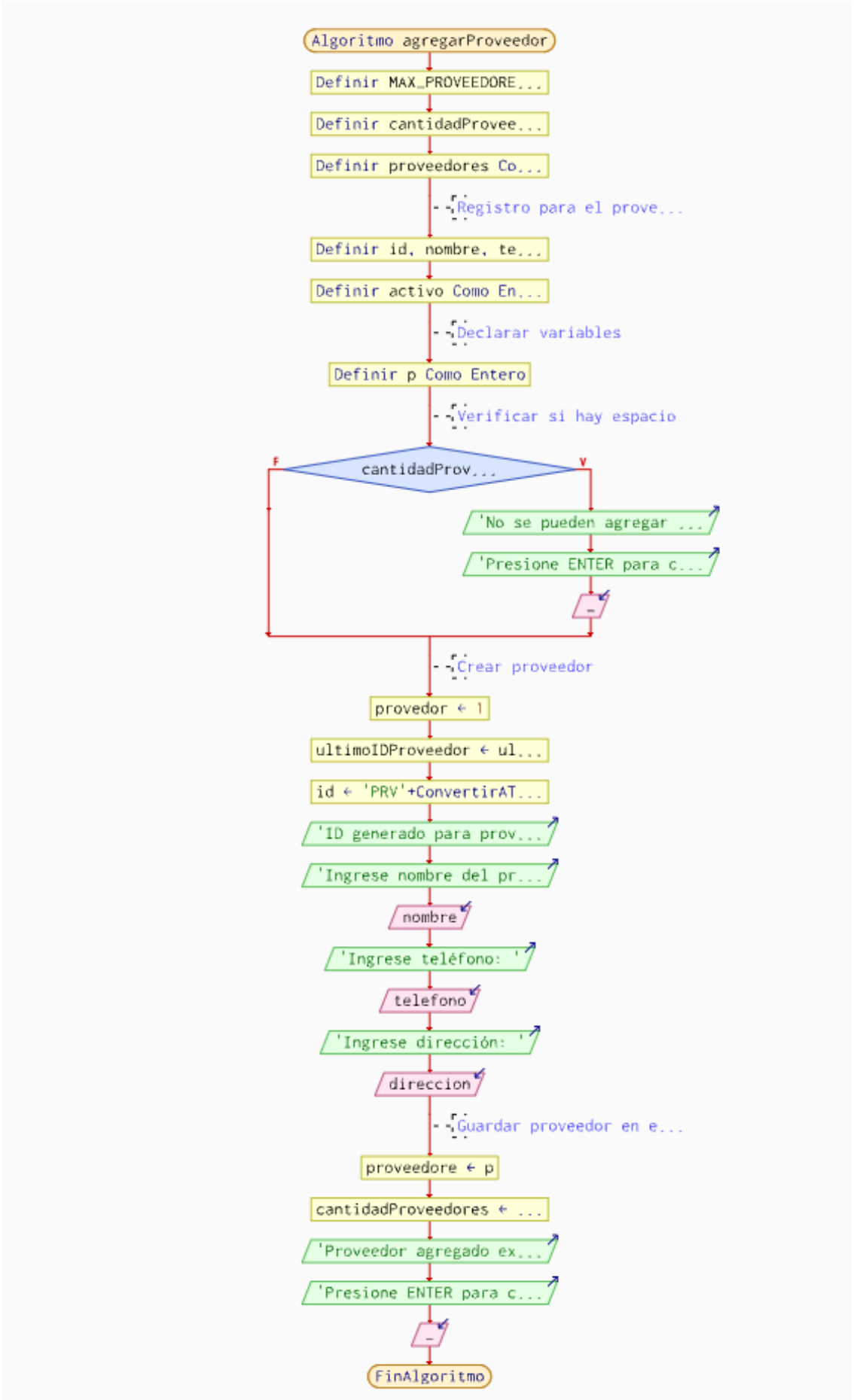
- V(G) = A – N + 2
  V(G)= 49 – 36 + 2= 10

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

Prueba caja blanca de Requisito N° 9: Cómo gestiono proveedores

1. CÓDIGO FUENTE

```
394 void agregarProveedor() {
395     if (cantidadProveedores >= MAX_PROVEEDORES) {
396         printf("No se pueden agregar mas proveedores.\n");
397         pausa();
398         return;
399     }
400     Proveedor p;
401     p.activo = 1;
402 
403     ultimoIDProveedor++;
404     strcpy(p.id, generarID(ultimoIDProveedor - 1));
405     printf("ID generado para proveedor: %s\n", p.id);
406 
407     printf("Ingrese nombre del proveedor: ");
408     fgets(p.nombre, sizeof(p.nombre), stdin);
409     p.nombre[strcspn(p.nombre, "\n")] = '\0';
410     convertirMayusculas(p.nombre);
411 
412     printf("Ingrese telefono: ");
413     fgets(p.telefono, sizeof(p.telefono), stdin);
414     p.telefono[strcspn(p.telefono, "\n")] = '\0';
415 
416     printf("Ingrese direccion: ");
417     fgets(p.direccion, sizeof(p.direccion), stdin);
418     p.direccion[strcspn(p.direccion, "\n")] = '\0';
419 
420     proveedores[cantidadProveedores++] = p;
421     guardarEnArchivos();
422     printf("Proveedor agregado exitosamente.\n");
423     pausa();
424 }
```


Algoritmo agregarProveedor
Definir MAX_PROVEEDORE...
Definir cantidadProvee...
Definir proveedores Co...
Registro para el prove...
Definir id, nombre, te...
Definir activo Como En...
Declarar variables
Definir p Como Entero
Verificar si hay espacio
cantidadProv...
F
V
'No se pueden agregar ...
'Presione ENTER para c...
_
Crear proveedor
provedor ← 1
ultimoIDProveedor ← ul...
id ← 'PRV'+ConvertirAT...
'ID generado para prov...
'Ingrese nombre del pr...
nombre
'Ingrese teléfono: '
telefono
'Ingrese dirección: '
direccion
Guardar proveedor en e...
proveedore ← p
cantidadProveedores ← ...
'Proveedor agregado ex...
'Presione ENTER para c...
_
FinAlgoritmo

