

Prueba de Caja Blanca

“Sistema automatizado de control de Inventario”

Versión 2.0

Integrantes:

Caizapanta Muela Tammy Amarilis
Guaiguacundo Aguirre Valeria Naomi
Pincha Llanos Estefany Anahi
Robalino Zaldumbide Alejandro Benjamín

Fecha

2025-01-21

1. REQ 001 - Agregar productos

1.1. CÓDIGO FUENTE

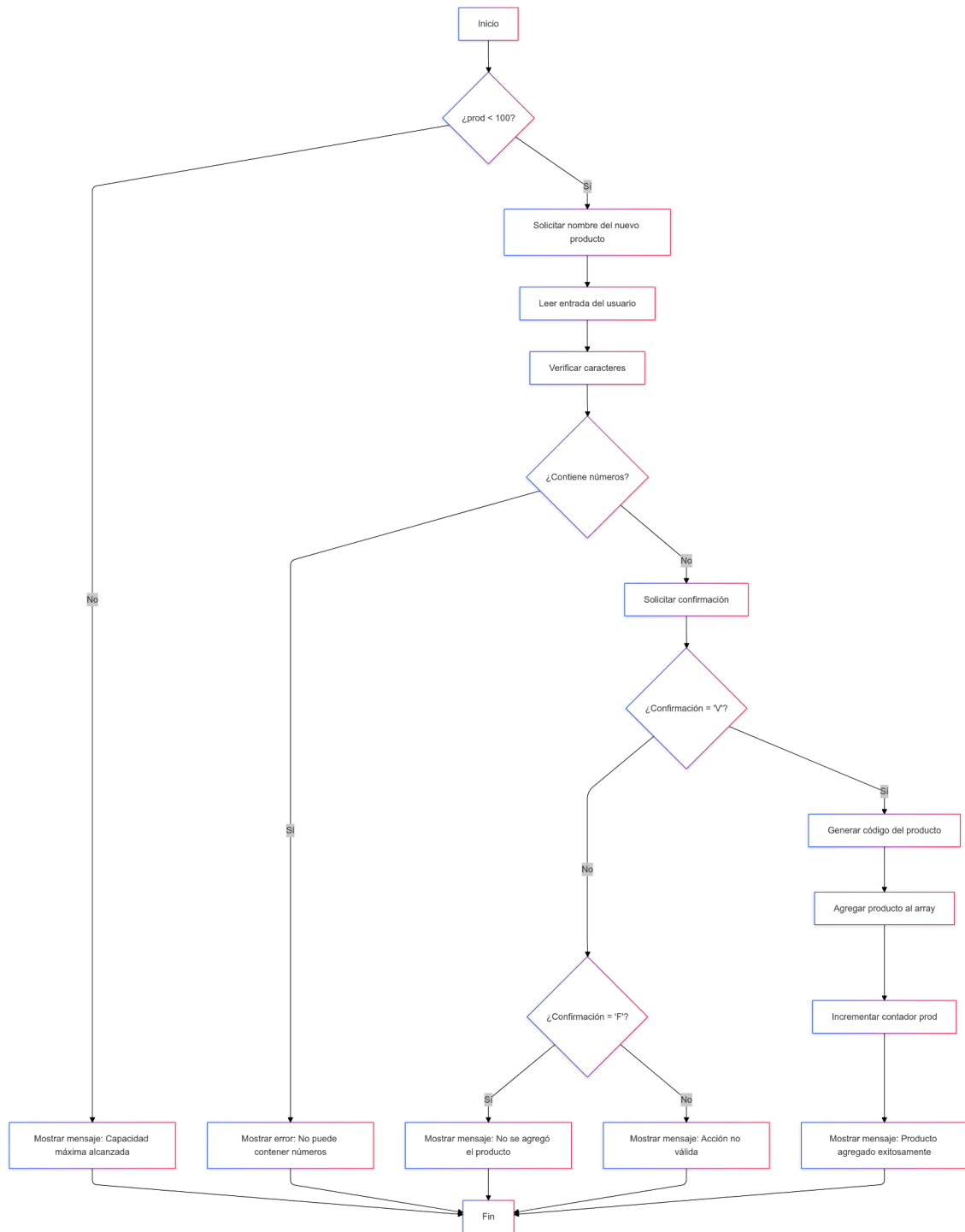
```
// función para agregar productos
void agregar_producto() {
    if (prod < 100) {
        char nuevo_producto[50];
        char confirmar;

        printf("Ingrese el nombre del nuevo producto: ");
        getchar();
        fgets(nuevo_producto, 50, stdin);

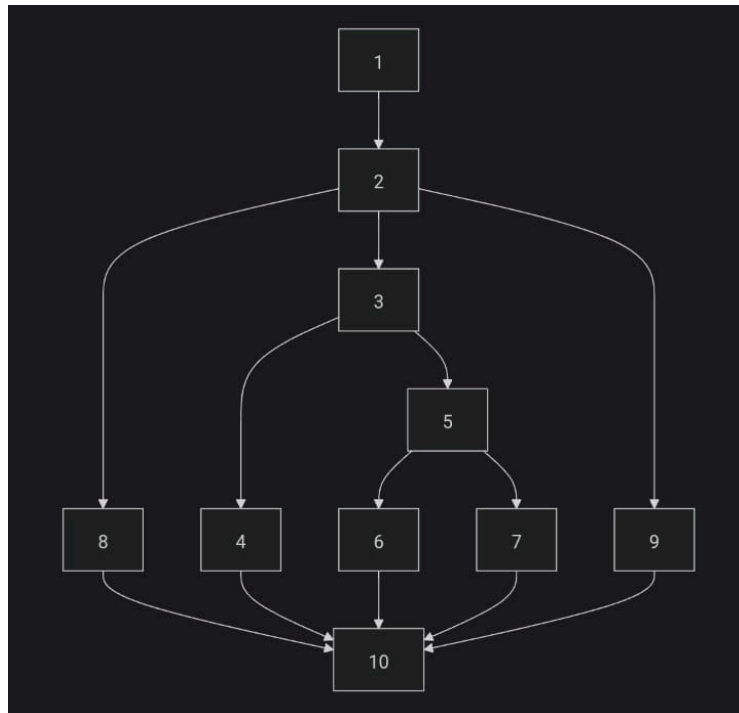
        nuevo_producto[strcspn(nuevo_producto, "\n")] = '\0';
        for (int i = 0; nuevo_producto[i] != '\0'; i++) {
            if (isdigit(nuevo_producto[i])) {
                printf("Error: El nombre del producto no puede contener numeros.\n");
                return;
            }
        }
        printf("Esta seguro de que desea agregar el producto '%s' (V/F): ", nuevo_producto);
        scanf(" %c", &confirmar);
        confirmar = toupper(confirmar);

        if (confirmar == 'V') {
            sprintf(producto[prod].product, "%03d.%s", prod + 1, nuevo_producto);
            prod++;
            printf("Producto agregado exitosamente.\n");
        } else if (confirmar == 'F') {
            printf("No se agrego el producto.\n");
        } else {
            printf("Accion no valida. \n");
        }
    } else {
        printf("No se pueden agregar mas productos. Capacidad maxima alcanzada.\n");
    }
}
```

