

Taller Colaborativo

ABET

Nombre de Integrantes: Andrés Mullo, Cristian Tutin

9

**Introducción**

La Fundación Salud para Todos, es una organización no gubernamental sin fines de lucro, la cual, a través de donaciones de empresas privadas y población civil, financia hasta un 90% de los tratamientos de pacientes con enfermedades raras en el Ecuador considerando su criticidad y situación socio económica, sin embargo, debido a la pandemia producida por el COVID 19, los ingresos de la fundación se han visto reducidos en gran medida, teniendo que dejar de financiar parcialmente o en su totalidad varios de los tratamientos que hasta antes de la pandemia se encontraban cubiertos.

En tal motivo y con el fin de volver a brindar los tratamientos completos a los pacientes, a la vez de optimizar el recurso económico, la Fundación con ayuda de entes internacionales implementó una unidad médica gestionada y administrada por la misma Fundación.

Sin embargo, una vez puesta en marcha dicha unidad médica, la Fundación se encontró con el siguiente problema:

El agendamiento de citas el cual es realizado manualmente por el personal de la fundación es deficiente, produciéndose una perdida de información y confusión en las citas agendadas.

Por lo tanto, se busca crear un sistema de agendamiento de citas que evite la pérdida de esta información tan importante. Mediante la programación se puede conseguir una manera más eficiente de almacenar las citas para poder ayudar al personal y evitar errores graves.

El objetivo de este trabajo es exponer los resultados del programa.

**Justificación del trabajo**

Mediante el uso de archivos, funciones, estructuras de datos en el lenguaje C se puede solucionar este problema al crear un programa que permita hacer más fácil el control de esta información que la fundación necesita para ejercer correctamente su trabajo.

Que una persona haga el agendamiento de citas manualmente puede ser demasiado contraproducente si esta persona pierde la información. Mientras que un sistema que está programado tiene mejor capacidad de almacenamiento.

**Objetivos**

* Crear un sistema que permita almacenar el agendamiento de citas para solucionar el problema de la pérdida de la información.
* Optimizar el código para un correcto uso y asegurarse que su uso sea fácil de manejar para el personal de la fundación.
* Analizar los métodos necesarios para lograr crear un sistema eficiente y comprensible.

