Ingeniería en Ciberseguridad

Curso: Programación I

Tema: S6 – Taller Programa de gestión de inventario de productos

REALIZADO POR:

Juan Daniel Acosta

Daniel Argoti

DOCENTE

Ing. Rina Maribel Guerra Chiriboga

FECHA

22 de mayo, 2025

Contenido

1. Introducción

El programa desarrollado en C tiene con como objetivo ayudar a gestionar un inventario pequeño de productos. A través de un menú interactivo, el usuario puede ingresar hasta diez productos con sus respectivos precios, y luego acceder a varias funciones útiles como calcular el precio total del inventario, encontrar el producto más caro y el más barato, conocer el precio promedio de todos los productos, o buscar uno en específico para consultar su precio. Aunque es un sistema básico, puede ser muy útil en negocios pequeños, ferias estudiantiles o cualquier situación en la que se necesite tener un control rápido y claro de productos y precios. Su diseño busca ser práctico y fácil de usar, sin necesidad de conocimientos avanzados en informática. Antes de comenzar a programar, uno de los retos fue pensar en cómo organizar los datos para que todo funcionara de forma ordenada. También se tomó en cuenta la importancia de validar bien los datos que el usuario ingresa, para evitar errores que puedan afectar los resultados. Este proyecto fue una buena oportunidad para poner en práctica lo aprendido en programación, y demostrar cómo con funciones simples se pueden resolver necesidades reales de forma efectiva.

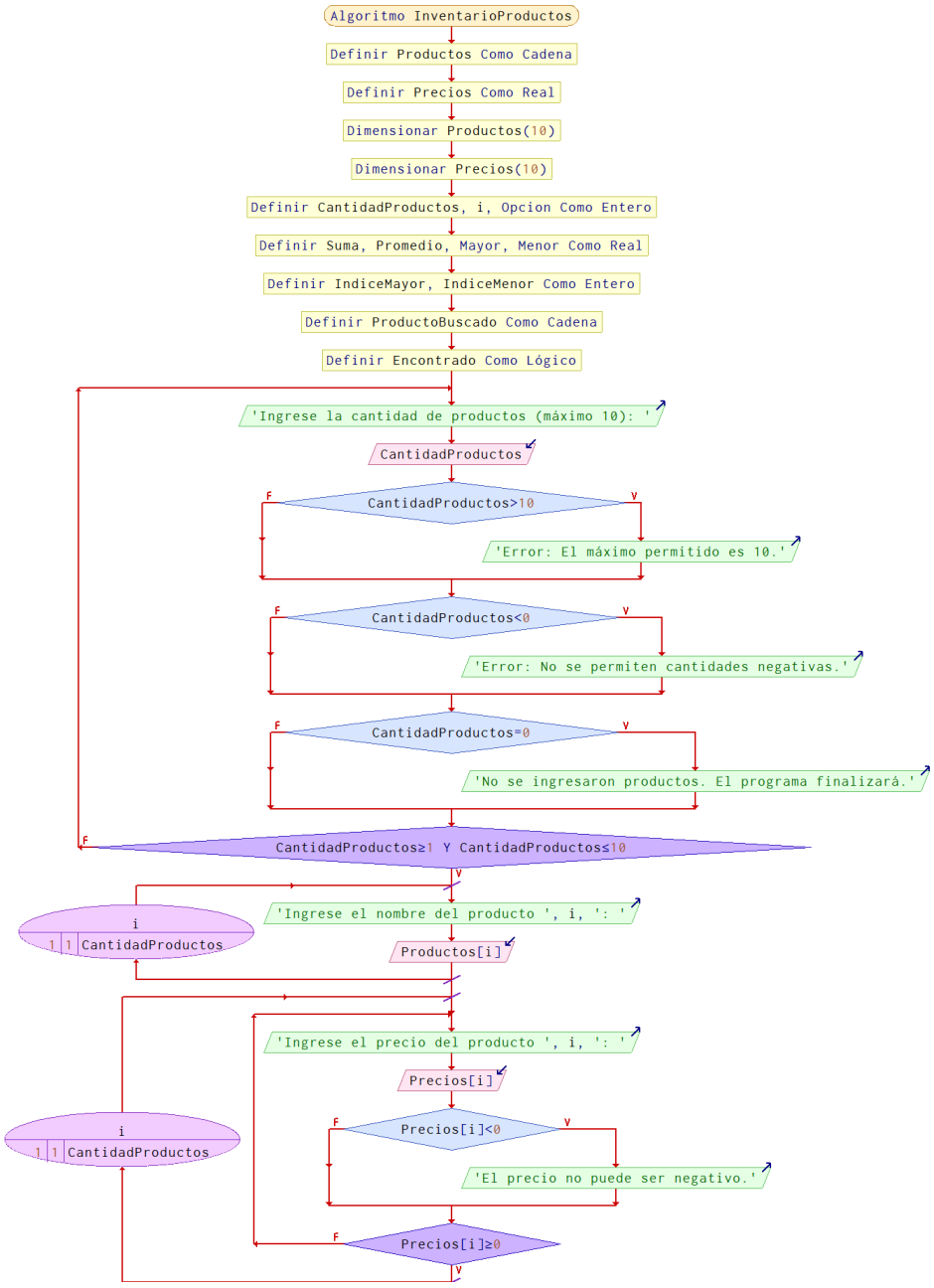
2. Análisis del Problema y Diseño de la Solución