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

3. **GRAFO DE FLUJO (GF)**

4. **IDENTIFIACCIÒN DE LAS RUTAS (Camino básico)**

RUTAS

R1: 1-2-3-4-5-6-7-8-9-10-11
R2: 1-2-3-4-6-7-8-9-10-11

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)= 11-11+2= 2

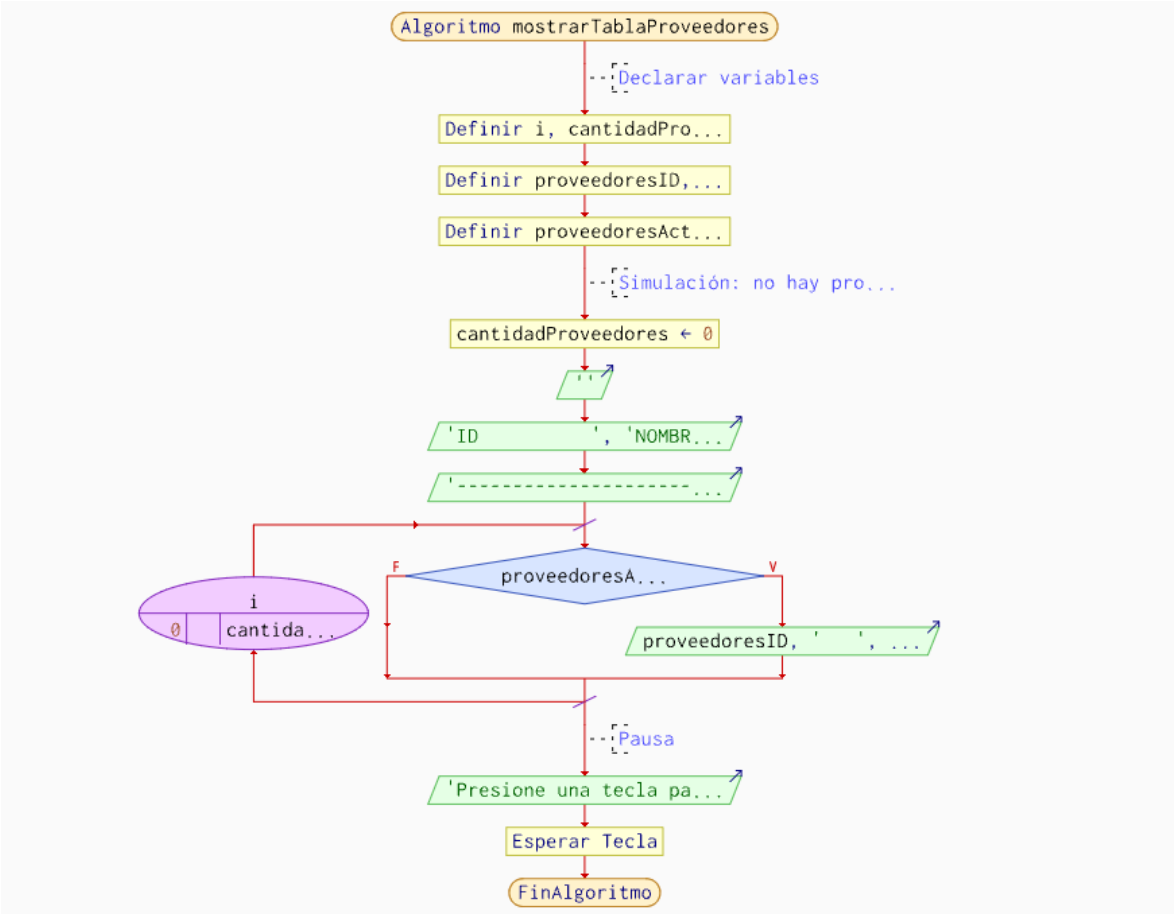
DONDE:
P: Número de nodos predicado
A: Número de aristas
N: Número de nodos

Prueba caja blanca de Requisito N° 10: Cómo veo proveedores

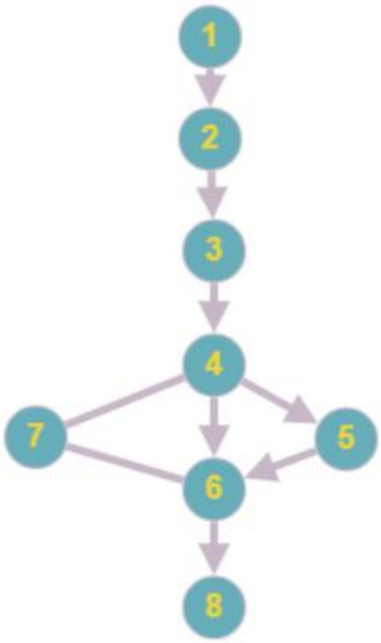
1. **CÓDIGO FUENTE**

```
143 void mostrarTablaProveedores() {
144     printf("\n%-12s%-20s%-20s%-30s\n", "ID", "NOMBRE", "TELEFONO", "DIRECCION");
145     printf("-------------------------------------------------------------------------------------\n");
146     for (int i = 0; i < cantidadProveedores; i++) {
147         if (proveedores[i].activo) {
148             printf("%-12s%-20s%-20s%-30s\n",
149                    proveedores[i].id,
150                    proveedores[i].nombre,
151                    proveedores[i].telefono,
152                    proveedores[i].direccion);
153         }
154     }
155     pausa();
156 }
```

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



3. GRAFO DE FLUJO (GF)



4. IDENTIFIACCIÒN DE LAS RUTAS (Camino básico)

RUTAS

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

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)=
  7- 7+2= 2

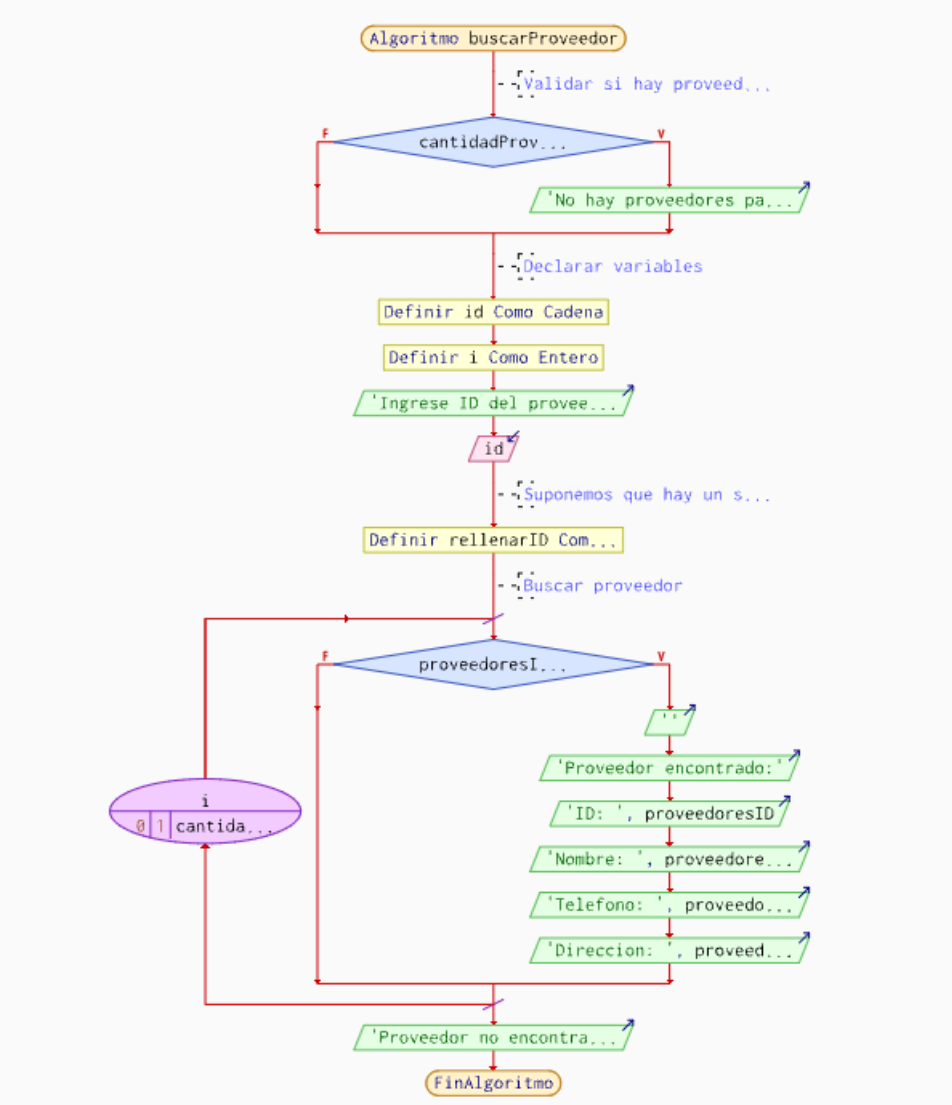
DONDE:
P: Número de nodos predicado
A: Número de aristas
N: Número de nodos

