Taller Colaborativo

SEMANA 15

Nombre de Integrantes:
Andrés Mullo, Cristian Tutin

Introducción

Una tienda de víveres se ha visto afectada debido a la poca efectiva forma de manejo del almacenamiento de los productos. Mediante el uso de agendas se ha mantenido la información de los productos disponibles en el local de forma manual, sin embargo, tanta información escrita en agendas se ha perdido y la confusión ha causado conflictos al momento de registrar los ingresos y la cantidad de productos que se contaban al principio. El uso de tácticas antiguas como las agendas no han dado los frutos esperados para los encargados de este negocio. Por lo tanto, se busca encontrar una manera moderna y eficiente para poder crear un sistema de inventario que guarde la información importante registrada por los encargados del negocio, de forma que no haya más problemas y el negocio pueda funcionar de una mejor manera.

Objetivos

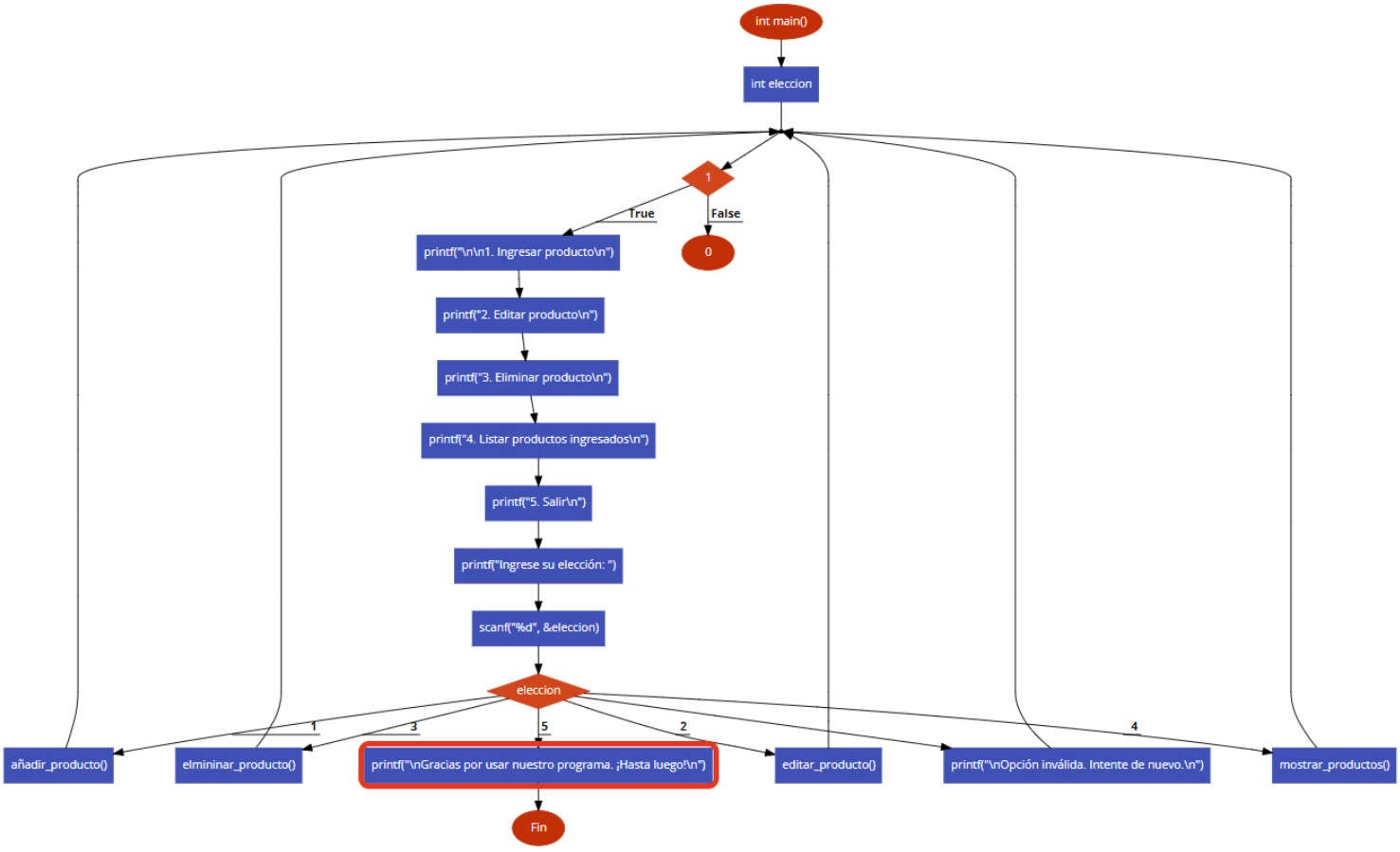
- Usar los conocimientos adquiridos para solucionar el problema y crear un sistema de inventario eficiente que facilite el almacenamiento de los productos en archivos planos.
- Exponer los resultados del programa propuesto y dar una clara explicación sobre el funcionamiento del programa.
- Analizar los métodos necesarios para que el programa pueda funcionar de mejor manera.