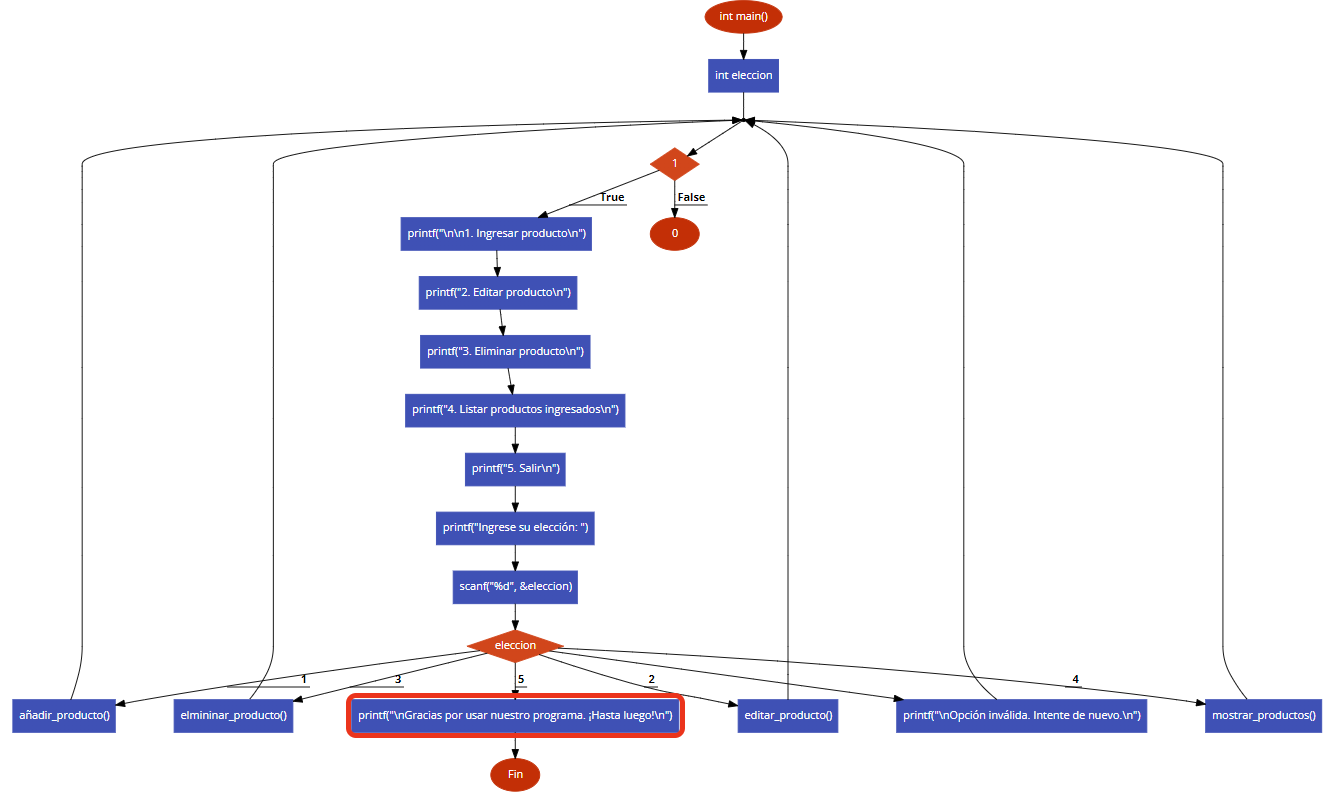
**Análisis del problema elegido**

La fundación se ha visto afectada en ciertos ámbitos por la pandemia del COVID 19. Probablemente, el hecho de que una persona realice el agendamiento de citas fue un camino que tuvo que tomarse para darle prioridad a los temas financieros, sin embargo, que una persona maneje toda esta información por su cuenta, es deficiente y al final solo causara frustración y confusión con la pérdida de la información.

|  |  |
| --- | --- |
| Variables de entrada | Descripción |
| 1. Nombre del producto:   Char nombres[PRODUCTOS\_MAXIMOS][50];   1. Precio del producto:   float precios[PRODUCTOS\_MAXIMOS];   1. Cantidad del producto   int cantidades[PRODUCTOS\_MAXIMOS];   1. Número de productos   int numero\_productos = 0; | 1. El nombre del producto es una variable de tipo char y es un arreglo bidimensional que almacena los nombres de la cantidad máxima de productos y que puede almacenar una cadena de hasta 49 caracteres, sin contar el espacio adicional para el carácter nulo. 2. El precio es una variable de tipo flotante que almacena el precio de los productos máximos que serían 100. 3. La cantidad es una variable de tipo entero que contiene la cantidad de productos máximos almacenados. 4. El número es una variable de tipo entero que pretende funcionar como un contador inicializado en 0. |
| Procesos | Descripción |
| 1. Uso de headers 2. Funciones:  * void añadir\_producto() * void editar\_producto() * void elmininar\_producto()  1. Función añadir producto 2. Función editar producto 3. Función eliminar producto 4. Función mostrar producto 5. Sentencia switch | 1. Los datos de entrada se colocaron en un archivo.h para poder crear una biblioteca que facilite el proceso principal que es permitir al usuario enlistar, añadir o quitar productos. Se definió una variable global que hace que los productos máximos sean 100. 2. En estas funciones se encuentra lo que el usuario desea realizar con el programa todas inician con una variable de tipo void ya que no se espera que se devuelva algún valor. 3. En esta función el usuario puede añadir productos hasta que llegue al limite de productos máximos. 4. Mediante la función strcpy que permite copiar una cadena de caracteres almacenados en una fuente hacia un destino, para así almacenar la nueva información del nombre, precio, cantidad de un producto en la matriz. 5. En la función eliminar producto, se pide al usuario el índice del producto que desea eliminar. Se usa una función strcpy y una sentencia for para eliminar un producto en cuestión y almacenar el restante. 6. La función permite mediante una sentencia for mostrar los productos enlistados. 7. En esta sentencia el usuario escoge la opción entre ingresar un producto, editar un producto, eliminar un producto, enlistar los productos ingresados o dar por terminado el programado dependiendo de cuál sea la elección del usuario. |
| Variables de Salida | Descripción |
| * Nombre del producto * Precio * Cantidad | El objetivo del programa es enlistar la información proporcionada por el usuario sobre los productos de la tienda, así que, como resultado el sistema de inventario debe mostrar los datos que proporciono el usuario. |