1.2. DIAGRAMA DE FLUJO (MERMAID)



1.3. GRAFO



1.3.1. RUTAS

R1: $1 \rightarrow 2 \rightarrow 3 \rightarrow 7$ (confirmar == 'V')

R2: $1 \rightarrow 2 \rightarrow 4 \rightarrow 7$ (confirmar == 'F')

R3: $1 \rightarrow 2 \rightarrow 5 \rightarrow 7$ (confirmar no es ni 'V' ni 'F')

R4: $1 \rightarrow 2 \rightarrow 6 \rightarrow 7$ (capacidad máxima alcanzada)

1.4. COMPLEJIDAD CICLOMÁTICA

Por aristas y nodos:

Aristas (E) = 8

Nodos (N) = 7

$V(G) = E - N + 2P$

$V(G) = 9 - 7 + 2(1)$

$V(G) = 4$

Por nodos predicado:

Nodos predicado = 3

$V(G) = P + 1$

$$V(G) = 3 + 1$$

$$V(G) = 4$$

2. REQ 002 - Buscar productos

2.1. CÓDIGO FUENTE

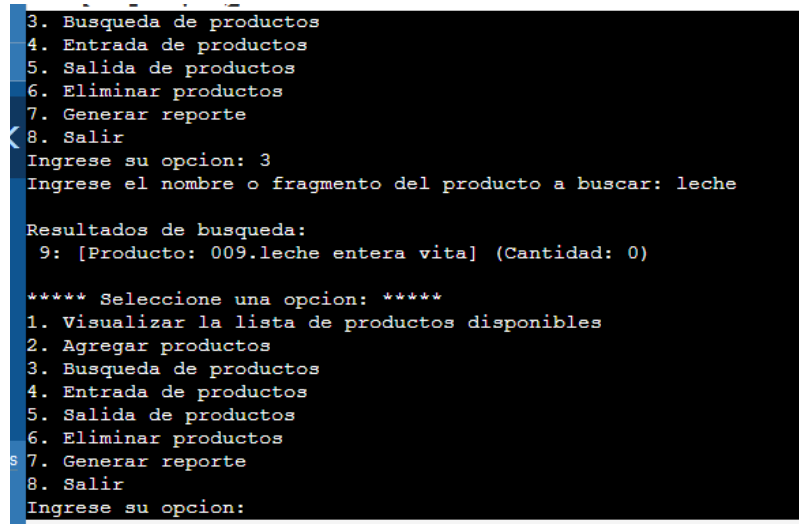
```
// Función para buscar productos
void buscar_producto() {
    char termino[50];
    int encontrado = 0;

    1. printf("Ingrese el nombre o fragmento del producto a buscar: ");
    getchar();
    2. fgets(termino, 50, stdin);

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

    3. printf("\nResultados de busqueda:\n");
    4. for (int i = 0; 5. i < prod; i++) {
        6. if (strstr(producto[i].product, termino) != NULL) {
            7. printf(" %d: [Producto: %s] (Cantidad: %d)\n", i + 1,
                producto[i].product, producto[i].cantidad);
            encontrado = 1;
        }
    }

    8. if (!encontrado) {
        printf("No se encontraron productos que coincidan con '%s'.\n",
            termino);
    }
}
9. Fin
```

