**Prueba de Caja Blanca**

*“Sistema automatizado de control de Inventario”*

**Versión 1.0**

**Integrantes:**

Caizapanta Muela Tammy Amarilis

Guaiguacundo Aguirre Valeria Naomi

Pincha Llanos Estefany Anahí

Robalino Zaldumbide Alejandro Benjamín

**Fecha**

2025-01-21

# REQ 001 - Agregar productos

## CÓDIGO FUENTE

## 

## DIAGRAMA DE FLUJO

## 

## GRAFO

### 

### RUTAS

## 

## COMPLEJIDAD CICLOMÁTICA

# REQ 002 - Buscar productos

## CÓDIGO FUENTE

string credencial;

**1:** printf(“Ingrese la id, el nombre, o categoría del producto, seleccione la credencial para la búsqueda”);

scanf(“%s, &credencial);

//VALIDAR QUE EL FORMATO DE LA ID SEA CORRECTO

int validarID(char \*id) {

**2:** switch{

**3:** id:

**4:**  if (strncmp(id, "lac", 3) != 0 && strncmp(id, "lim", 3) != 0){

**5:** printf("\nProducto(s) encontrado(s):\n");

printf("-------------------------------------------------------\n");

printf("%-10s %-20s %-10s %-10s %-20s\n", "ID", "Nombre", "Categoría", "Existencias", "Proveedor");

printf("-------------------------------------------------------\n");

}

**6:** else

{

printf(“Formato no válido”)

}

**15:** break;

}

//VALIDAR FORMATO PARA EL NOMBRE DEL PRODUCTO

**7:** producto:

int validarNombre(char \*nombre) {

for (int i = 0; i < strlen(nombre); i++) {

**8:** if (!isalpha(nombre[i]) && nombre[i] != ' ') {

**9:** printf(“Imprimir informacion”);

printf("\nProducto(s) encontrado(s):\n");

printf("-------------------------------------------------------\n");

printf("%-10s %-20s %-10s %-10s %-20s\n", "ID", "Nombre", "Categoría", "Existencias", "Proveedor");

printf("-------------------------------------------------------\n");

}

**10:** else

{

printf(“El nombre es inválido”);

}

**15:** break;

}

//VALIDAR FORMATO DEL NOMBRE DEL PROVEEDOR

**11:** categoria:

int validarProveedor(char \*proveedor) {

// VALIDAR QUE EL PROVEEDOR SE ENCUENTRE EN LA LISTA

**12:** if (strcmp(proveedor, "mario sanchez") == 0 || strcmp(proveedor, "emilio chavez") == 0) {

**13:** printf("\nProducto(s) encontrado(s):\n");

printf("-------------------------------------------------------\n");

printf("%-10s %-20s %-10s %-10s %-20s\n", "ID", "Nombre", "Categoría", "Existencias", "Proveedor");

printf("-------------------------------------------------------\n");