**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.



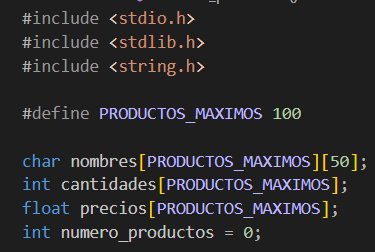
**GitHub**

<https://github.com/AndresMV10/TALLER-COLABORATIVO-ABET---JOSUE-MULLO---CRISTIAN-TUTIN>

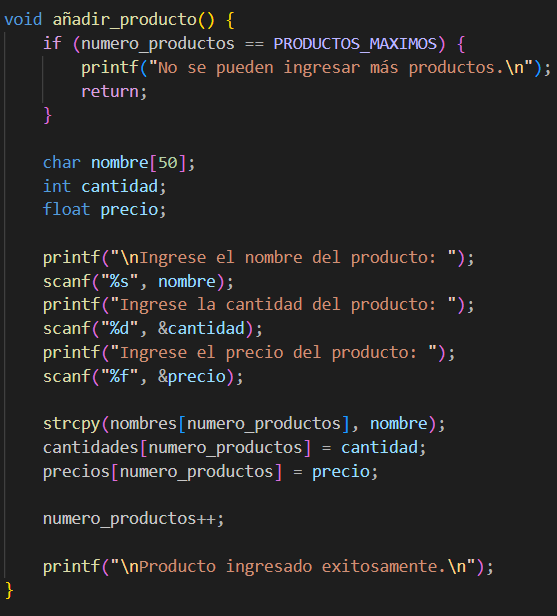
**Explicación del algoritmo implementado en C**

**Funcionalidades.h**

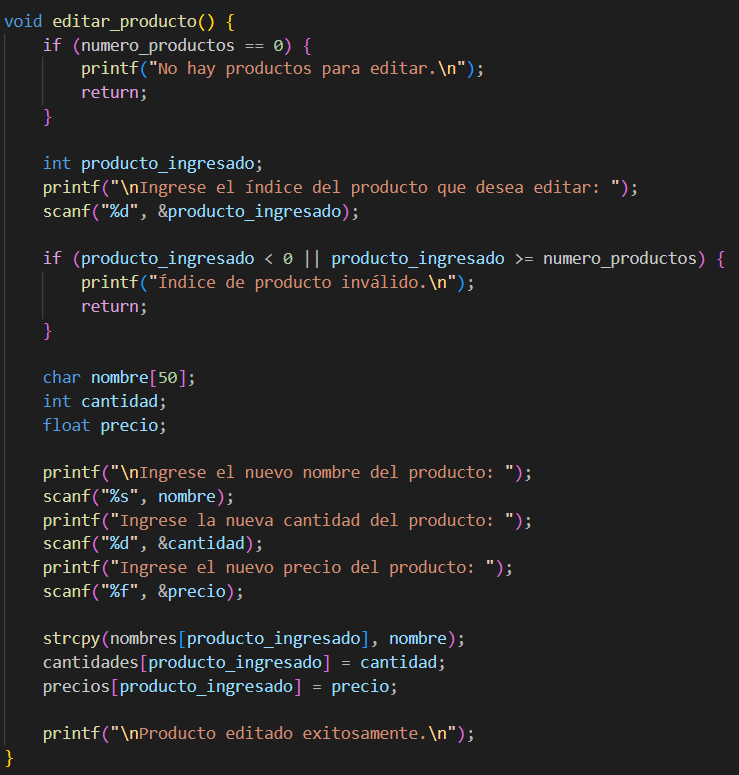
1. Este parte del código incluye las librerías necesarias para el desarrollo de las funciones, además declaramos algunas variables y arreglos globales, como también una macro que servirá como límite para ingresar productos.