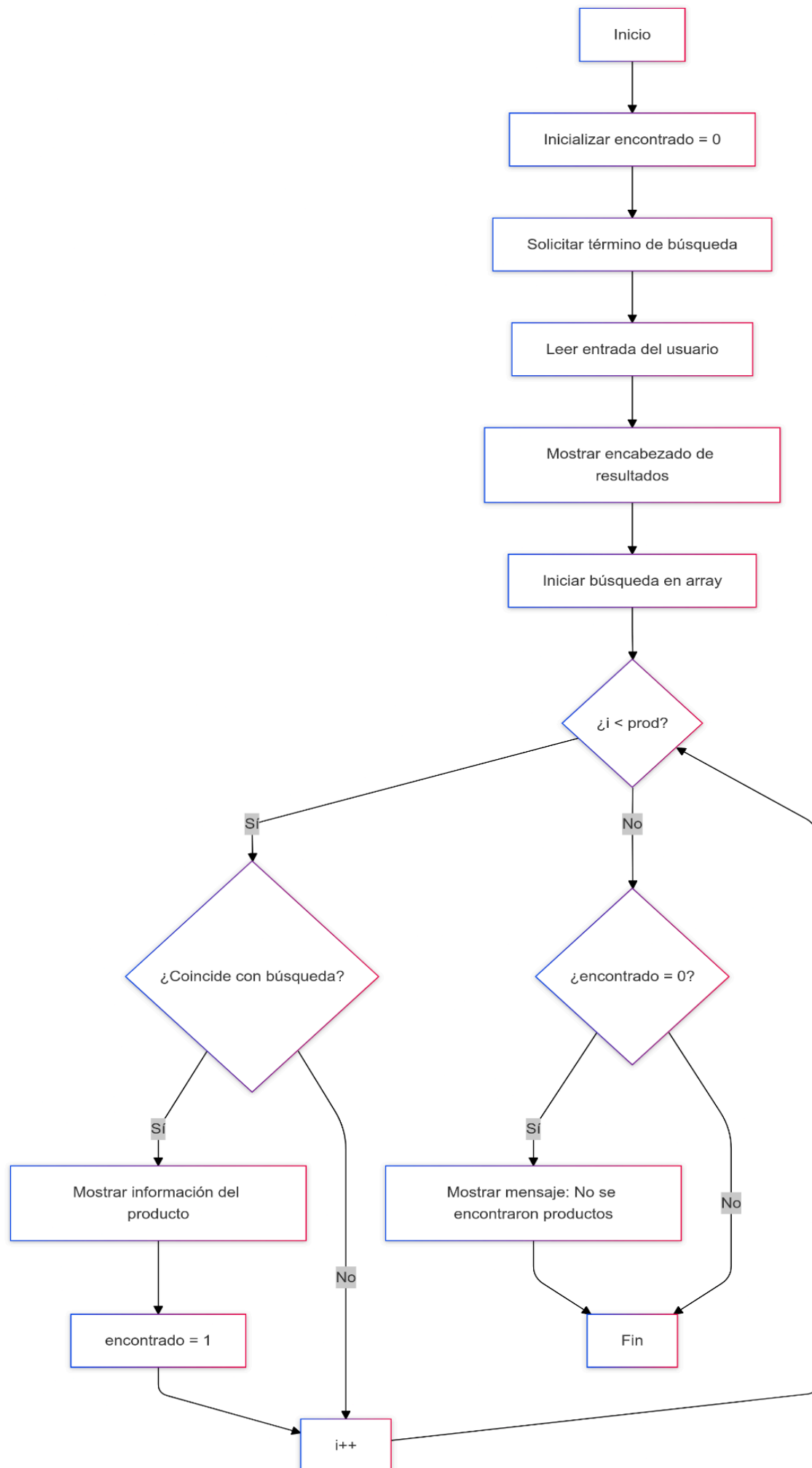


```
3. Busqueda de productos
4. Entrada de productos
5. Salida de productos
6. Eliminar productos
7. Generar reporte
8. Salir
Ingrese su opcion: 3
Ingrese el nombre o fragmento del producto a buscar: leche

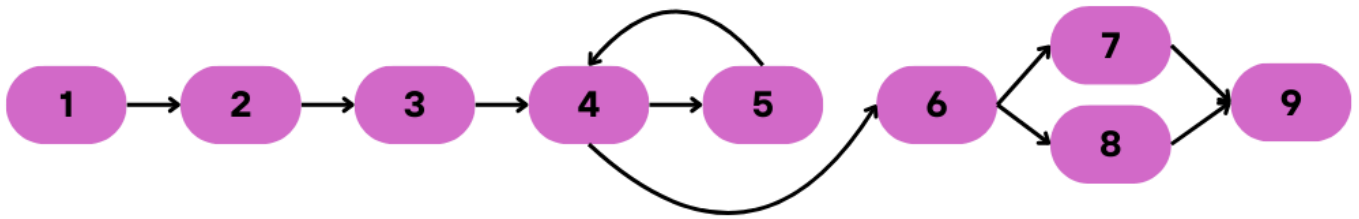
Resultados de busqueda:
9: [Producto: 009.leche entera vita] (Cantidad: 0)

**** Seleccione una opcion: ****
1. Visualizar la lista de productos disponibles
2. Agregar productos
3. Busqueda de productos
4. Entrada de productos
5. Salida de productos
6. Eliminar productos
7. Generar reporte
8. Salir
Ingrese su opcion:
```