Nombre del sistema	*Tienda minorista de víveres*
Usuarios	Encargados de la tienda
Objetivo	Almacenar información esencial (nombre, cantidad y precio) de los productos disponibles en la tienda

Contexto del problema (Considerar los contextos económico, social y ambiental, dentro del sector productivo en el que funcionaría el sistema de inventarios)	La administración del inventario es un tema central para evitar problemas financieros en las organizaciones, es un componente fundamental en la productividad de una empresa, ya que es el activo corriente de menor liquidez que manejan y que además contribuye a generar rentabilidad. Es el motor que mueve a la organización, pues es la base para la comercialización de la empresa que le permite obtener ganancias. (Durán, Y. 2012. Pg.3)
Restricciones (Consideraciones externas que se deben tener para diseñar y desarrollar el proyecto, ejemplo: porcentaje de impuesto, registro sanitario u otros)	Existen consideraciones externas mayormente enfocadas en como realizar correctamente un sistema de inventario que pueda beneficiar a la tienda en cuestión, ya que como todo negocio se necesita ser cauteloso al momento de administrar correctamente sus productos. se requiere del uso de diferentes técnicas de inventario, a fin de determinar su nivel óptimo y así disminuir los costos totales implicados en el inventario y optimizar las utilidades.
Limitaciones (Consideraciones internas que se deben tener para diseñar y desarrollar el proyecto, ejemplo: lenguaje de programación, tipos de almacenamiento u otros)	Las consideraciones internas se relacionan al proceso del desarrollo del programa ya que, siendo un desarrollo, un tanto extenso se debe tener cuidado al momento al momento de usar las sentencias y las funciones, ya que un error puede dificultar la ejecución del programa. Además, que se debió adaptar el problema a las herramientas disponibles en el entorno de programación.

Diagramas de flujo

Para poder tener una idea de cómo realizar el algoritmo en el lenguaje C se desarrolló un simple diagrama de flujo para reflejar lo que vera el usuario, ósea escoger entre 5 opciones para que se de paso a la ejecución de las instrucciones que están contenidas en una función.



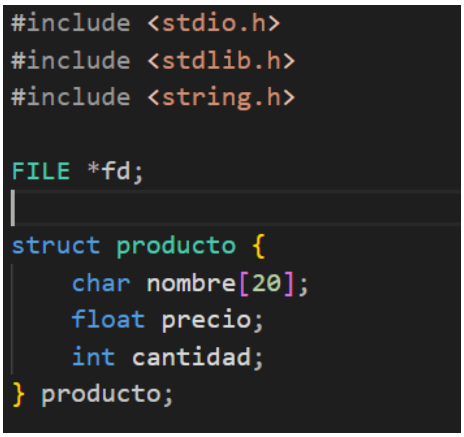
GitHub

https://github.com/AndresMV10/Sistema-de-Inventario

Explicación del algoritmo implementado en C

sistema.h

1. Este parte del código incluye las librerías necesarias para el desarrollo de las funciones, además declaramos un archivo global y una estructura de datos que nos servida para almacenar los datos necesarios para el programa.