Prueba caja blanca de Requisito N° 11: Cómo busco proveedores

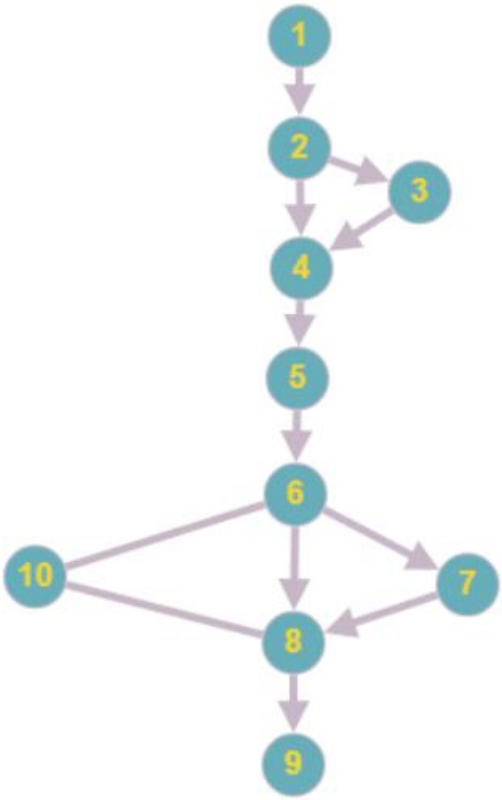
1. CÓDIGO FUENTE

```
525 void buscarProveedor() {
526     if (cantidadProveedores == 0) {
527         printf("No hay proveedores para buscar.\n");
528         pausa();
529         return;
530     }
531     char id[20];
532     printf("Ingrese ID del proveedor a buscar (numeros, se rellenara a 10 digitos): ");
533     fgets(id, sizeof(id), stdin);
534     id[strcspn(id, "\n")] = '\0';
535     rellenarID(id);
536 
537     for (int i = 0; i < cantidadProveedores; i++) {
538         if (strcmp(proveedores[i].id, id) == 0 && proveedores[i].activo) {
539             printf("\nProveedor encontrado:\n");
540             printf("ID: %s\n", proveedores[i].id);
541             printf("Nombre: %s\n", proveedores[i].nombre);
542             printf("Telefono: %s\n", proveedores[i].telefono);
543             printf("Direccion: %s\n", proveedores[i].direccion);
544             pausa();
545             return;
546         }
547     }
548     printf("Proveedor no encontrado.\n");
549     pausa();
550 }
```

2. DIAGRAMA DE FLUJO (DF) PSEINT



3. GRAFO DE FLUJO (GF)



4. IDENTIFIACCIÒN DE LAS RUTAS (Camino básico)

RUTAS

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

5. COMPLEJIDAD CICLOMÁTICA

Se puede calcular de las siguientes formas:

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

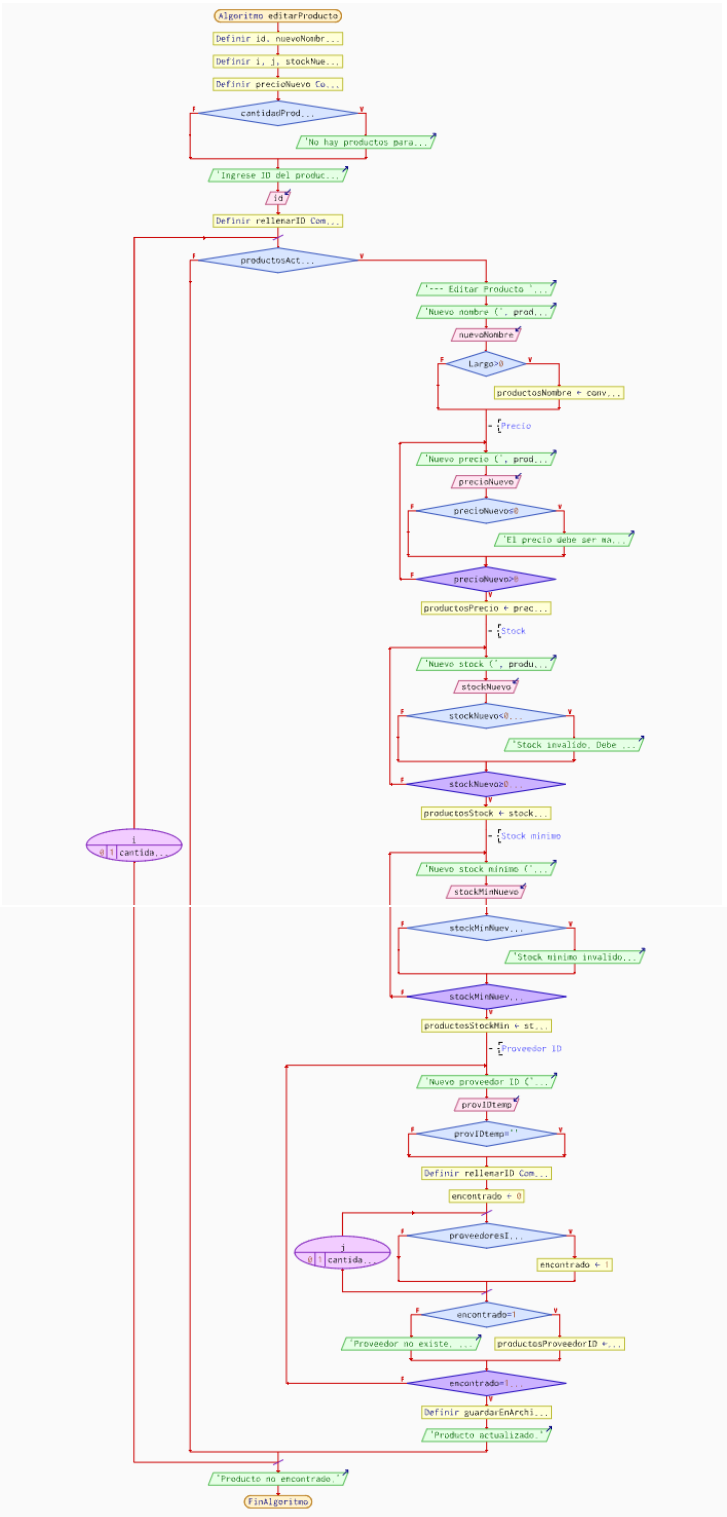
- V(G) = A – N + 2
  V(G)= 10 – 9 + 2= 3