2.2. DIAGRAMA DE FLUJO



2.3. GRAFO



2.4. RUTAS

R1: 1→2→3→4→5→6→7→9

R2: 1→2→3→4→5→6→8→9

R3: 1→2→3→4→4→5→6→7→9

2.5. COMPLEJIDAD CICLOMÁTICA

Se puede calcular de las siguientes formas:

- $V(G) = \text{número de nodos predados(decisiones)} + 1$
 $V(G) = 2 + 1 = 3$
- $V(G) = A - N + 2$
 $V(G) = 10 - 9 + 2 = 3$

DONDE:

P: Número de nodos predado

A: Número de aristas

N: Número de nodos

3. REQ 003 - Entrada de productos

3.1. CÓDIGO FUENTE

```
// Función para entrada de productos
void entrada_producto() {
    char termino[50];
    int cantidad;
    int encontrado = 0;
    char confirmar;

    printf("Ingrese el codigo o nombre del producto para aumentar su cantidad:
");
    getchar();
    fgets(termino, 50, stdin);
    termino[strcspn(termino, "\n")] = '\0';

    if (strlen(termino) == 0) {
        printf("Error: No ingreso ningun valor. Intente nuevamente.\n");
        return;
    }
}
```



```

for (int i = 0; i < prod; i++) {
    if (strstr(producto[i].product, termino) != NULL) {
        printf("\nProducto encontrado: %s (Cantidad actual: %d)\n",
producto[i].product, producto[i].cantidad);
        printf("Ingrese la cantidad a agregar: ");
        if (scanf("%d", &cantidad) != 1 || cantidad <= 0) {
            printf("Error: Cantidad invalida. Solo se aceptan numeros
positivos.\n");
            while (getchar() != '\n');
            return;
        }

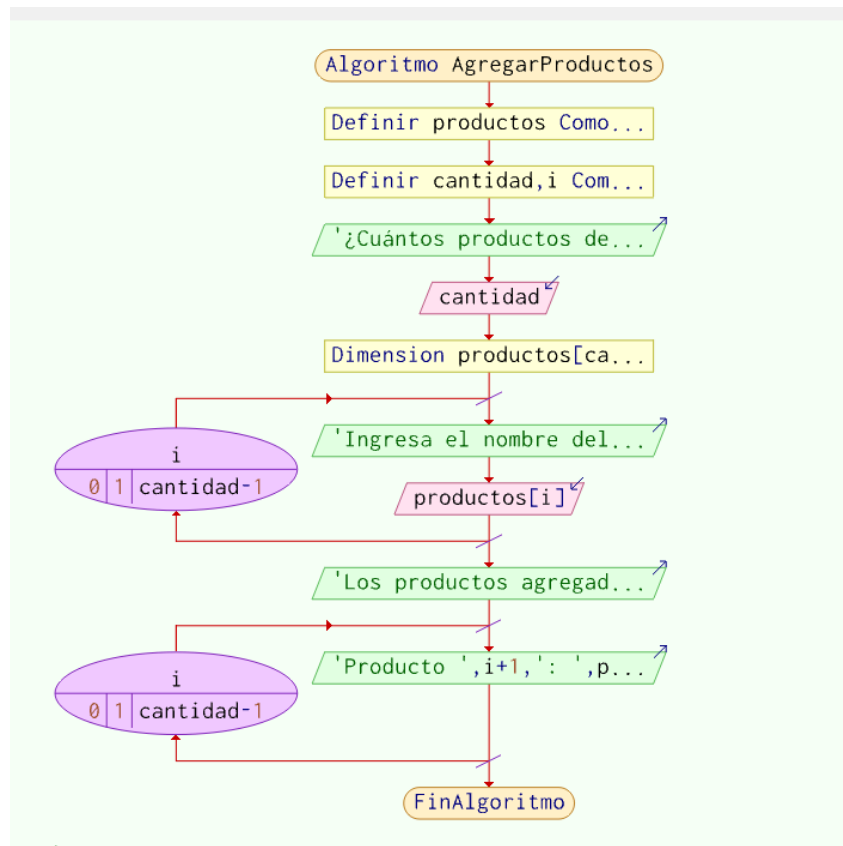
        printf("Esta seguro de que desea agregar %d a '%s' (V/F): ", cantidad,
producto[i].product);
        scanf(" %c", &confirmar);
        confirmar = toupper(confirmar);

        if (confirmar == 'V') {
            producto[i].cantidad += cantidad;
            printf("\nCantidad actualizada. Nueva cantidad: %d\n",
producto[i].cantidad);
        } else if (confirmar == 'F') {
            printf("\n No se realizo ningun cambio.\n");
        } else {
            printf("\nOpcion no valida..\n");
        }

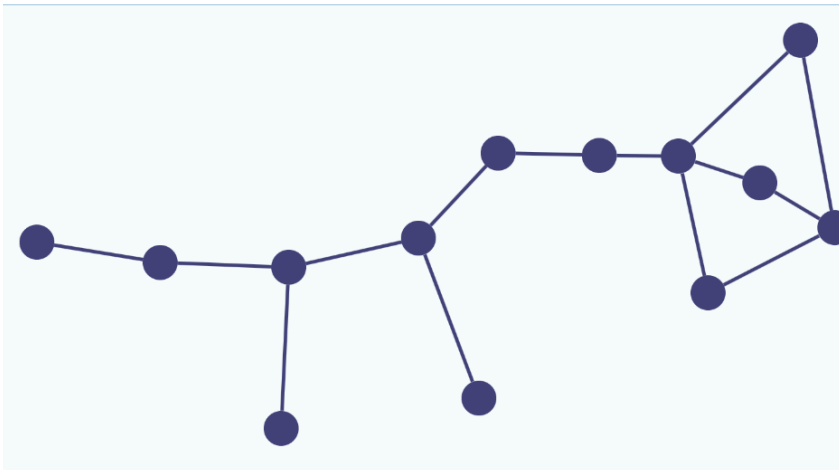
        encontrado = 1;
        break;
    } if (!encontrado) {
        printf("\nNo se encontro el producto con el codigo o nombre '%s'.\n",
termino);
    }
}