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

FILE *fd;

struct producto {
    char nombre[20];
    float precio;
    int cantidad;
} producto;
```

2. La función agregarProducto() facilita la incorporación de nuevos productos al inventario. Primero, se abre el archivo "productos.txt" en modo de adición de texto ("at"), permitiendo la escritura al final del archivo. Si hay algún problema al abrir el archivo, se muestra un mensaje de error y se sale de la función. A continuación, se verifica si el archivo está vacío. En caso afirmativo, se inserta la palabra "Inventario" en el medio del archivo, seguido de las etiquetas de columna para los datos de los productos. Esta acción asegura un formato adecuado para la presentación de los productos en el archivo. Posteriormente, se presenta un mensaje indicando que se está agregando un producto y se inicia un bucle que permite al usuario ingresar información sobre el producto, como el nombre, precio y cantidad. Los datos proporcionados se escriben en el archivo en un formato estructurado, y se pregunta al

usuario si desea agregar más productos. El bucle continuará mientras la respuesta sea 's' (sí). Finalmente, se cierra el archivo después de completar la adición de productos al inventario. Este procedimiento proporciona una interfaz sencilla y eficiente para mantener y actualizar el inventario de productos de manera dinámica.

```c
void agregarProducto() {
    char rpt;
    fd = fopen("productos.txt", "at"); // add text - añadir un texto
    if (fd == NULL) {
        printf("Error al tratar de abrir el archivo");
        return;
    }

    // Verificar si el archivo está vacío
    fseek(fd, 0, SEEK_END);
    long fileSize = ftell(fd);
    if (fileSize == 0) {
        // Insertar "Inventario" en medio del archivo
        fseek(fd, 0, SEEK_END);  // Ir al final del archivo
        long halfSize = fileSize / 2;  // Calcular la mitad del tamaño
        // Calcular la mitad del tamaño y restar un poco para asegurar que esté al principi
        halfSize = (fileSize / 2) - 10;

        fseek(fd, halfSize, SEEK_SET);  // Ir a la mitad del archivo
        fprintf(fd, "Inventario\n");

        // Volver al final del archivo para agregar productos
        fseek(fd, 0, SEEK_END);

        // Agregar etiquetas de columna al principio de cada línea
        fprintf(fd, "%-20s%-15s%-10s\n", "Nombre", "Precio", "Cantidad");
        fprintf(fd, "-----------------------------------------\n");
    }
    printf("\n\t.:Agregando producto:.\n");
    do {
        fflush(stdin);
        printf("\nNombre : ");
        scanf("%s", producto.nombre);
        printf("Precio : ");
        scanf("%f", &producto.precio);
        printf("Cantidad : ");
        scanf("%d", &producto.cantidad);
        fprintf(fd, "%-20s%-15.2f%-10d\n", producto.nombre, producto.precio, producto.canti
        printf("\nDesea agregar más productos (s) : ");
        scanf(" %c", &rpt);
    } while (rpt == 's');
    fclose(fd);
}
```

3. La función visualizarProductos() se encarga de presentar de manera organizada y legible la información almacenada en el

archivo "productos.txt". Primero, se abre el archivo en modo de lectura de texto ("r"), y se realiza una verificación para asegurarse de que el archivo se abrió correctamente. En caso de error, se muestra un mensaje y se sale de la función. A continuación, se leen y omiten las líneas de encabezado que contienen información como "Inventario" y las etiquetas de columna. Después, se imprime un encabezado en la consola con los nombres de las columnas ("Nombre", "Precio" y "Cantidad") y una línea de separación para una presentación ordenada. Luego, mediante un bucle, se lee cada línea del archivo, y se extraen los datos del producto, como el nombre, precio y cantidad. Estos datos se imprimen en la consola en un formato tabular, asegurando una presentación organizada y fácil de entender.