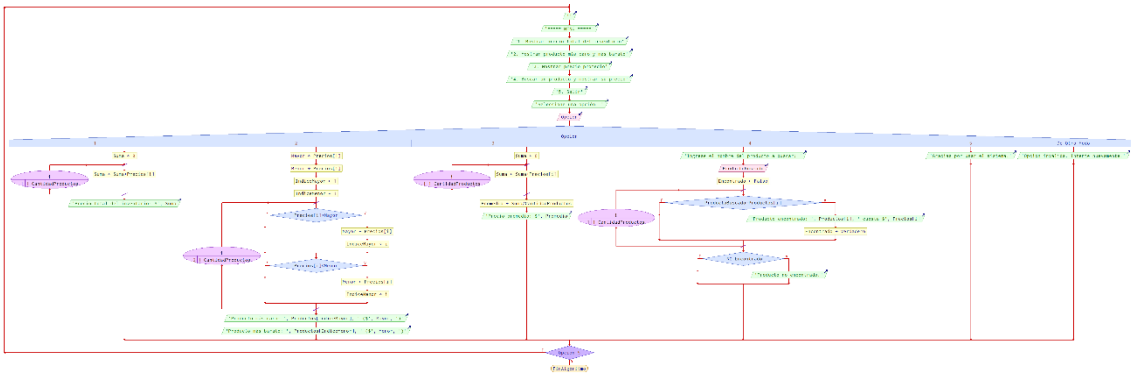
Antes de comenzar a programar, lo primero fue entender bien lo que se quería lograr: un sistema que permitiera registrar productos y sus precios, y que además pudiera mostrar información útil como el precio total del inventario, el precio promedio, el producto más caro y el más barato, y también buscar un producto específico por su nombre. El enfoque debía ser simple, pero funcional. Como parte de los requisitos, no se podían utilizar estructuras ni punteros, por lo que fue necesario encontrar una solución que cumpliera con esa condición. Para esto, se optó por usar dos arreglos paralelos: uno para almacenar los nombres de los productos y otro para los precios. En el arreglo de nombres se usó un arreglo bidimensional de tipo char, ya que cada nombre es una cadena de texto. Para los precios, se usó un arreglo de tipo float. Cada posición de ambos arreglos corresponde a un producto, es decir, el producto en la posición 0 del arreglo de nombres tiene su precio en la posición 0 del arreglo de precios, y así sucesivamente.

También se creó una variable para almacenar la cantidad total de productos, la cual se pide al inicio del programa. Como máximo se pueden ingresar 10 productos, lo que ayuda a mantener el código manejable. Aparte de eso, se incluyó un menú que permite al usuario elegir qué acción desea realizar, repitiéndose hasta que el usuario decida salir del programa. Este menú se controla

con una estructura Do While y un switch que dirige la ejecución según la opción seleccionada. Durante el diseño se pensaron dos posibles formas de resolver el problema:

- **Primera opción (la implementada):** Usar arreglos paralelos para nombres y precios. Esta opción se ajusta perfectamente a los requisitos del proyecto, ya que no utiliza estructuras ni punteros. Es fácil de implementar y suficiente para el tipo de operaciones que el sistema debe realizar.





- **Segunda opción (descartada por los requisitos):** Utilizar estructuras para agrupar el nombre y el precio de cada producto. Aunque esta alternativa permitiría un código más ordenado y fácil de escalar, no fue posible aplicarla debido a las restricciones del proyecto.

En resumen, se eligió una solución práctica y compatible con las condiciones establecidas. El uso de arreglos paralelos permitió desarrollar un programa claro, funcional y fácil de entender, cumpliendo con todos los objetivos planteados.