```

3.2 DIAGRAMA DE FLUJO



3.2. GRAFO



3.2.1. RUTAS

RUTAS:7

R1: 1 - 2 - 3 - 11

R2: 1 - 2 - 3 - 4 - 5 - 10 - 11

R3: 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 11

R4:1 - 2 - 3 - 4 - 5 - 6 - 7 - 9 - 11

R5:1 - 2 - 3 - 4 - 5 - 6 - 7 - 10 - 11

R6:1 - 2 - 3 - 4 - 5 - 6 - 11

R7:1 - 2 - 3 - 4 - 5 - 10 - 1

3.3 COMPLEJIDAD CICLOMÁTICA

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

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

- $V(G) = A - N + 2$
 $V(G) = 16 - 11 + 2 = 7$

DONDE:

P: Número de nodos predicado

A: Número de aristas

N: Número de nodos

4. REQ 004 Salida de Productos

4.1. CÓDIGO FUENTE

```
void salida_producto() {
    char termino[50];
    int cantidad;
    int encontrado = 0;
    char confirmar;

    printf("Ingrese el código o nombre del producto para disminuir su cantidad: ");
    getchar();
    fgets(termino, 50, stdin);
    termino[strcspn(termino, "\n")] = '\0';

    for (int i = 0; i < prod; i++) {
        if (strstr(producto[i].product, termino) != NULL) {
            printf("\nProducto encontrado: %s (Cantidad actual: %d)\n", producto[i].product, producto[i].cantidad);
            printf("Ingrese la cantidad a retirar: ");
            if (scanf("%d", &cantidad) != 1 || cantidad <= 0) {
                printf("Error: Cantidad invalida.\n");
                while (getchar() != '\n');
                return;
            }

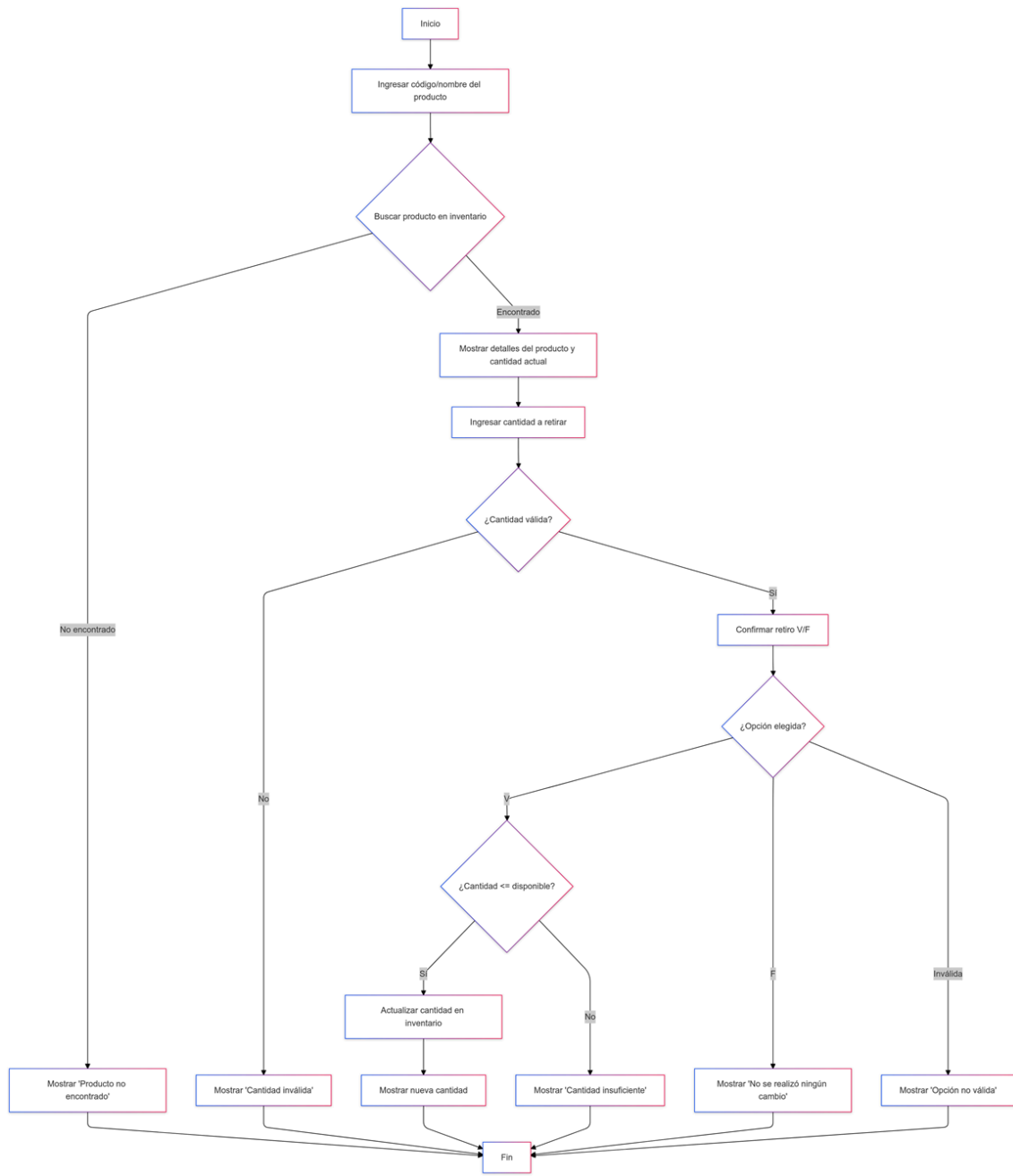
            printf("Esta seguro de que desea retirar %d de '%s' (V/F): ", cantidad, producto[i].product);
            scanf(" %c", &confirmar);
            confirmar = toupper(confirmar);

            if (confirmar == 'V') {
                if (cantidad <= producto[i].cantidad) {
                    producto[i].cantidad -= cantidad;
                    printf("\nCantidad actualizada. Nueva cantidad: %d\n", producto[i].cantidad);
                } else {
                    printf("\nCantidad insuficiente en inventario.\n");
                }
            } else if (confirmar == 'F') {
                printf("\nNo se realizo ningun cambio.\n");
            } else {
                printf("\nOpcion no valida.\n");
            }

            encontrado = 1;
            break;
        }
    }

    if (!encontrado) {
        printf("\nNo se encontro el producto con el código o nombre '%s'.\n", termino);
    }
}
```