```c
void visualizarProductos() {
    int c;
    fd = fopen("productos.txt", "r"); // read text - leer texto

    if (fd == NULL) {
        printf("Error al tratar de abrir el archivo");
        return;
    }

    // Leer y omitir las líneas de encabezado
    char buffer[100];
    fgets(buffer, sizeof(buffer), fd); // Leer "Inventario" y avanzar a la siguiente línea
    fgets(buffer, sizeof(buffer), fd); // Leer la línea de etiquetas de columna
    fgets(buffer, sizeof(buffer), fd); // Leer la línea de separación

    printf("\n%-20s%-15s%-10s\n", "Nombre", "Precio", "Cantidad");
    printf("-----------------------------------------------\n");

    while (fscanf(fd, "%s%f%d", producto.nombre, &producto.precio, &producto.cantidad) !=
        printf("%-20s%-15.2f%-10d\n", producto.nombre, producto.precio, producto.cantidad)
    }

    fclose(fd);
}
```

4. La función editarProducto() permite al usuario modificar la información de un producto específico en el inventario

almacenado en el archivo "productos.txt". Inicialmente, se abre el archivo en modo de lectura y escritura ("r+"), y se realiza una verificación para asegurarse de que el archivo se abrió correctamente. En caso de error, se muestra un mensaje y se sale de la función. Luego, se leen y omiten las líneas de encabezado del archivo para evitar que se sobrescriban durante la edición. A continuación, se solicita al usuario que ingrese el nombre del producto que desea editar. Se crea un archivo temporal ("temp.txt") en modo de escritura ("w") para almacenar la información actualizada del inventario. Se escribe la palabra "Inventario" y el encabezado en el archivo temporal para mantener la estructura del archivo original. Luego, se inicia un bucle que recorre cada línea del archivo original. Si el nombre del producto coincide con el nombre proporcionado por el usuario, se le pide al usuario que ingrese el nuevo precio y la nueva cantidad para ese producto. Los datos actualizados se escriben en el archivo temporal. Si no se encuentra coincidencia, se copia la línea original sin cambios al archivo temporal. Finalmente, se cierran ambos archivos, se elimina el archivo original "productos.txt" y se renombra el archivo temporal como "productos.txt". Se imprime un mensaje indicando que el producto ha sido editado exitosamente. Esta función proporciona una forma interactiva y dinámica de modificar la información de un producto específico en el inventario.

```c
void editarProducto() {
    char nombreEditar[20];
    fd = fopen("productos.txt", "r+"); // Cambiar a "r+" para lectura y escritura

    if (fd == NULL) {
        printf("Error al tratar de abrir el archivo");
        return;
    }
    // Leer y omitir las líneas de encabezado
    char buffer[100];
    fgets(buffer, sizeof(buffer), fd); // Leer "Inventario" y avanzar a la siguiente línea
    fgets(buffer, sizeof(buffer), fd); // Leer la línea de etiquetas de columna
    fgets(buffer, sizeof(buffer), fd); // Leer la línea de separación
    printf("\n\t.:Editar producto:.\n");
    printf("Ingrese el nombre del producto a editar: ");
    scanf("%s", nombreEditar);
    FILE *tempFd = fopen("temp.txt", "w");

    // Escribir "Inventario" y el encabezado en el archivo temporal
    fprintf(tempFd, "Inventario\n");
    fprintf(tempFd, "%-20s%-15s%-10s\n", "Nombre", "Precio", "Cantidad");
    fprintf(tempFd, "-----------------------------------------\n");

    while (fscanf(fd, "%s%f%d", producto.nombre, &producto.precio, &producto.cantidad) != E
        if (strcmp(producto.nombre, nombreEditar) == 0) {
            printf("Nuevo precio para %s: ", producto.nombre);
            scanf("%f", &producto.precio);
            printf("Nueva cantidad para %s: ", producto.nombre);
            scanf("%d", &producto.cantidad);
        }
        fprintf(tempFd, "%-20s%-15.2f%-10d\n", producto.nombre, producto.precio, producto.c
    }

    fclose(fd);
    fclose(tempFd);

