UNIVERSIDAD AMERICANA

ASIGNATURA:

Logica y Algoritmos

INTEGRANTES:

1. Armando Jose Valdivia Reyes
2. Oscar Iván Alvarado Jirón
3. Manuel Alejandro Jirón Martínez
4. Inti Alexander Montalván Góngora

DOCENTE:

Jose Duran Garcia

Managua, Nicaragua 2 de julio del 2024

INTRODUCCIÓN

A lo largo de la documentación de nuestro proyecto nos basamos en distintas cosas que logramos aprender de la clase de lógica y algoritmo. De manera que buscando satisfacer las necesidades de la farmacia "Dayer" usando los distintos conocimientos adquiridos desarrollamos un programa de consola de gestión de productos utilizando la estructura explicada a continuación.

Registro de productos
Venta de producto
Análisis del mercado

Código en C++ para la gestión de un inventario de productos

El código fuente implementa las funciones para gestionar un inventario de productos. Este código se encuentra dividido en varias secciones que permiten registrar, mostrar, editar y eliminar productos del inventario.

Descripción detallada de las funciones

1. **ingresarDatosProductos()**:
    - **Propósito**: Permite al usuario ingresar los datos de múltiples productos y los añade al inventario.
    - **Proceso**:
        - Limpia la pantalla.
        - Solicita al usuario la cantidad de productos a registrar.
        - Itera para cada producto, solicitando y almacenando su ID, nombre, marca, clasificación, nicho de mercado, precio y cantidad.
        - Agrega cada producto al arreglo inventario.
        - Incrementa el contador numProductos.
    - **Salida**: Mensajes de confirmación después de registrar cada producto.
2. **cargarInventario()**:
    - **Propósito**: Carga los datos del inventario desde un archivo de texto (inventario.txt) al arreglo inventario.
    - **Proceso**:
        - Intenta abrir el archivo inventario.txt para lectura.

        - Lee los datos de cada producto del archivo y los almacena en el arreglo inventario.
        - Cierra el archivo después de la lectura.
    - **Salida**: Retorna 1 si la carga del inventario fue exitosa, 0 si falló.
3. **guardarInventario()**:
    - **Propósito**: Guarda todos los productos del inventario actual en un archivo de texto (inventario.txt).
    - **Proceso**:
        - Abre el archivo inventario.txt para escritura.
        - Escribe los datos de cada producto (ID, nombre, marca, clasificación, nicho de mercado, precio, cantidad) en el archivo.
        - Cierra el archivo después de la escritura.
    - **Salida**: Mensajes de error si no se puede abrir el archivo.
4. **mostrarInventario()**:
    - **Propósito**: Muestra todos los productos almacenados en el inventario.
    - **Proceso**:
        - Imprime una lista numerada de los nombres de los productos almacenados.
        - Solicita al usuario seleccionar un producto por su número para mostrar sus detalles (ID, nombre, marca, clasificación, nicho de mercado, precio, cantidad).
        - Muestra los detalles del producto seleccionado.
    - **Salida**: Mensajes de error si la opción seleccionada no es válida.
5. **editarProducto()**:
    - **Propósito**: Permite al usuario editar los detalles de un producto existente en el inventario por su ID.
    - **Proceso**:
        - Solicita al usuario ingresar el ID del producto a editar.
        - Busca el producto por su ID en el arreglo inventario.
        - Permite al usuario editar el nombre, marca, clasificación, nicho de mercado, precio y cantidad del producto.
        - Guarda los cambios realizados en el inventario.