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

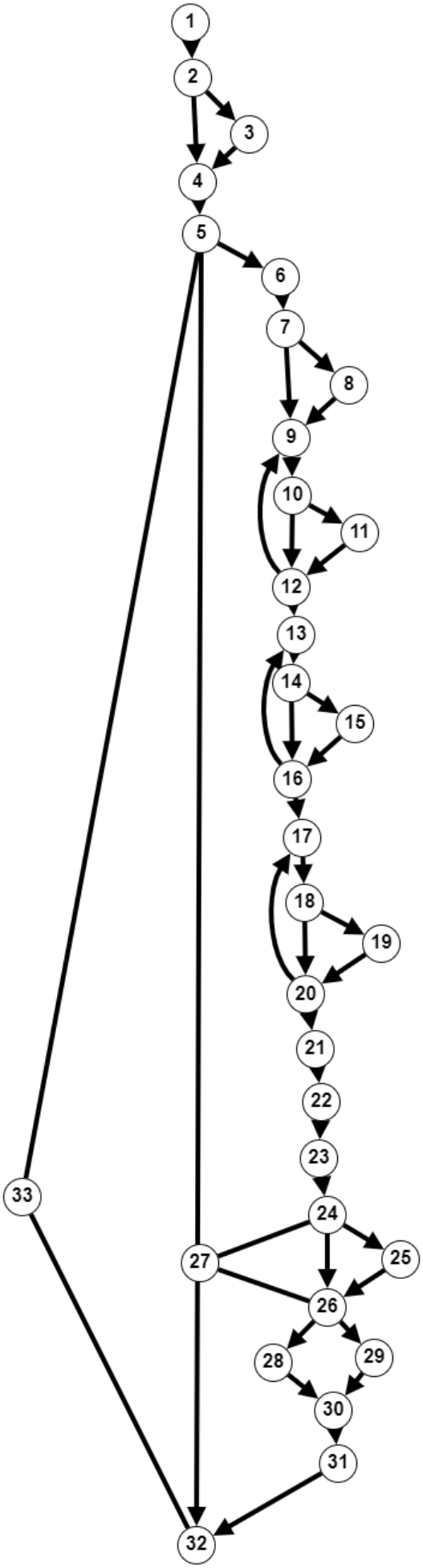
Prueba caja blanca de Requisito N° 12: Cómo edito proveedores

1. CÓDIGO FUENTE

```
void editarProducto() {
    if (cantidadProductos == 0) {
        printf("No hay productos para editar.\n");
        pausa();
        return;
    }
    char id[20];
    printf("Ingrese ID del producto a editar (numeros, se rellenara a 10 digitos): ");
    fgets(id, sizeof(id), stdin);
    id[strcspn(id, "\n")] = '\0';
    rellenarID(id);

    for (int i = 0; i < cantidadProductos; i++) {
        if (strcmp(productos[i].id, id) == 0 && productos[i].activo) {
            printf("\n--- Editar Producto %s ---\n", productos[i].id);

            printf("Nuevo nombre (%s): ", productos[i].nombre);
            fgets(productos[i].nombre, sizeof(productos[i].nombre), stdin);
            productos[i].nombre[strcspn(productos[i].nombre, "\n")] = '\0';
            convertirMayusculas(productos[i].nombre);

            // Precio
            float precioNuevo;
            do {
                printf("Nuevo precio (%.2f): ", productos[i].precio);
                if (scanf("%f", &precioNuevo) != 1) {
                    printf("Entrada invalida. Intente de nuevo.\n");
                    limpiarBuffer();
                    continue;
                }
                if (precioNuevo <= 0) {
                    printf("El precio debe ser mayor a cero.\n");
                    limpiarBuffer();
                       continue;
                   }
                  limpiarBuffer();
                  break;
            } while (1);
            productos[i].precio = precioNuevo;

            // Stock
            int stockNuevo;
            do {
                printf("Nuevo stock (%d): ", productos[i].stock);
                if (scanf("%d", &stockNuevo) != 1) {
                    printf("Entrada invalida. Intente de nuevo.\n");
                    limpiarBuffer();
                    continue;
                }
                if (stockNuevo < 0 || stockNuevo > 1000) {
                    printf("Stock invalido. Debe estar entre 0 y 1000.\n");
                    limpiarBuffer();
                    continue;
                }
                limpiarBuffer();
                break;
            } while (1);
            productos[i].stock = stockNuevo;

            // Stock mínimo
            int stockMinNuevo;
            do {
                printf("Nuevo stock minimo (%d): ", productos[i].stockMinimo);
                if (scanf("%d", &stockMinNuevo) != 1) {
                    printf("Entrada invalida. Intente de nuevo.\n");
                    limpiarBuffer();
                   continue;
                }
                if (stockMinNuevo < 0 || stockMinNuevo > productos[i].stock) {
                    printf("Stock minimo invalido. Debe estar entre 0 y stock actual.\n");
                    limpiarBuffer();
                    continue;
                }
                limpiarBuffer();
                break;
            } while (1);
            productos[i].stockMinimo = stockMinNuevo;

            // Proveedor ID
            char provIDtemp[20];
            do {
                printf("Nuevo proveedor ID (%s): ", productos[i].proveedorID);
                fgets(provIDtemp, sizeof(provIDtemp), stdin);
                provIDtemp[strcspn(provIDtemp, "\n")] = '\0';

                if(strlen(provIDtemp) == 0) {
                    // Si no cambia, dejar igual
                    break;
                }

                rellenarID(provIDtemp);

                int encontrado = 0;
                for (int j = 0; j < cantidadProveedores; j++) {
                    if (strcmp(proveedores[j].id, provIDtemp) == 0 && proveedores[j].activo) {
                        encontrado = 1;
                        break;
                    }
                }
                if (!encontrado) {
                    printf("Proveedor no existe. Intente de nuevo.\n");
                } else {
                    strcpy(productos[i].proveedorID, provIDtemp);
                    break;
                }
            } while (1);

            guardarEnArchivos();
            printf("Producto actualizado.\n");
            pausa();
            return;
        }
    }
    printf("Producto no encontrado.\n");
    pausa();
}
```