    remove("productos.txt");
    rename("temp.txt", "productos.txt");

    printf("Producto editado exitosamente.\n");
}
```

5. La función eliminarProducto() permite al usuario eliminar un producto específico del inventario almacenado en el archivo "productos.txt". Al igual que en las funciones anteriores, se inicia abriendo el archivo en modo de lectura ("r"). Se realiza una verificación para asegurarse de que el archivo se abrió correctamente. En caso de error, se muestra un mensaje y se sale de la función. A continuación, se leen y omiten las líneas de encabezado del archivo para evitar que se sobrescriban durante la

eliminación. Se solicita al usuario que ingrese el nombre del producto que desea eliminar. Se crea un archivo temporal ("temp.txt") en modo de escritura ("w") para almacenar la información actualizada del inventario. Se escribe la palabra "Inventario" y el encabezado en el archivo temporal para mantener la estructura del archivo original. Luego, se inicia un bucle que recorre cada línea del archivo original. Si el nombre del producto coincide con el nombre proporcionado por el usuario, se omite la escritura de esa línea, indicando que el producto ha sido eliminado. Si no se encuentra coincidencia, se copia la línea original sin cambios al archivo temporal. Finalmente, se cierran ambos archivos, se elimina el archivo original "productos.txt" y se renombra el archivo temporal como "productos.txt". Se imprime un mensaje indicando si el producto ha sido eliminado exitosamente o si no se encontró el producto con ese nombre.

```c
void eliminarProducto() {
    char nombreEliminar[20];
    fd = fopen("productos.txt", "r");
    if (fd == NULL) {
        printf("Error al tratar de abrir el archivo");
        return;
    }
    // Leer y omitir las líneas de encabezado
    char buffer[100];
    fgets(buffer, sizeof(buffer), fd); // Leer "Inventario" y avanzar a la siguiente línea
    fgets(buffer, sizeof(buffer), fd); // Leer la línea de etiquetas de columna
    fgets(buffer, sizeof(buffer), fd); // Leer la línea de separación

    printf("\n\t.:Eliminar producto:.\n");
    printf("Ingrese el nombre del producto a eliminar: ");
    scanf("%s", nombreEliminar);

    FILE *tempFd = fopen("temp.txt", "w");
    int encontrado = 0;
    // Escribir "Inventario" y el encabezado en el archivo temporal
    fprintf(tempFd, "Inventario\n");
    fprintf(tempFd, "%-20s%-15s%-10s\n", "Nombre", "Precio", "Cantidad");
    fprintf(tempFd, "-----------------------------------------\n");

    while (fscanf(fd, "%s%f%d", producto.nombre, &producto.precio, &producto.cantidad) != E
        if (strcmp(producto.nombre, nombreEliminar) != 0) {
            fprintf(tempFd, "%-20s%-15.2f%-10d\n", producto.nombre, producto.precio, produc
        } else {
            encontrado = 1;
        }
    }