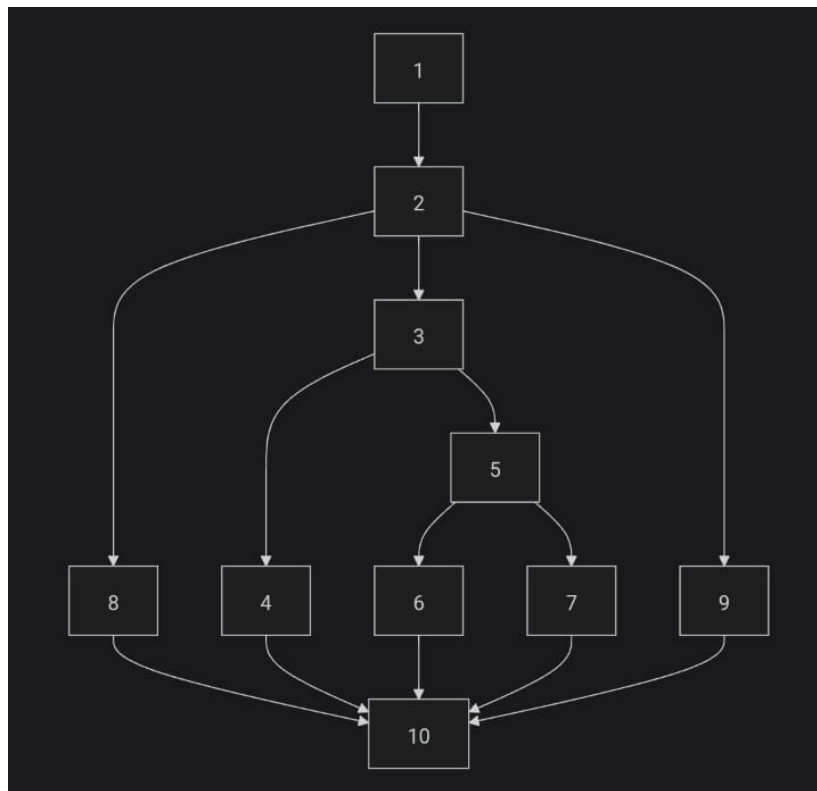
4.2. DIAGRAMA DE FLUJO



flowchart TD

```
A[Inicio] --> B[Ingresar código/nombre del producto]
B --> C{Buscar producto en inventario}
C -->|No encontrado| D[Mostrar 'Producto no encontrado']
C -->|Encontrado| E[Mostrar detalles del producto y cantidad actual]
E --> F[Ingresar cantidad a retirar]
F --> G{¿Cantidad válida?}
G -->|No| H[Mostrar 'Cantidad inválida']
G -->|Sí| I[Confirmar retiro V/F]
I --> J{¿Opción elegida?}
J -->|V| K{¿Cantidad <= disponible?}
J -->|F| L[Mostrar 'No se realizó ningún cambio']
J -->|Inválida| M[Mostrar 'Opción no válida']
K -->|Sí| N[Actualizar cantidad en inventario]
K -->|No| O[Mostrar 'Cantidad insuficiente']
N --> P[Mostrar nueva cantidad]
H --> Q[Fin]
D --> Q
L --> Q
M --> Q
O --> Q
P --> Q
```

4.3. GRAFO



4.3.1. RUTAS

Ruta 1: 1 → 2 → 3 → 4 → 8

Ruta 2: $1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 8$

Ruta 3: $1 \rightarrow 2 \rightarrow 6 \rightarrow 8$

Ruta 4: $1 \rightarrow 2 \rightarrow 7 \rightarrow 8$

4.4. COMPLEJIDAD CICLOMÁTICA

$$V(G) = 10 - 8 + 2(1) \quad V(G) = 4$$

$V(G)$ = número de nodos prediados(decisiones)+1

$$V(G) = 3 + 1 = 4$$

5. REQ 005 - Eliminar Productos

5.1. CÓDIGO FUENTE

```
void eliminar_producto() {
    char termino[50];
    int encontrado = -1;