****

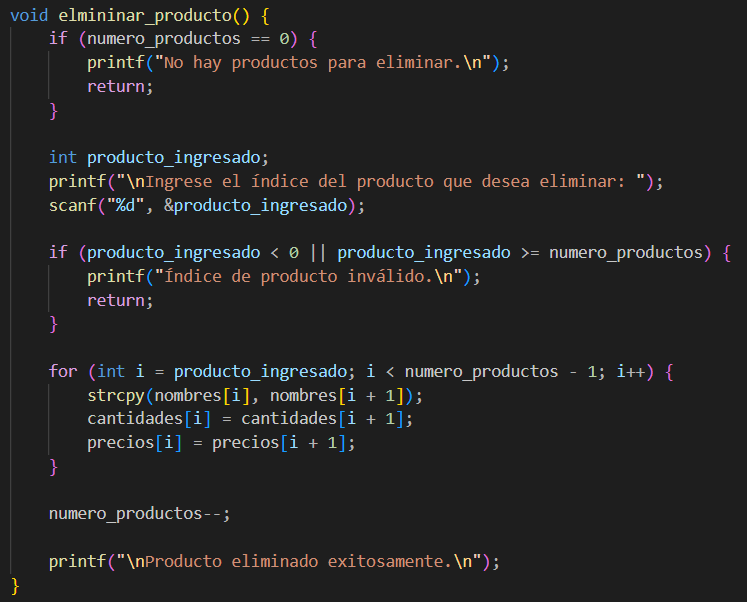
1. Hacemos uso de un procedimiento para implementar la funcionalidad de añadir productos, en la primera parte podemos ver una estructura if que nos ayudara a restringir la adicción de productos si se llega al máximo. Después declaramos algunas variables con las cuales les asignaremos datos mediante el usuario, por ultimo guardaremos la variable obtenida en su arreglo correspondiente, y también actualizamos el contador de números de productos para un próximo ingreso de datos. Consecuentemente mostraremos al usuario un mensaje para informar el ingreso de datos correcto.

****

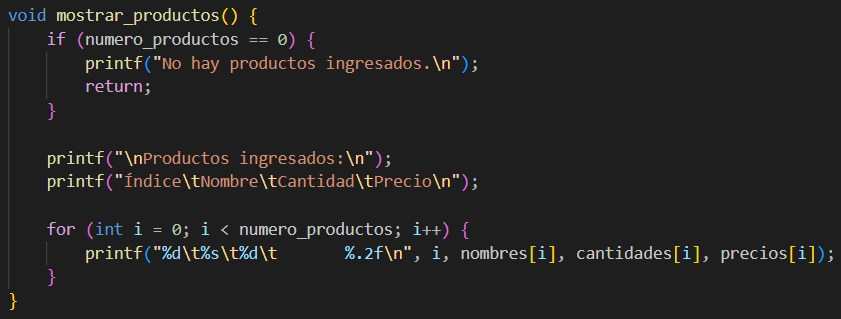
1. **Como en la explicación anterior, aquí también hacemos uso de un procedimiento, en este caso controlamos con un “if” que cuando no haya ningún producto ingresado, aparezca un mensaje informándonos que no hay producto para ingresar. En esta función declaramos una variable de tipo entera llamada “producto\_ingresado”, el cual nos ayudara a pedir el índice del producto que desea editar (el índice comienza en 0), después se hará una comprobación para ver si el índice que nos pide es menor a 0 o es mayor al número de productos existentes, lo cual no tendría sentido porque no existen, por lo tanto se nos mostrara un mensaje de índice de producto invalido, lo cual nos regresara al menú principal. Después de hacer la validación del índice, pediremos los datos nuevamente con variables antes vistas, y haremos el mismo procedimiento que en la función anterior, pero guardaremos en los arreglos en la posición correspondiente que nos indica el índice. Por ultimo mostraremos un mensaje información de que el producto se ha editado.**

****

1. **Comenzamos con la verificación de la existencia de productos con la estructura “if”, si no hay productos ingresados pues no se podrá eliminar. Seguiremos con la validación del indice, si es un indice existente o no, como vemos en la imagen nos mostrara un mensaje en caso de que el indice sea invalido. Despues mediante un bucle “for” recorreremos las listas de nombres, cantidades y precios a partir de la posición producto\_ingresado. Para cada elemento en estas listas, se copia el contenido del elemento siguiente en el mismo índice, efectivamente eliminando el producto en la posición producto\_ingresado. Por último se decrementa el contador “numero\_productos”, para asi poder reflejar la nueva cantidad de productos existentes, además de mostrar un mensaje indicando que se eliminó el producto.**