    fclose(fd);
    fclose(tempFd);
    remove("productos.txt");
    rename("temp.txt", "productos.txt");
    if (encontrado) {
        printf("Producto eliminado exitosamente.\n");
    } else {
        printf("No se encontró el producto con ese nombre.\n");
    }
}
```

Proyecto_3.c

1. Agregamos el archivo “sistema.h”, la cual es una librería que contiene las funciones antes descritas.

2. Creamos un menú simple en el cual el usuario puede eligir entre las cinco opciones existentes mediante el uso de los números en el teclado,

además controlamos que no se ingrese caracteres, puesto que este generaría un bucle infinito.

3. Este programa en C implementa un menú interactivo para gestionar un inventario de productos. El usuario puede realizar las siguientes operaciones:

Agregar Producto: Permite al usuario ingresar información sobre un nuevo producto, como nombre, precio y cantidad, y lo agrega al inventario.

Visualizar Productos: Muestra en la consola la lista de productos almacenados en el inventario, incluyendo detalles como nombre, precio y cantidad.

Editar Producto: Permite al usuario modificar la información de un producto existente, como el precio y la cantidad, ingresando el nombre del producto a editar.

Eliminar Producto: Permite al usuario eliminar un producto existente ingresando el nombre del producto a eliminar.

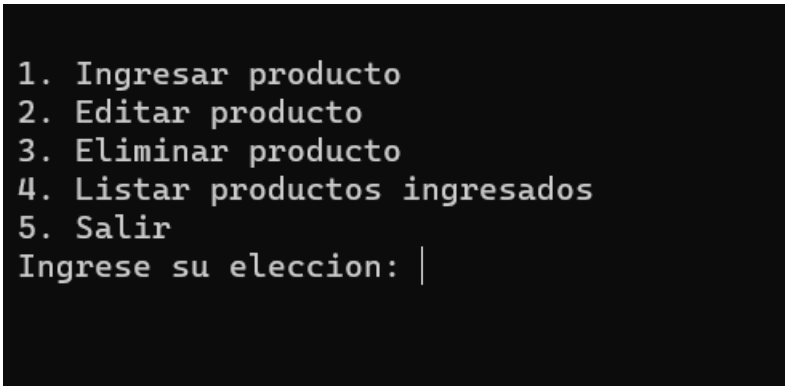
Salir: Finaliza el programa.

El bucle principal (do-while) del programa permite al usuario seleccionar una opción del menú, ejecutar la operación correspondiente y repetir el proceso hasta que elija salir (opc == 5). Además, se ha agregado una validación para garantizar que el usuario ingrese una opción numérica válida, evitando bucles infinitos causados por entradas no válidas.

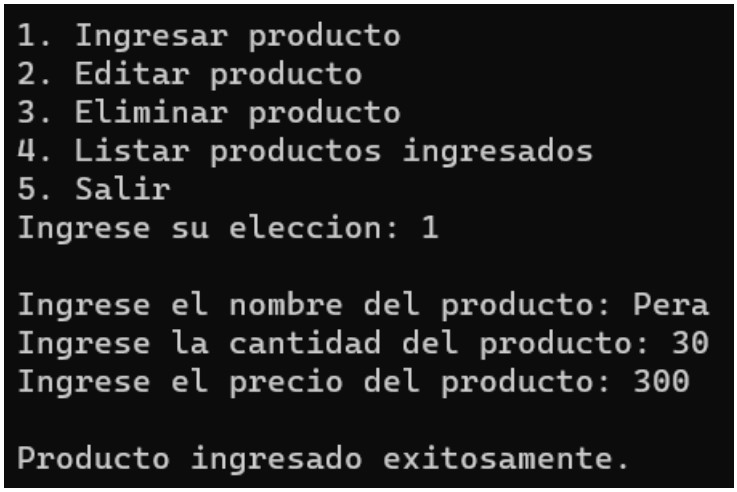
```c
int main() {
    int opc;

    do {
        printf("\n\t.:MENU:.\n");
        printf("1. Agregar Producto\n");
        printf("2. Visualizar Productos\n");
        printf("3. Editar Producto\n");
        printf("4. Eliminar Producto\n");
        printf("5. Salir\n");

        printf("Opcion : ");
        if (scanf("%i", &opc) != 1) {
            // Limpiar el búfer de entrada en caso de entrada no válida
            scanf("%*s");
            printf("Por favor, ingrese un número válido.\n");
            continue;  // Vuelve al principio del bucle
        }

        switch (opc) {
            case 1:
                agregarProducto();
                break;
            case 2:
                visualizarProductos();
                break;
            case 3:
                editarProducto();
                break;
            case 4:
                eliminarProducto();
                break;
            case 5:
                printf("Saliendo...\n");
                break;
            default:
                printf("Opción no válida. Por favor, elija una opción del 1 al 5.\n");
        }