}

**14:** else

{

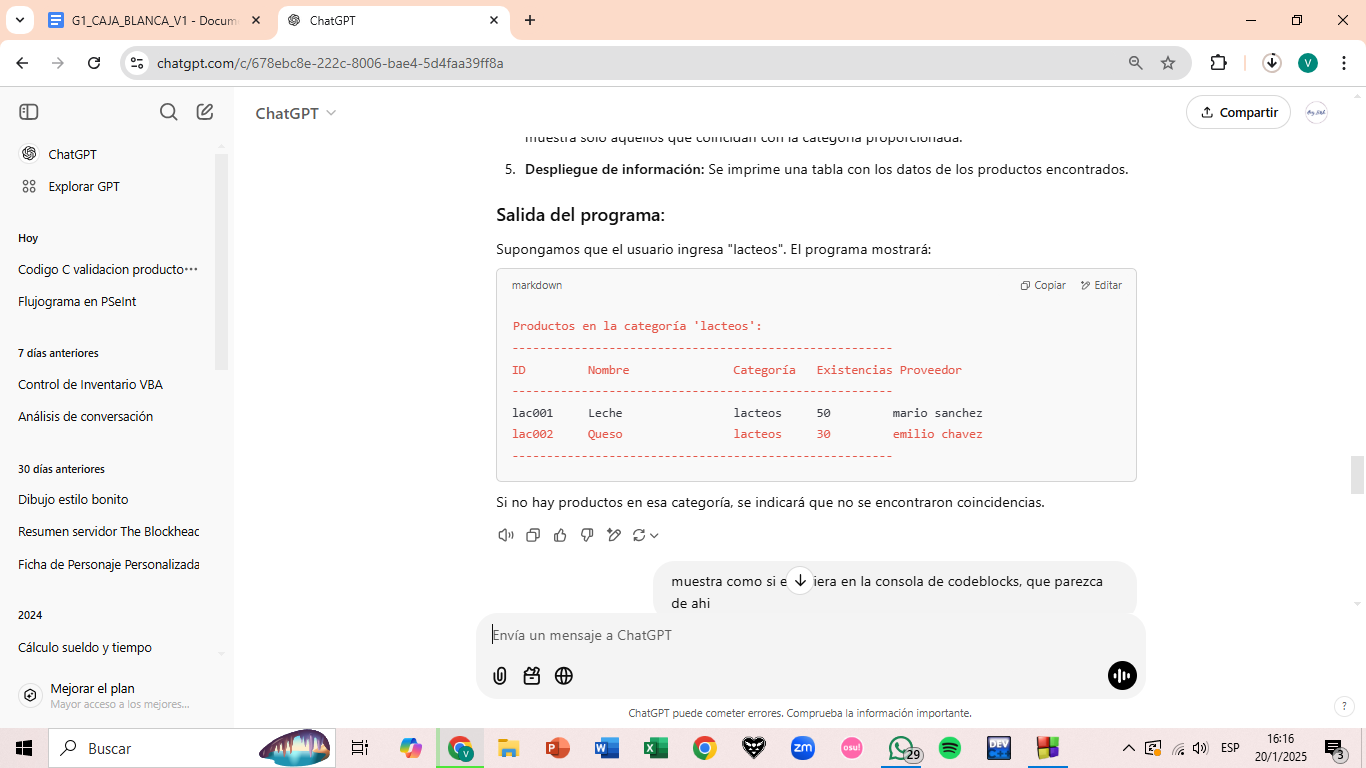
printf(“El formato es inválido”);

}

**15:** break;

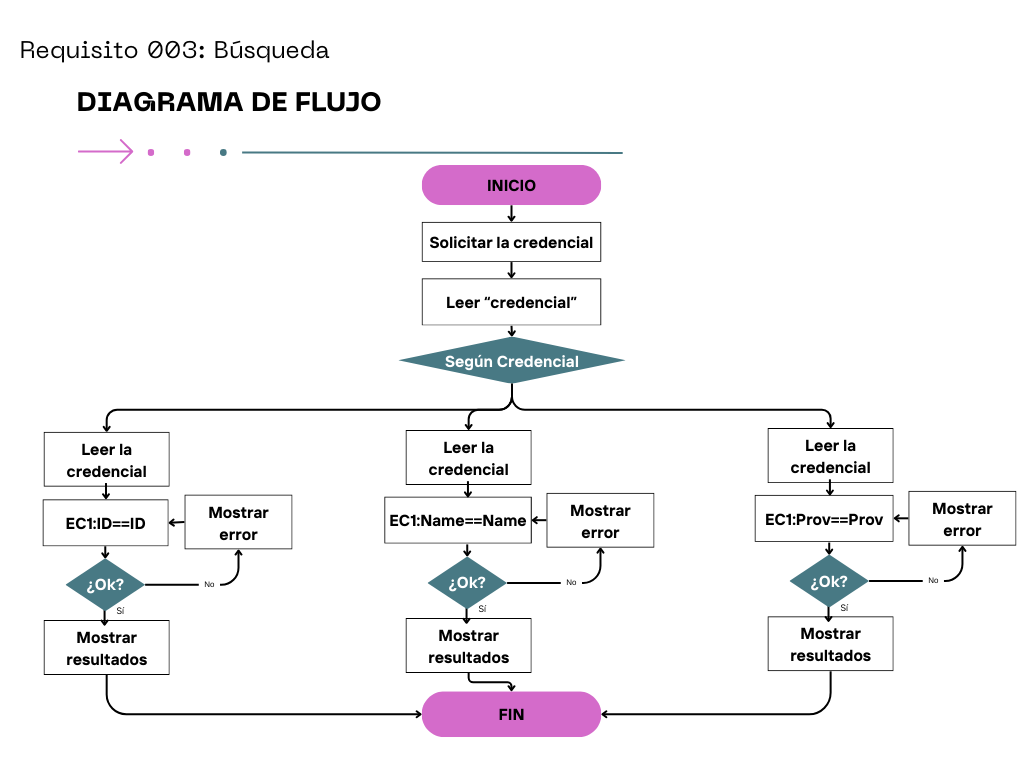
}

return 0;