****

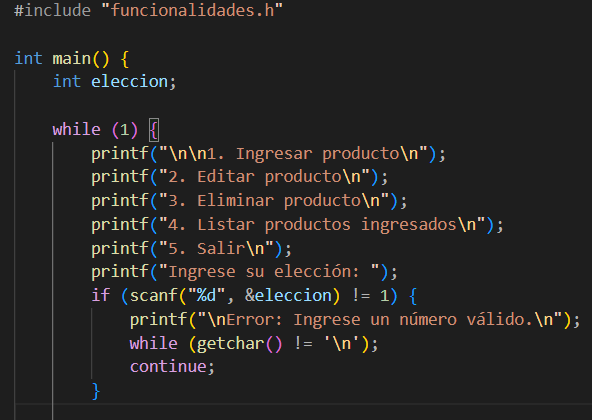
1. **En este procedimiento llamado “mostrar\_productos” primero verificamos si hay un producto para mostrar o no, después imprimos los productos ingresados, con ayuda de dos “printfs” y un bucle for, que recorrerá cada arreglo desde la posición cero hasta la cantidad de números existentes.**

****

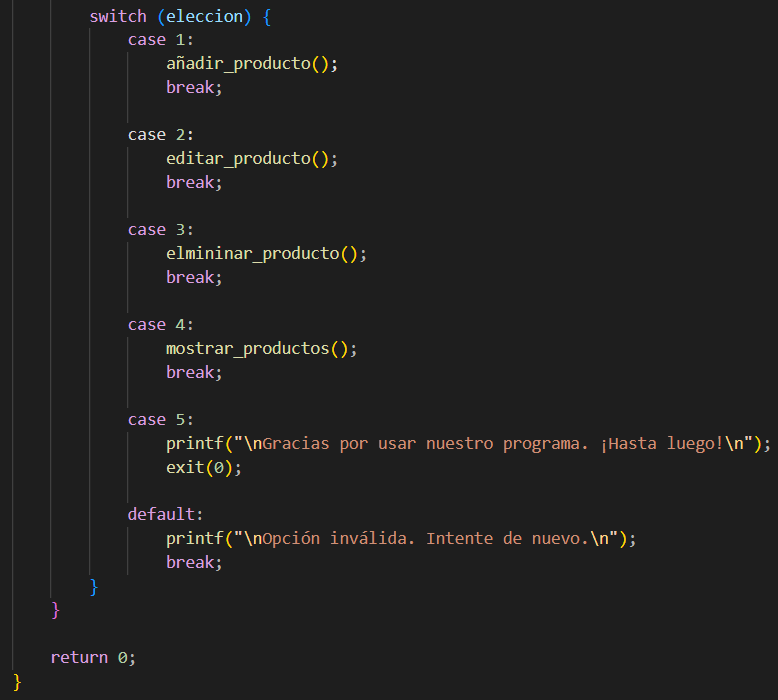
**Proyecto.c**1. Agregamos el archivo “funcionalidades”, la cual es una librería que contiene las funciones antes descritas.

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. Controlamos que el ingreso de datos de la variable elección sea un número, mediante un if, que comparara si el dato obtenido mediante la función scanf de tipo entero es diferente de 1, pues en términos técnicos, el 1 es verdadero, y el 0 es falso, y como vemos en la imagen estamos pidiendo un dato de tipo entero, si no es un numero mostraremos un mensaje de error, después para limpiar el búfer de entrada, utilizaremos un bucle “while” que utilizará la función “getchar” para leer caracteres del búfer hasta que se encuentra con un salto de línea. Esto nos ayudara a comprobar que caracteres no deseados se limpie del búfer antes de continuar. Por ultimo con la instrucción “continue” se reiniciará el bucle while para solicitar una nueva entrada después de eliminar el carácter no deseado.