    } while (opc != 5);

    return 0;
}
```

Explicación de la ejecución de cada sección del programa

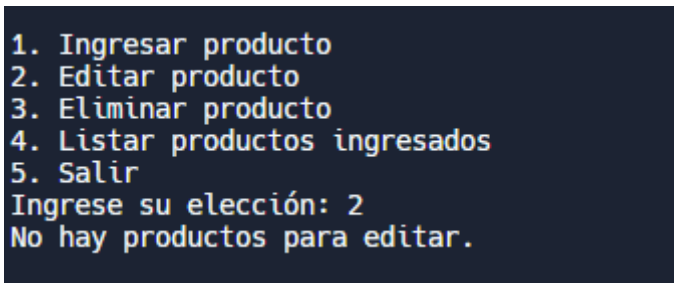
1. Al ejecutar el programa, nos aparece esta pantalla con un menú simple, en el cual hay 5 opciones a elegir por el usuario, y claro su mensaje pidiendo su elección.



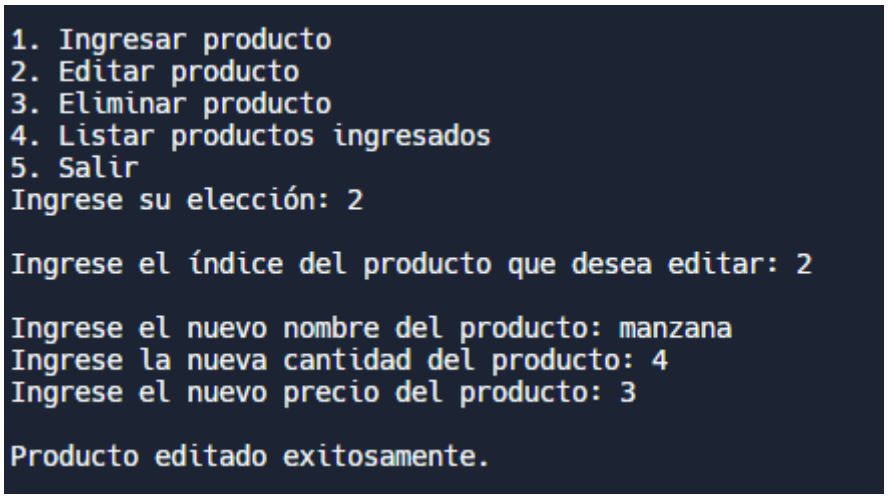
2. Si elegimos la primera opción, el programa nos pedirá el nombre, cantidad y precio del producto, consecuentemente nos aparecerá un mensaje de “producto ingresado “exitosamente” y se nos devolverá al menú.



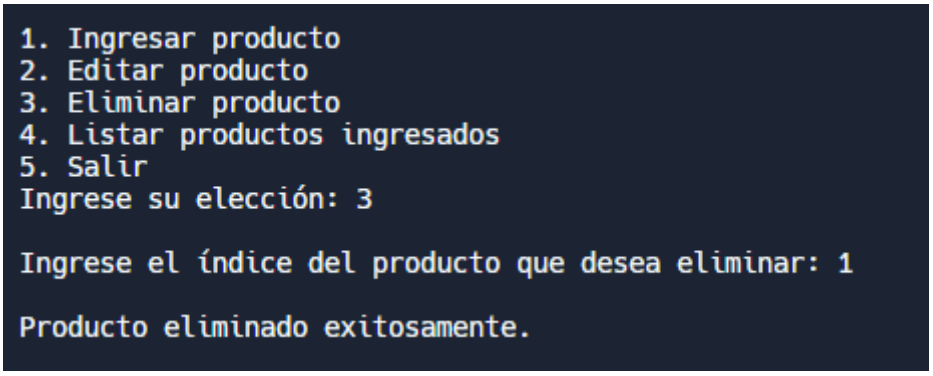
3. Si elegimos la segunda opción, nos pedirá ingresar el índice del producto que se quiere editar. Si no hay un producto ingresado entonces aparecerá el siguiente mensaje:



Por el contrario, si se ha ingresado cierta cantidad de productos al momento de editar la información podremos escoger entre los productos que se ingresaron y cambiar tanto el nombre, el precio y la cantidad para enlistarlo.



4,- Si se elige la tercera opción, nos pedirá ingresar el índice del producto que se desea eliminar de la lista:



Si no hay productos que eliminar el programa mostrará el siguiente mensaje:

```
1. Ingresar producto
2. Editar producto
3. Eliminar producto
4. Listar productos ingresados
5. Salir
Ingrese su elección: 3
No hay productos para eliminar.
```

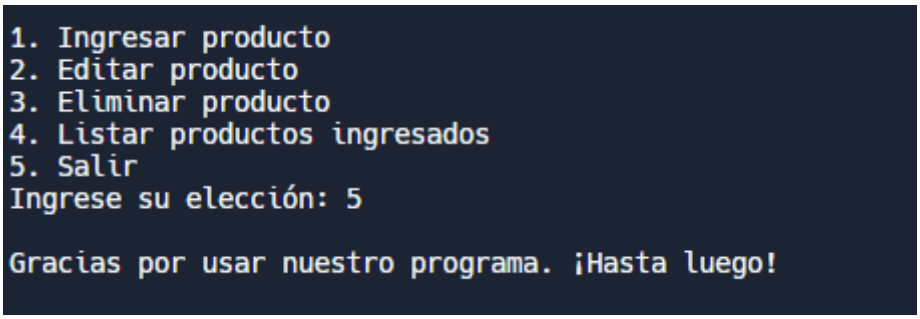
5.- Al elegir la cuarta opción, se desplegará la lista de los productos ingresados hasta el momento, junto con toda la información que se proporcionó.

```
1. Ingresar producto
2. Editar producto
3. Eliminar producto
4. Listar productos ingresados
5. Salir
Ingrese su elección:
4

Productos ingresados:
Índice  Nombre  Cantidad    Precio
0   carro   1   25000.00
1   banana  45  4.00
```

Si no hay productos para mostrar el programa desplegará lo siguiente:

```
1. Ingresar producto
2. Editar producto
3. Eliminar producto
4. Listar productos ingresados
5. Salir
Ingrese su elección: 4
No hay productos ingresados.
```

6.- Al escoger la quinta opción, el programa se dará por finalizado y nos agradecerá por su uso.



Recomendaciones

- Colocar las librerías principales, datos de salida y parte del proceso en un header ya que de esa manera se ahorrará espacio en las líneas de código del programa en C, para poder evitar errores y tener una mejor organización.
- Usar una sentencia switch para poder crear el menú, de las sentencias de repetición es el que tiene una estructura que puede facilitar el proceso.
- Las estructuras de repetición if son muy útiles para evitar que se ingresen datos erróneos al momento de pedir información del producto. Por ejemplo, que se ingresen números cuando se pide el nombre.

Conclusiones

En conclusión, el programa en C presentado proporciona una interfaz de consola para la gestión de inventarios de productos, utilizando tanto un array en memoria como un archivo para el almacenamiento persistente de datos. La aplicación permite llevar a cabo operaciones como agregar,

visualizar, editar y eliminar productos, ofreciendo así una herramienta básica para el control de inventarios.

La implementación de este sistema podría resultar útil en diversos contextos, desde pequeños negocios hasta entornos más amplios que requieran un seguimiento y organización eficientes de productos. A pesar de que el alcance del programa es relativamente simple, su estructura modular y la interacción con archivos proporcionan una base que puede ser ampliada y adaptada según las necesidades específicas del usuario.

En el ámbito de la programación, el código demuestra conceptos clave como el manejo de archivos, el uso de estructuras de datos (structs), bucles, funciones y control de flujo. Aunque se ha abordado y corregido un problema específico en la implementación original, el código sigue siendo una base sólida que puede mejorarse y expandirse para satisfacer requisitos adicionales o escenarios más complejos.

En resumen, este programa en C ofrece una herramienta funcional para la gestión de inventarios, y su flexibilidad permite su adaptación a diversas necesidades y entornos. Se posiciona como un punto de partida valioso para proyectos más extensos y, a pesar de sus limitaciones actuales, constituye una base sólida sobre la cual construir soluciones más robustas.