### 

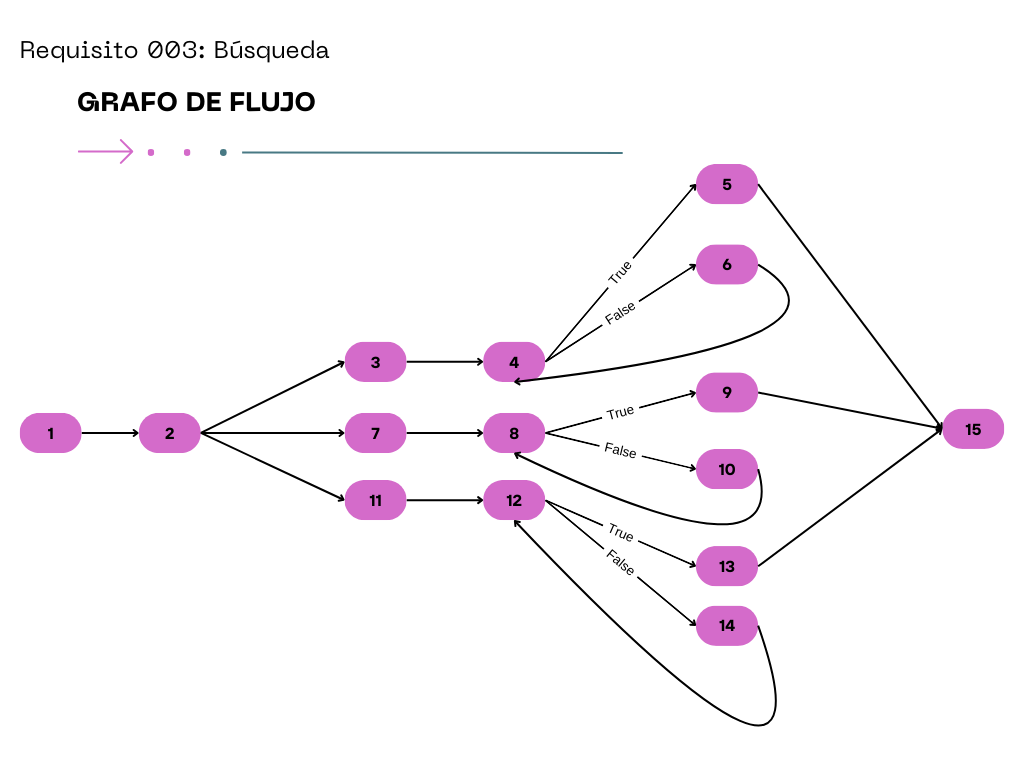
## DIAGRAMA DE FLUJO

****

## 

## 

## GRAFO

****

### RUTAS

**RUTAS:** 6

**R1:** 1–>2-->3–>4–>5–>15

**R2:** 1–>2–>3–>4–>6-->4-->5–>15

**R3:** 1–>2–>7–>8->9–>15

**R4:** 1–>2–>7–>8–>10–>8->9–>15

**R5:** 1–>2–>11-->12->13–>15

**R6:** 1–>2–>11–>12–>14—>12–>13–>15

## 

## 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)= 4 - 2 + 2 = 4

DONDE:

**P:** Número de nodos predicado

**A:** Número de aristas

**N:** Número de nodos

# REQ 003 - Entrada de productos

## CÓDIGO FUENTE

// Cargar productos iniciales

void cargarproductos() {

strcpy(producto[0].product, "001.Pimiento"); producto[0].cantidad = 50;

strcpy(producto[1].product, "002.Cebolla"); producto[1].cantidad = 30;

strcpy(producto[2].product, "003.Ajo"); producto[2].cantidad = 20;

strcpy(producto[3].product, "004.Pepino"); producto[3].cantidad = 40;

strcpy(producto[4].product, "005.Fideos Don Vitorio"); producto[4].cantidad = 100;

strcpy(producto[5].product, "006.lb Arroz(Super Extra)"); producto[5].cantidad = 200;

strcpy(producto[6].product, "007.lb Harina"); producto[6].cantidad = 150;

strcpy(producto[7].product, "008.lb Azucar"); producto[7].cantidad = 90;

// Función para entrada de productos

void entrada\_producto() {

char termino[50];

int cantidad;

int encontrado = 0;

printf("Ingrese el codigo o nombre del producto para aumentar 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("Producto encontrado: %s (Cantidad actual: %d)\n", producto[i].product, producto[i].cantidad);

printf("Ingrese la cantidad a agregar: ");

scanf("%d", &cantidad);

producto[i].cantidad += cantidad;

printf("Cantidad actualizada. Nueva cantidad: %d\n", producto[i].cantidad);

encontrado = 1;

break;

}

}

if (!encontrado) {

printf("No se encontro el producto con el codigo o nombre '%s'.\n", termino);

}

}

## DIAGRAMA DE FLUJO

## 

## GRAFO

### 

### RUTAS

## 

## COMPLEJIDAD CICLOMÁTICA

# 

# 

# REQ 004

## CÓDIGO FUENTE

## 

## DIAGRAMA DE FLUJO

## 

## GRAFO

### 

### RUTAS

## 

## COMPLEJIDAD CICLOMÁTICA

# 

# 

# REQ 005 - Eliminar Productos

## CÓDIGO FUENTE

// Función para eliminar un producto del inventario

void deleteProduct() {

char code[4];

int confirm;

if (productCount == 0) {

printf("El inventario esta vacio.\n");

return;

}

printf("Ingrese el codigo del producto a eliminar: ");

scanf(" %s", code);

int found = 0;

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

if (strcmp(inventory[i].code, code) == 0) {

found = 1;

printf("Esta seguro que desea eliminar este producto? (1: Si, 0: No): ");

scanf("%d", &confirm);

if (confirm == 1) {

for (int j = i; j < productCount - 1; j++) {

inventory[j] = inventory[j + 1];

}

productCount--;

printf("Producto eliminado exitosamente.\n");

} else {

printf("Eliminacion cancelada.\n");

}

break;