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


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33

3. **GRAFO DE FLUJO (GF)**

4. **IDENTIFIACCIÒN DE LAS RUTAS (Camino básico)**

RUTAS

RF1: 1-2-4-5-6-7-8-9-10-12-13-14-16-17-18-20-21-22-23-24-26-28-30-31-32
RF2: 1-2-4-5-6-7-8-9-10-12-13-14-16-17-18-20-21-22-23-24-26-29-30-31-32
RF3: 1-2-4-5-6-7-8-9-10-12-13-14-16-17-18-20-21-22-23-24-25-26-28-30-31-32
RF4: 1-2-4-5-6-7-8-9-10-12-13-14-16-17-18-20-21-22-23-24-25-26-29-30-31-32
RF5: 1-2-4-5-6-7-8-9-10-12-13-14-15-14-16-17-18-20-21-22-23-24-26-28-30-31-32
RF6: 1-2-4-5-6-7-8-9-10-12-13-14-15-14-16-17-18-20-21-22-23-24-26-29-30-31-32
RF7: 1-2-4-5-6-7-8-9-10-12-13-14-15-14-16-17-18-20-21-22-23-24-25-26-28-30-31-32
RF8: 1-2-4-5-6-7-8-9-10-12-13-14-15-14-16-17-18-20-21-22-23-24-25-26-29-30-31-32
RF9: 1-2-4-5-6-7-8-9-10-12-13-14-16-17-18-19-18-20-21-22-23-24-26-28-30-31-32
RF10: 1-2-4-5-6-7-8-9-10-12-13-14-16-17-18-19-18-20-21-22-23-24-26-29-30-31-32
RF11: 1-2-4-5-6-7-8-9-10-12-13-14-16-17-18-19-18-20-21-22-23-24-25-26-28-30-31-32
RF12: 1-2-4-5-6-7-8-9-10-12-13-14-16-17-18-19-18-20-21-22-23-24-25-26-29-30-31-32
RF13: 1-2-4-5-6-7-8-9-10-12-13-14-15-14-16-17-18-19-18-20-21-22-23-24-26-28-30-31-32
RF14: 1-2-4-5-6-7-8-9-10-12-13-14-15-14-16-17-18-19-18-20-21-22-23-24-26-29-30-31-32

5. **COMPLEJIDAD CICLOMÁTICA**

Se puede calcular de las siguientes formas:

- V(G) = número de nodos predicados(decisiones)+1
  V(G)= 13+1= 14

- V(G) = A – N + 2
  V(G)=43-31+2= 14

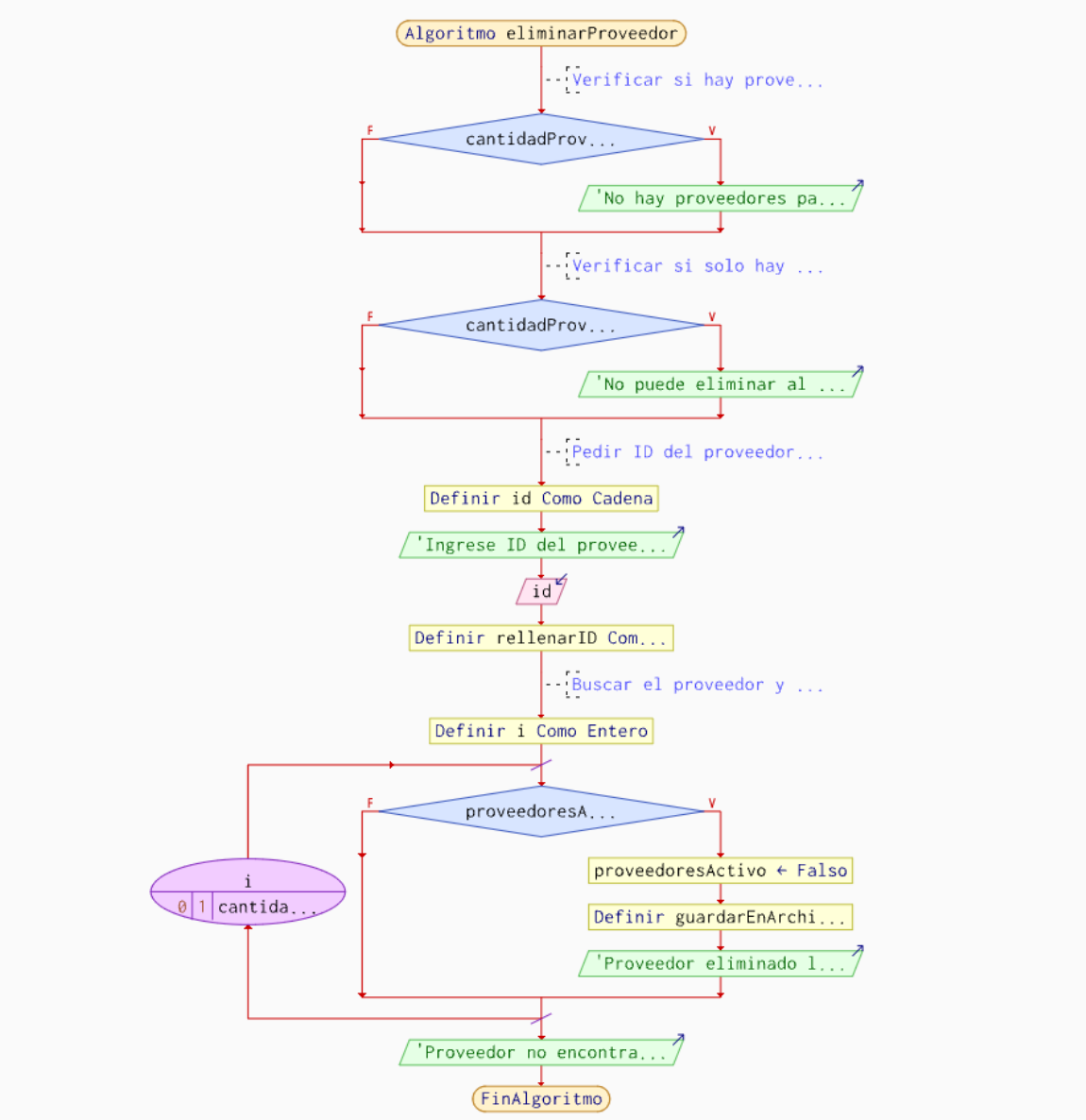
DONDE:
P: Número de nodos predicado
A: Número de aristas
N: Número de nodos