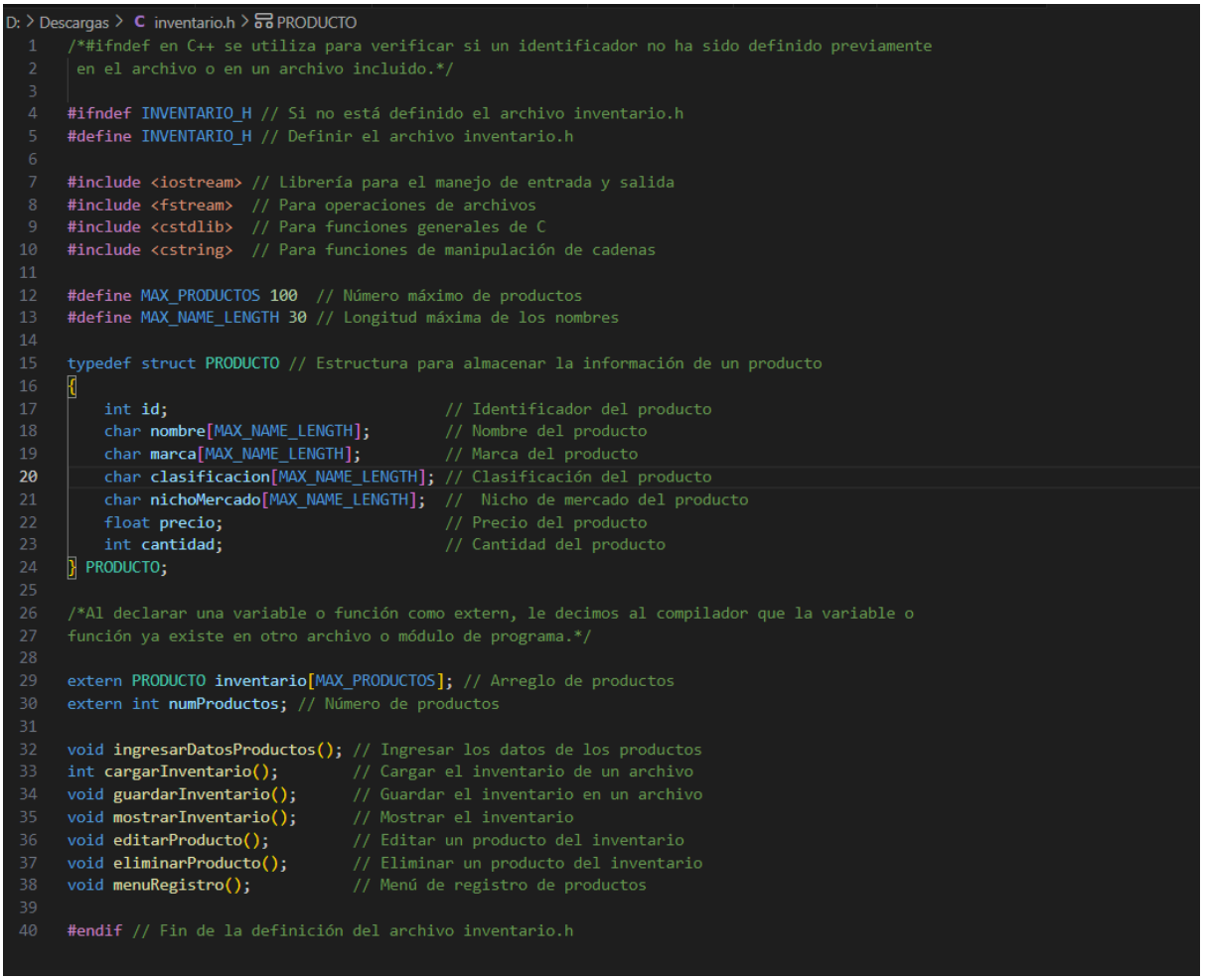
- **Salida**: Mensajes de confirmación después de editar el producto o mensaje de error si el producto no se encuentra.

6. **eliminarProducto()**:
    - **Propósito**: Permite al usuario eliminar un producto del inventario por su ID.
    - **Proceso**:
        - Solicita al usuario ingresar el ID del producto a eliminar.
        - Busca el producto por su ID en el arreglo inventario.
        - Elimina el producto del arreglo y ajusta el tamaño del inventario.
        - Guarda los cambios realizados en el inventario.
    - **Salida**: Mensajes de confirmación después de eliminar el producto o mensaje de error si el producto no se encuentra.
7. **menuInventario()**:
    - **Propósito**: Muestra un menú de opciones para gestionar el inventario de productos.
    - **Proceso**:
        - Limpia la pantalla.
        - Muestra las opciones del menú (ingresar datos de productos, mostrar inventario, editar producto, eliminar producto, volver).
        - Solicita al usuario seleccionar una opción del menú.
        - Ejecuta la función correspondiente según la opción seleccionada.
    - **Salida**: Mensajes de error si la opción seleccionada no es válida.

Estas funciones trabajan en conjunto para permitir la gestión completa del inventario de productos, incluyendo la carga desde y guardado en archivos de texto, la edición y eliminación de productos, y la visualización detallada y general del inventario.

Archivo de encabezado (inventario.h)

Este archivo define las estructuras y constantes necesarias para la gestión del inventario de productos.



Descripción detallada de las partes del archivo inventario.h:

1. **Directiva de preprocesador #ifndef y #define**:
    - **Propósito**: Evita la inclusión múltiple del archivo inventario.h en el mismo programa.
    - **Funcionamiento**: Si INVENTARIO_H no está definido (es decir, no ha sido incluido previamente), se define INVENTARIO_H. Esto asegura que el contenido del archivo se procese solo una vez durante la compilación.
2. **Inclusión de librerías estándar**:
    - iostream: Para manejo estándar de entrada y salida.
    - fstream: Para operaciones de archivos.
    - cstdlib: Para funciones generales de C.
    - cstring: Para funciones de manipulación de cadenas.