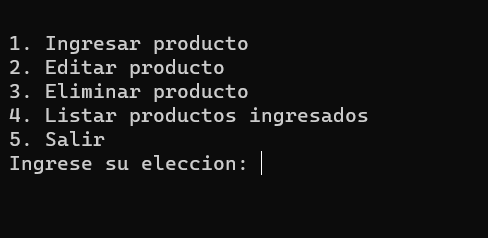
****

4. Ahora dentro de la estructura “while” y con el dato “elección” obtenido mediante el usuario, utilizaremos una estructura “switch” en la cual habrá 5 casos, los cuales se ejecutarán según la elección del usuario, además podemos ver que en en los primeros 4 casos están como instrucción ejecutar la función respectiva a lo que se pide, en el caso 5, que es la opción de salir, se mostrara un mensaje de salida, y cerraremos el programa mediante la instrucción “exit(0)”, y por ultimo tenemos el caso “default” que ocurrirá cuando el usuario ingrese un número que no ejecute ninguna instrucción, lo cual mostrada un mensaje de “intentar de nuevo” y nos llevara a la parte de pedir elegir al usuario que desea hacer.

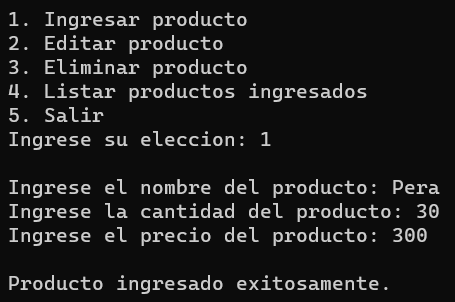
****

**Explicación de la ejecución de cada sección del programa**

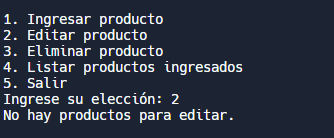
1. Al ejecutar el programa, nos aparece esta pantalla con un menú simple, en el cual hay 5 opciones a elegir por el usuario, y claro su mensaje pidiendo su elección.

****

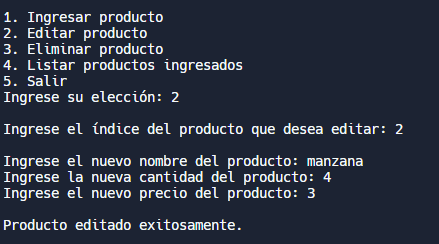
1. Si elegimos la primera opción, el programa nos pedirá el nombre, cantidad y precio del producto, consecuentemente nos aparecerá un mensaje de “producto ingresado “exitosamente” y se nos devolverá al menú.

****

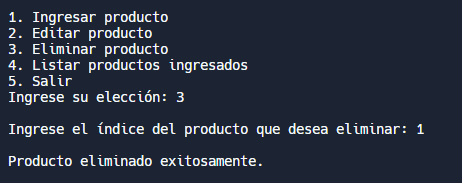
1. Si elegimos la segunda opción, nos pedirá ingresar el índice del producto que se quiere editar. Si no hay un producto ingresado entonces aparecerá el siguiente mensaje:



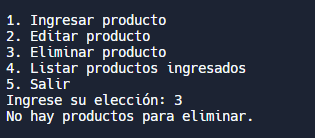
Por el contrario, si se ha ingresado cierta cantidad de productos al momento de editar la información podremos escoger entre los productos que se ingresaron y cambiar tanto el nombre, el precio y la cantidad para enlistarlo.

****

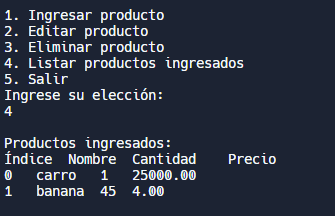
**4,-** Si se elige la tercera opción, nos pedirá ingresar el índice del producto que se desea eliminar de la lista:

****

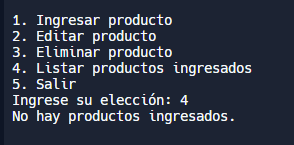
Si no hay productos que eliminar el programa mostrará el siguiente mensaje:

****

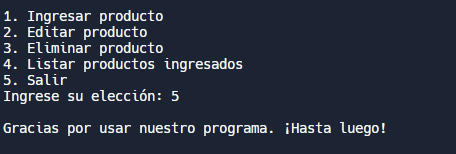
**5.-** Al elegir la cuarta opción, se desplegará la lista de los productos ingresados hasta el momento, junto con toda la información que se proporcionó.

****

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

****

**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 sistema de inventario es necesario para facilitar el almacenamiento de los productos de una tienda, para realizar correctamente este sistema se debe tener en cuenta el uso correcto de las sentencias de repetición, de condición y las funciones ya que de esta forma el código puede resultar más funcional.