Prueba caja blanca de Requisito N° 13: Cómo elimino proveedores

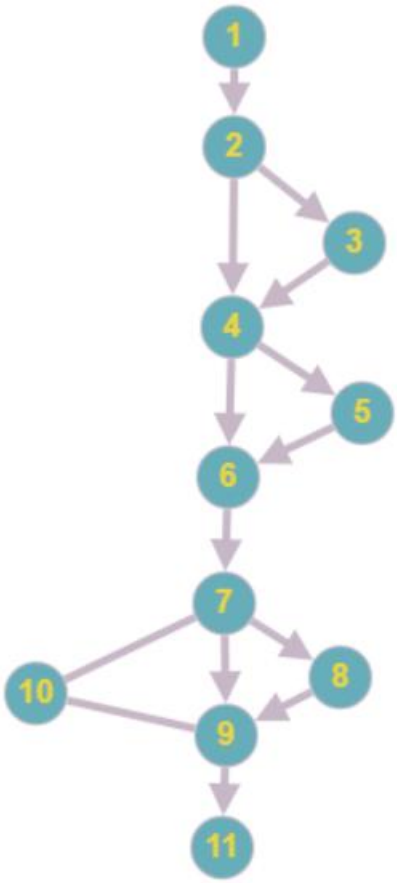
1. **CÓDIGO FUENTE**

```c
void eliminarProveedor() {
    if (cantidadProveedores == 0) {
        printf("No hay proveedores para eliminar.\n");
        pausa();
        return;
    }
    if (cantidadProveedores == 1) {
        printf("No puede eliminar al unico proveedor existente.\n");
        pausa();
        return;
    }

    char id[20];
    printf("Ingrese ID del proveedor a eliminar (numeros, se rellenara a 10 digitos): ");
    fgets(id, sizeof(id), stdin);
    id[strcspn(id, "\n")] = '\0';
    rellenarID(id);

    for (int i = 0; i < cantidadProveedores; i++) {
        if (strcmp(proveedores[i].id, id) == 0 && proveedores[i].activo) {
            proveedores[i].activo = 0;
            guardarEnArchivos();
            printf("Proveedor eliminado logicamente.\n");
            pausa();
            return;
        }
    }
    printf("Proveedor no encontrado.\n");
    pausa();
}
```

2. DIAGRAMA DE FLUJO (DF) PSEINT



3. **GRAFO DE FLUJO (GF)**



4. **IDENTIFIACCIÒN DE LAS RUTAS (Camino básico)**

RUTAS

R1: 1-2-3-4-5-6-7-9-11
R2: 1-2-4-5-6-7-9-11
R3: 1-2-3-4-5-6-7-8-11
R4: 1-2-3-4-5-6-7-8-9-11

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)=12 – 10+2= 4

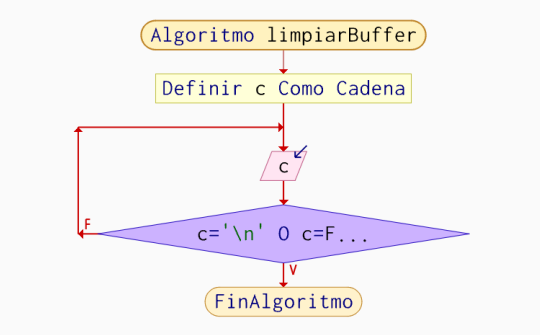
DONDE:
P: Número de nodos predicado
A: Número de aristas
N: Número de nodos

Prueba caja blanca de Requisito N° 14: Cómo mejorar la interfaz

1. **CÓDIGO FUENTE**

```
67  void limpiarBuffer() {
68      int c;
69      while ((c = getchar()) != '\n' && c != EOF);
70  }
71
```

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



3. **GRAFO DE FLUJO (GF)**

4. **IDENTIFIACCIÒN DE LAS RUTAS (Camino básico)**

RUTAS

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

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

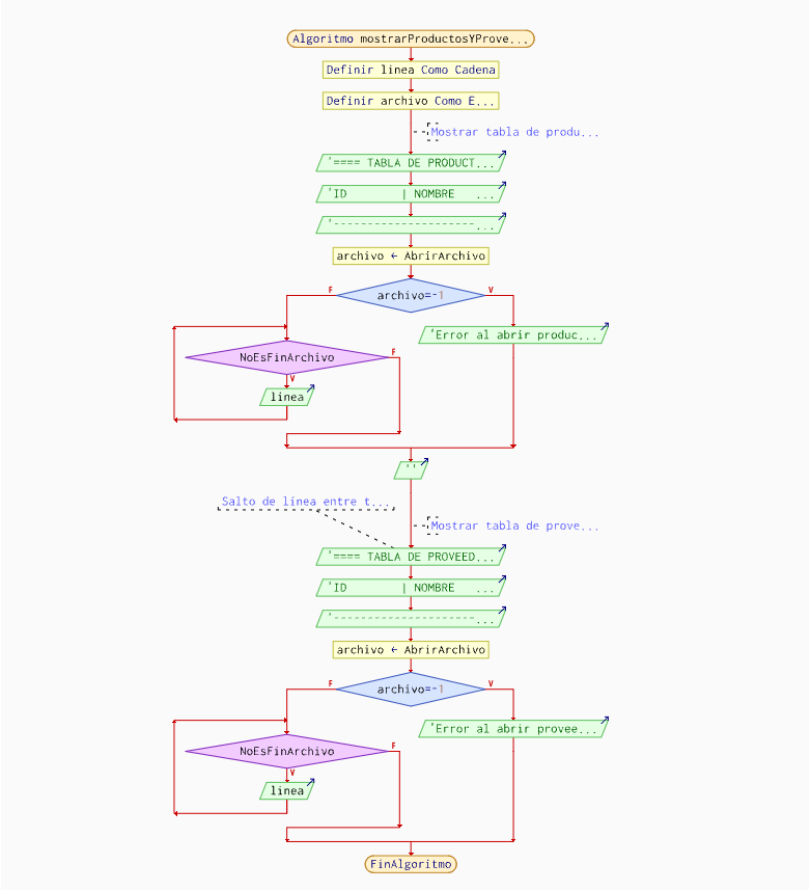
DONDE:
P: Número de nodos predicado
A: Número de aristas
N: Número de nodos

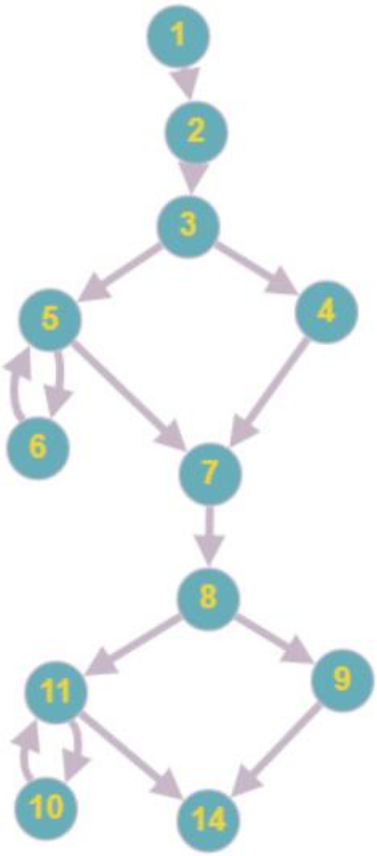
Prueba caja blanca de Requisito N° 15: Cómo guardar cambios

1. **CÓDIGO FUENTE**

```
552 void guardarEnArchivos() {
553     FILE *fp;
554 
555     // Guardar productos
556     fp = fopen("productos.txt", "w");
557     if (fp == NULL) {
558         printf("Error al abrir archivo productos.txt\n");
559         return;
560     }
561     for (int i = 0; i < cantidadProductos; i++) {
562         if (productos[i].activo) {
563             fprintf(fp, "%s|%s|%.2f|%d|%d|%s\n",
564                     productos[i].id,
565                     productos[i].nombre,
566                     productos[i].precio,
567                     productos[i].stock,
568                     productos[i].stockMinimo,
569                     productos[i].proveedorID);
570         }
571     }
572     fclose(fp);
573 
574     // Guardar proveedores
575     fp = fopen("proveedores.txt", "w");
576     if (fp == NULL) {
577         printf("Error al abrir archivo proveedores.txt\n");
578         return;
579     }
580     for (int i = 0; i < cantidadProveedores; i++) {
581         if (proveedores[i].activo) {
582             fprintf(fp, "%s|%s|%s|%s\n",
583                     proveedores[i].id,
584                     proveedores[i].nombre,
585                     proveedores[i].telefono,
586                     proveedores[i].direccion);
587         }
588     }
589     fclose(fp);
590 }
```