3. Implementación del Código

Para el desarrollo de este programa se lo estructuró en tres archivos: uno con las funciones, otro con el archivo header .h que contiene las declaraciones, y el archivo principal main. Esta organización permite que el código sea más ordenado, modular y fácil de mantener. El programa comienza solicitando al usuario la cantidad de productos a ingresar. Luego, se capturan los nombres y precios usando dos arreglos paralelos: uno bidimensional de tipo char para los nombres y otro de tipo float para los precios. Para evitar errores comunes al ingresar cadenas con fgets, se utilizó una línea clave que elimina el salto de línea al final del texto ingresado:

```
36          Productos[i][strcspn(Productos[i], "\n")] = '\0';
```

Esto asegura que, al imprimir los productos junto con su precio, todo aparezca en una sola línea y no haya espacios innecesarios ni saltos de línea no deseados. Esta misma lógica se aplicó en la función de búsqueda de productos, donde se usó:

```
120     ProductoPorBuscar[strcspn(ProductoPorBuscar, "\n")] = '\0';
```

De esta manera, se puede comparar correctamente lo que el usuario escribe con los nombres almacenados en el arreglo. Para realizar la búsqueda del producto, se utilizó la función strcmp de la biblioteca string.h, que compara cadenas carácter por carácter:

```
124             if (strcmp(Productos[i], ProductoPorBuscar)==0)
```

4. Validación y Pruebas

Después de terminar el código, se realizaron varias pruebas para asegurarse de que el programa funcionara correctamente y respondiera bien ante diferentes tipos de entradas. Una de las primeras validaciones importantes fue la cantidad de productos. El programa no permite ingresar más de 10, y si el usuario intenta hacerlo, muestra un mensaje de advertencia:

```
Ingrese la cantidad de productos que desea agregar: 11
El numero maximo de productos que el sistema maneja es de 10. Intentelo nuevamente.
Ingrese la cantidad de productos que desea agregar:
```

También evita que se ingresen números negativos o cero, lo cual es útil para que no se guarden datos sin sentido. Otro punto clave fue la validación de los precios. Se probó ingresando valores negativos, y el programa respondió como se esperaba, pidiendo al usuario que introduzca un precio válido:

```
Ingrese el precio del producto 2: 1
Ingrese el precio del producto 3: -0.50
El precio del producto no puede ser negativo. Intentelo nuevamente.
Ingrese el precio del producto 3:
```

Esto ayuda a evitar errores al momento de hacer cálculos como el total del inventario o el precio promedio. Se hicieron pruebas con los productos “Pastel”, “Refresco” y “Caramelo” y se les asignaron precios de $5, $1 y $0.50 respectivamente. En cada una de las opciones, los cálculos fueron realizados correctamente:

```
Seleccione una opcion: 1
El precio total del inventario es: $6.50
```

```
Seleccione una opcion: 2
El producto mas caro es Pastel con un precio de: $5.00
El producto mas barato es Caramelo con un precio de: $0.50
```

```
Seleccione una opcion: 3
El precio promedio de todos los productos es de: $2.17
```

Durante la prueba de la función de búsqueda, se verificó que, al escribir el nombre de un producto, el sistema lo reconociera correctamente:

```
Seleccione una opcion: 4
Ingrese el nombre del producto que desea buscar: Refresco
El precio del producto Refresco es: $1.00
```

Con esto, fue posible corroborar que el programa se comportaba de acuerdo a lo esperado.

5. Conclusiones y Mejoras Futuras

Este proyecto fue una buena oportunidad para aplicar lo aprendido en programación básica con C. A pesar de ser un programa sencillo, permitió trabajar con varios conceptos importantes como el uso de arreglos, funciones, validación de datos y estructuras de control como ciclos y condicionales. Uno de los mayores retos fue encontrar una forma de organizar los datos sin usar estructuras ni punteros, ya que el programa debía cumplir con esa restricción. También quedó clara la importancia de validar correctamente lo que el usuario ingresa, ya que pequeños errores pueden afectar los resultados finales. A través de pruebas, se pudo comprobar que el sistema responde bien a situaciones inesperadas, como precios negativos o nombres mal escritos. En cuanto a posibles mejoras, una futura versión del programa podría permitir editar o eliminar productos después de ingresarlos, o incluso guardar la información en archivos para no perder los datos al cerrar el programa. Además, si se levantaran las restricciones del proyecto, se podría usar estructuras para agrupar el nombre y el precio de cada producto, haciendo el código más organizado y fácil de mantener. Por último, agregar una interfaz gráfica facilitaría el uso del programa.