}

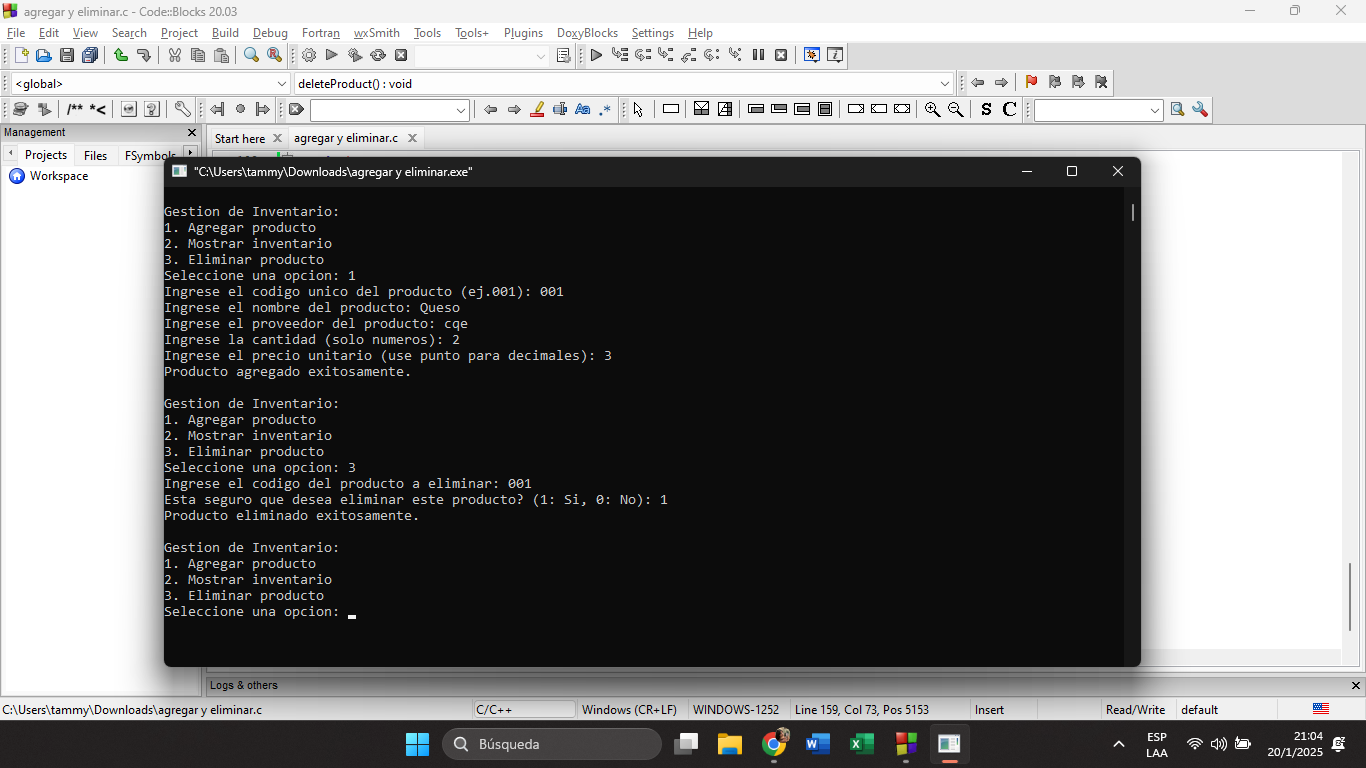
}

if (!found) {

printf("Producto con codigo %s no encontrado.\n", code);

}

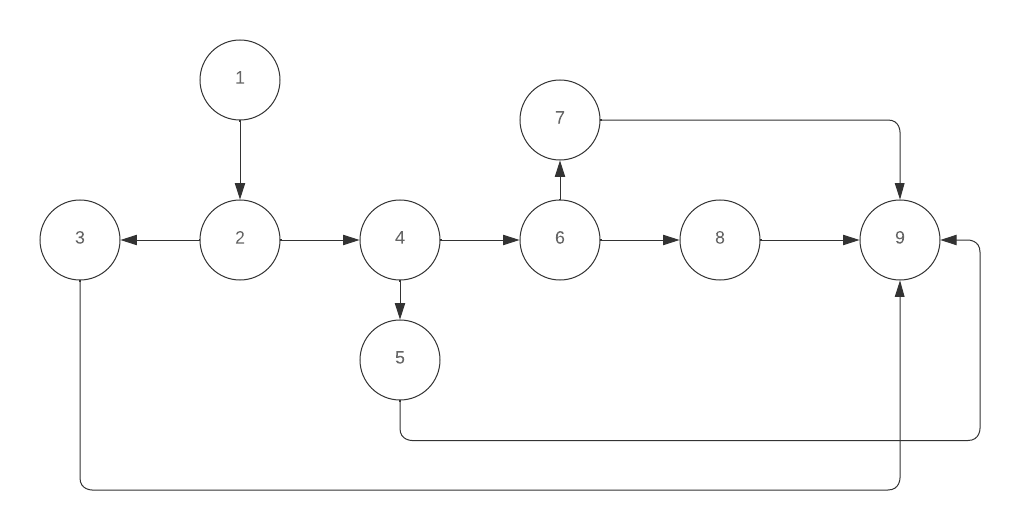
}

****

## DIAGRAMA DE FLUJO

## 

## GRAFO



### RUTAS

RUTAS: 4

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

## 

## COMPLEJIDAD CICLOMÁTICA

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

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

**V(G) = A – N + 2**

V(G) = 10 – 8 + 2 = **4**

DONDE:

**P:** Número de nodos predicado

**A:** Número de aristas

**N:** Número de nodos

# REQ 006

## CÓDIGO FUENTE

## 

## DIAGRAMA DE FLUJO

## 

## GRAFO

### 

### RUTAS

## 

## COMPLEJIDAD CICLOMÁTICA