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


Algoritmo mostrarProductosYProve...
Definir linea Como Cadena
Definir archivo Como E...
Mostrar tabla de produ...
'==== TABLA DE PRODUCT...
'ID | NOMBRE ...
'---------- ...
archivo ← AbrirArchivo
archivo=-1
F
V
'Error al abrir produc...
NoEsFinArchivo
F
V
linea
''
Salto de linea entre t...
Mostrar tabla de prove...
'==== TABLA DE PROVEED...
'ID | NOMBRE ...
'---------- ...
archivo ← AbrirArchivo
archivo=-1
F
V
'Error al abrir provee...
NoEsFinArchivo
F
V
linea
FinAlgoritmo


1
2
3
4
5
6
7
8
9
10
11
14

3. GRAFO DE FLUJO (GF)

4. IDENTIFIACCIÒN DE LAS RUTAS (Camino básico)

RUTAS

R1: 1-2-3-4-5-6-7-8-9-14
R2: 1-2-3-4-7-8-11-10-11-14
R3: 1-2-3-5-6-5-7-8-9-14
R4: 1-2-3-4-5-6-5-7-8-11-10-11-14
R5: 1-2-3-5-6-5-7-8-9-14

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)=15-12+2=5

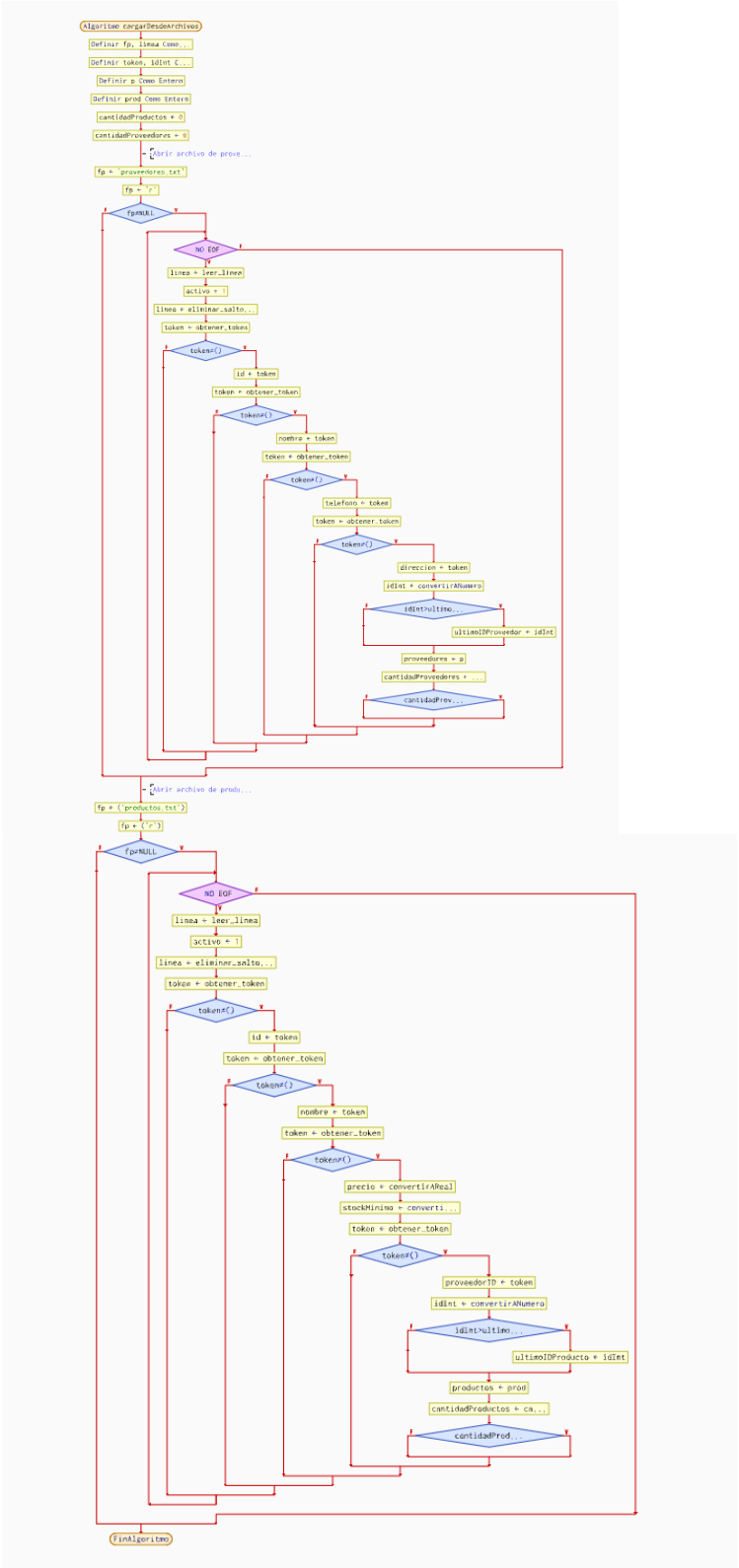
DONDE:
P: Número de nodos predicado
A: Número de aristas
N: Número de nodos