    char confirmar;

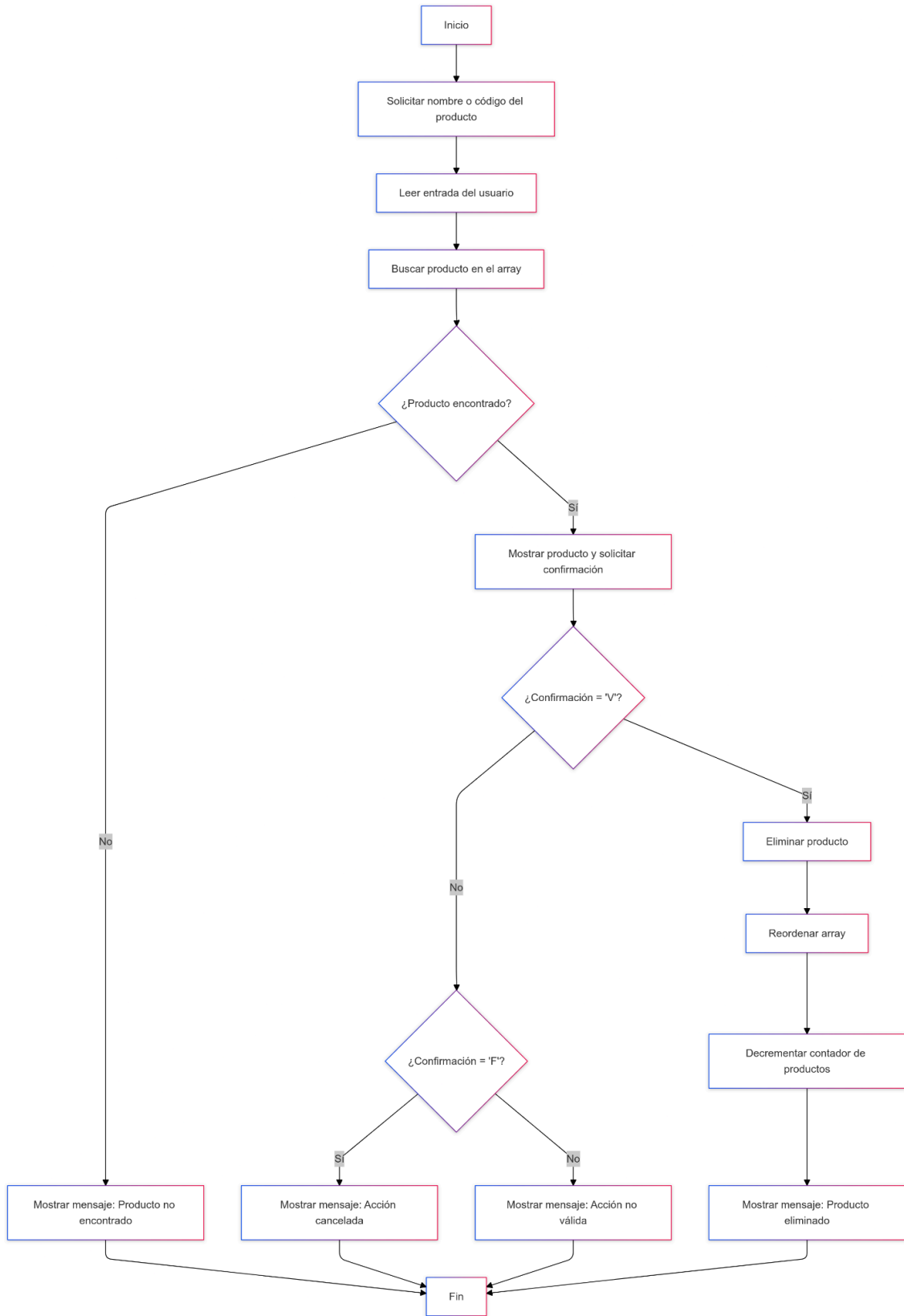
    printf("Ingrese el nombre o código del producto a eliminar: ");
    getchar();
    fgets(termino, 50, stdin);
    termino[strcspn(termino, "\n")] = 0;

    for (int i = 0; i < prod; i++) {
        if (strstr(producto[i].product, termino) != NULL) {
            encontrado = i;
            break;
        }
    }

    if (encontrado != -1) {
        // Preguntar si está segura de eliminar el producto
        printf("Producto encontrado: %s. Esta segura de que desea eliminarlo (V/F): ", producto[encontrado].product);
        scanf(" %c", &confirmar);
        confirmar = toupper(confirmar);

        if (confirmar == 'V') {
            printf("Producto eliminado: %s\n", producto[encontrado].product);
            for (int i = encontrado; i < prod - 1; i++) {
                strcpy(producto[i].product, producto[i + 1].product);
                producto[i].cantidad = producto[i + 1].cantidad;
            }
            prod--;
        } else if (confirmar == 'F') {
            printf("Acción cancelada. No se eliminó ningún producto.\n");
        } else {
            printf("Acción no válida.\n");
        }
    } else {
        printf("No se encontró un producto que coincida con '%s'.\n", termino);
    }
}
```