Prueba caja blanca de Requisito N° 16: Cómo cargar datos al iniciar

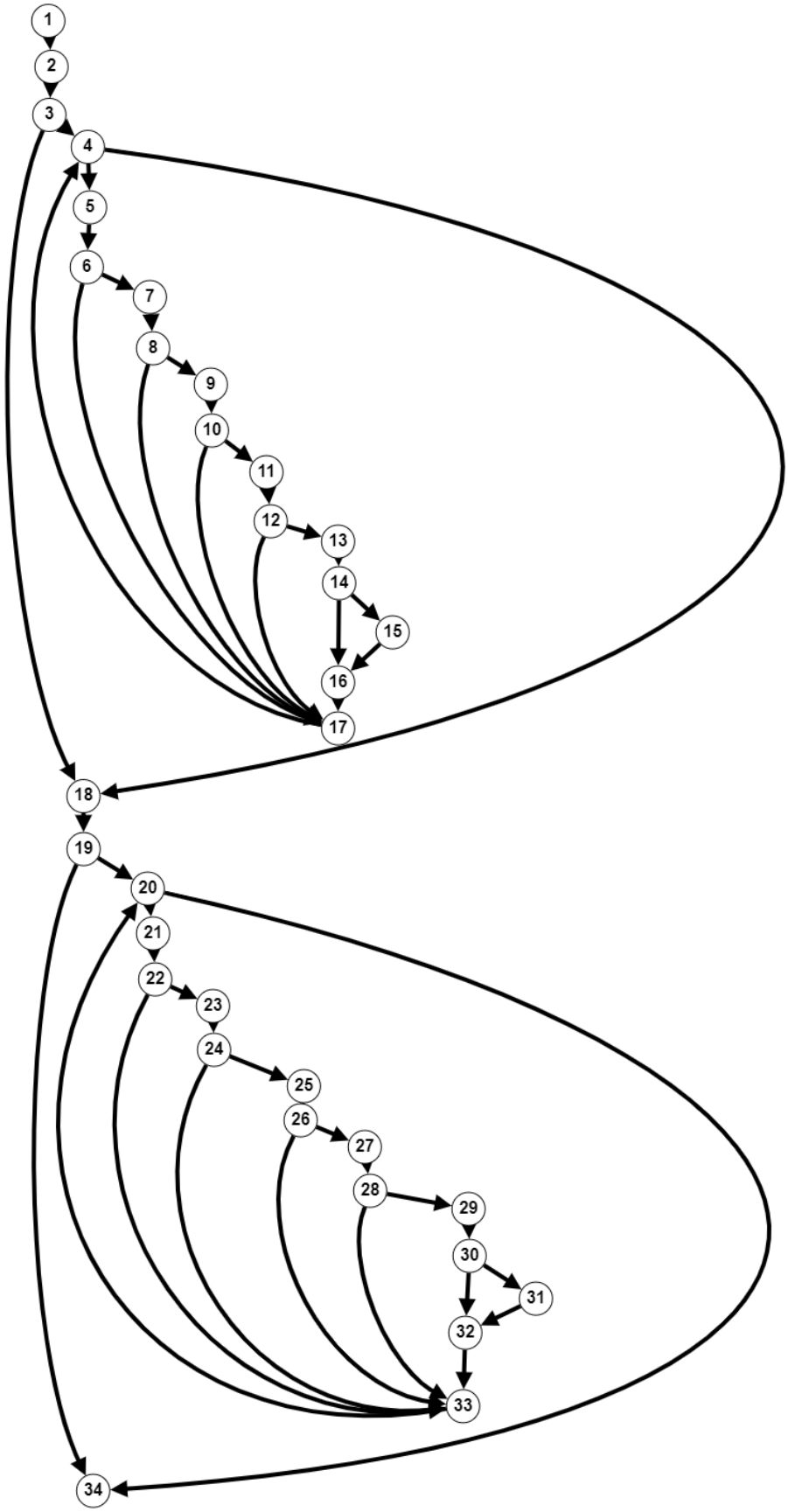
1. CÓDIGO FUENTE

```
592  void cargarDesdeArchivos() {
593      FILE *fp;
594      char linea[256];
595
596      cantidadProductos = 0;
597      cantidadProveedores = 0;
598
599      // Cargar proveedores primero (para validar al agregar productos)
600      fp = fopen("proveedores.txt", "r");
601      if (fp != NULL) {
602          while (fgets(linea, sizeof(linea), fp)) {
603              Proveedor p;
604              p.activo = 1;
605              linea[strcspn(linea, "\n")] = '\0';
606              char *token = strtok(linea, "|");
607              if (!token) continue;
608              strcpy(p.id, token);
609              token = strtok(NULL, "|");
610              if (!token) continue;
611              strcpy(p.nombre, token);
612              token = strtok(NULL, "|");
613              if (!token) continue;
614              strcpy(p.telefono, token);
615              token = strtok(NULL, "|");
616              if (!token) continue;
617              strcpy(p.direccion, token);
618
619              // Actualizar ultimoIDProveedor
620              int idInt = atoi(p.id);
621              if (idInt > ultimoIDProveedor) ultimoIDProveedor = idInt;
622
623              proveedores[cantidadProveedores++] = p;
624              if (cantidadProveedores >= MAX_PROVEEDORES) break;
625          }
626          fclose(fp);
627      }
628
629      // Cargar productos
630      fp = fopen("productos.txt", "r");
631      if (fp != NULL) {
632          while (fgets(linea, sizeof(linea), fp)) {
633              Producto p;
634              p.activo = 1;
635              linea[strcspn(linea, "\n")] = '\0';
636             char *token = strtok(linea, "|");
637             if (!token) continue;
638             strcpy(p.id, token);
639             token = strtok(NULL, "|");
640             if (!token) continue;
641             strcpy(p.nombre, token);
642             token = strtok(NULL, "|");
643             if (!token) continue;
644             p.precio = atof(token);
645             token = strtok(NULL, "|");
646             if (!token) continue;
647             p.stock = atoi(token);
648             token = strtok(NULL, "|");
649             if (!token) continue;
650             p.stockMinimo = atoi(token);
651             token = strtok(NULL, "|");
652             if (!token) continue;
653             strcpy(p.proveedorID, token);
654
655             // Actualizar ultimoIDProducto
656             int idInt = atoi(p.id);
657             if (idInt > ultimoIDProducto) ultimoIDProducto = idInt;
658
659             productos[cantidadProductos++] = p;
660             if (cantidadProductos >= MAX_PRODUCTOS) break;
661         }
662         fclose(fp);
663     }
664 }
665
```

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



3. GRAFO DE FLUJO (GF)



4. IDENTIFIACCIÒN DE LAS RUTAS (Camino básico) RUTAS

R1: 1-2-3-4-18-19-20-21-22-33-20-34
R2: 1-2-3-4-5-6-17-4-18-19-20-34
R3: 1-2-3-4-5-6-7-8-18-19-20-34
R4: 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-34
R5: 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-21-22-23-24-25-26-27-28-29-30-31-32-33-34
R6: 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-21-22-23-24-25-26-27-28-29-30-32-33-34
R7: 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-21-22-23-24-25-26-27-28-29-30-31-32-33-34
R8: 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-21-22-23-24-25-26-27-28-29-30-31-32-33-34
R9: 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-21-22-23-24-25-26-27-28-29-30-32-33-34
R10: 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-21-22-23-24-25-26-27-28-29-30-32-33-34
R11: 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-21-22-23-24-25-26-27-28-29-30-32-33-34
R12: 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-21-22-23-24-25-26-27-28-29-30-32-33-34
R13: 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-21-22-23-24-25-26-27-28-29-30-31-32-33-34
R14: 1-2-3-4-5-6-7-8-18-19-21-22-23-24-25-26-27-28-29-30-32-33-34
R15: 1-2-3-4-5-6-18-19-21-22-23-24-25-26-27-28-29-30-32-33-34
R16: 1-2-3-4-5-6-18-19-21-22-23-24-25-26-27-28-29-30-31-32-33-34
R17: 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-21-22-23-24-25-26-27-28-29-30-31-32-33-34

5. COMPLEJIDAD CICLOMÁTICA

Se puede calcular de las siguientes formas:

- V(G) = número de nodos predicados(decisiones)+1 V(G)= 16+1: 17
- V(G) = A – N + 2 V(G)=49 – 34 + 2= 17

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