5.2. DIAGRAMA DE FLUJO



5.3. GRAFO

5.3.1. RUTAS

R1 : $1 \rightarrow 2 \rightarrow 3 \rightarrow 7$ (confirmar == 'V')

R2 : $1 \rightarrow 2 \rightarrow 4 \rightarrow 7$ (confirmar == 'F')

R3: $1 \rightarrow 2 \rightarrow 5 \rightarrow 7$ (confirmar no es válido)

R4 : $1 \rightarrow 2 \rightarrow 6 \rightarrow 7$ (producto no encontrado)

5.4. COMPLEJIDAD CICLOMÁTICA

$V(G)$ = número de nodos predichos(decisiones)+1

$V(G) = 3+1$

$V(G)=4$

$V(G) = A - N + 2$

$V(G) = 9-7+2=4$

DONDE:

P: Número de nodos predichos

A: Número de aristas

N: Número de nodos

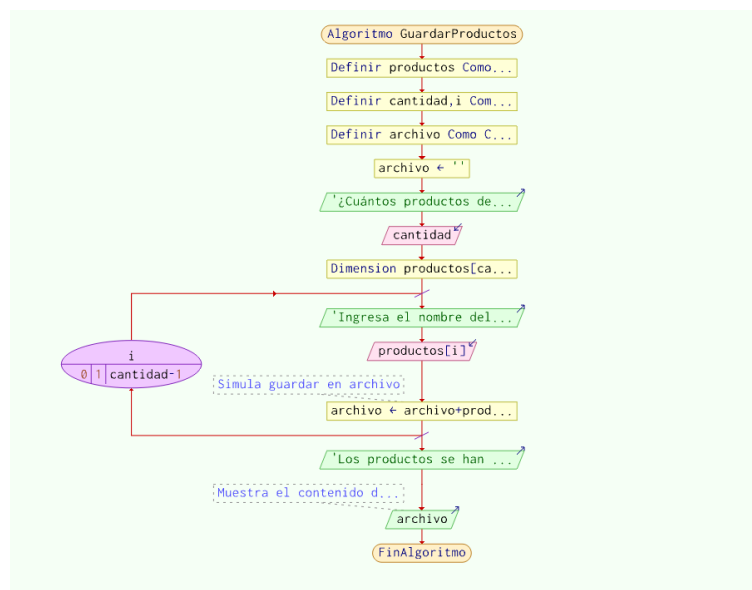
6. REQ 006 Guardar

6.1. CÓDIGO FUENTE

//Funcion para guardar el archivo

```
void guardar() {  
    // Abre el archivo en modo escritura  
    FILE *archivo = fopen("productos.txt", "w");  
    if (archivo == NULL) {  
        printf("Error al abrir el archivo.\n");  
        return;  
    }  
    for (int i = 0; i < prod; i++) {  
        fprintf(archivo, "%d: [Producto: %s]\n", i + 1, producto[i].product);  
    }  
    fclose(archivo);  
  
    printf("Lista de productos guardada exitosamente en 'productos.txt'.\n");  
}
```

6.2. DIAGRAMA DE FLUJO



6.3. GRAFO

6.3.1. RUTAS

RUTAS:3

R1:1 - 2 - 3 - 8

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

R3:1 - 2 - 3 - 4 - 5 - 6 - 7 - 8

6.4. COMPLEJIDAD CICLOMÁTICA

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

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

- $V(G) = A - N + 2$
 $V(G) = 9 - 8 + 2 = 3$

DONDE:

P: Número de nodos predicado

A: Número de aristas

N: Número de nodos