3. **Definición de constantes**:
    - #define MAX_PRODUCTOS 100: Define el número máximo de productos que pueden ser gestionados en el inventario.
    - #define MAX_NAME_LENGTH 100: Define la longitud máxima para los nombres de productos y otras cadenas relacionadas.
4. **Estructura de datos**:
    - typedef struct PRODUCTO { ... } PRODUCTO;: Define una estructura PRODUCTO que contiene varios campos para almacenar información detallada sobre cada producto en el inventario. Incluye campos como id, nombre, marca, clasificacion, nichoMercado, precio y cantidad.
5. **Declaraciones externas**:
    - extern PRODUCTO inventario[MAX_PRODUCTOS];: Declaración externa del arreglo inventario, que almacenará todos los productos disponibles.
    - extern int numProductos;: Declaración externa de la variable numProductos, que llevará el conteo del número actual de productos registrados en el inventario.
6. **Prototipos de funciones**:
    - **ingresarDatosProductos()**: Prototipo de función para ingresar datos de nuevos productos al inventario.
    - **cargarInventario()**: Prototipo de función para cargar los datos del inventario desde un archivo.
    - **guardarInventario()**: Prototipo de función para guardar los datos del inventario en un archivo.
    - **mostrarInventario()**: Prototipo de función para mostrar todos los productos en el inventario.
    - **editarProducto()**: Prototipo de función para editar la información de un producto existente en el inventario.
    - **eliminarProducto()**: Prototipo de función para eliminar un producto del inventario.
    - **menuRegistro()**: Prototipo de función para mostrar el menú de opciones relacionadas con el registro y gestión de productos.
7. **Directiva de preprocesador #endif**:

- **#endif**: Marca el final de la definición condicional de INVENTARIO_H. Todo el contenido entre #ifndef y #endif se incluirá solo si INVENTARIO_H no está definido previamente.

Este archivo de encabezado es esencial para organizar y facilitar la gestión del inventario de productos en un sistema más amplio de administración empresarial o comercial. Proporciona las estructuras de datos y funciones necesarias para manipular productos, realizar operaciones de carga y guardado desde y hacia archivos, así como interactuar con el usuario a través de menús y opciones específicas.

Código en C++ para la gestión de ventas de productos

Este código implementa las funciones necesarias para registrar y gestionar ventas de productos. A continuación, se explica cada función del código en detalle.

Descripción detallada de las funciones

1. **registrarVenta**:
   - **Propósito**: Registrar una nueva venta en el sistema.
   - **Proceso**:
     1. Limpia la consola.
     2. Solicita el ID del producto vendido.
     3. Busca el producto en el inventario.
     4. Verifica que haya suficiente stock del producto.
     5. Solicita la cantidad vendida y el nombre del cliente.
     6. Actualiza la cantidad en el inventario.
     7. Guarda la venta en el arreglo de ventas.
     8. Guarda los cambios en el inventario y las ventas en sus respectivos archivos.
   - **Salida**: Mensajes de confirmación o error según sea el caso.
2. **guardarVentas**:
   - **Propósito**: Guardar todas las ventas registradas en un archivo de texto (ventas.txt).
   - **Proceso**:

1. Abre el archivo ventas.txt en modo de anexado.
2. Verifica si el archivo se abrió correctamente.
3. Escribe los datos de cada venta en el archivo.
4. Cierra el archivo.

- **Salida**: Mensajes de error si no se puede abrir el archivo.

3. **cargarVentas**:
   - **Propósito**: Cargar las ventas desde el archivo de texto (ventas.txt) al arreglo de ventas.
   - **Proceso**:
     1. Abre el archivo ventas.txt en modo de lectura.
     2. Verifica si el archivo se abrió correctamente.
     3. Lee los datos de cada venta del archivo y los almacena en el arreglo de ventas.
     4. Cierra el archivo.
   - **Salida**: Retorna 1 si se cargaron las ventas exitosamente, 0 en caso contrario.
4. **mostrarVentas**:
   - **Propósito**: Mostrar todas las ventas registradas.
   - **Proceso**:
     1. Limpia la consola.
     2. Recorre el arreglo de ventas.
     3. Muestra los datos de cada venta (ID del producto, nombre del producto, cantidad vendida y cliente).
   - **Salida**: Lista de ventas en la consola.
5. **menuVentas**:
   - **Propósito**: Mostrar un menú de opciones para gestionar las ventas.
   - **Proceso**:
     1. Limpia la consola.
     2. Muestra las opciones del menú (registrar venta, mostrar ventas, volver).
     3. Solicita al usuario que seleccione una opción.
     4. Ejecuta la función correspondiente según la opción seleccionada.
   - **Salida**: Ejecuta la función correspondiente o muestra un mensaje de error si la opción no es válida.

Archivo de encabezado (ventas.h)

D: > Descargas > C ventas.h > ...

```c
/*#ifndef en C++ se utiliza para verificar si un identificador no ha sido definido previamente
 en el archivo o en un archivo incluido.*/

#ifndef VENTAS_H // Si no está definido el archivo ventas.h
#define VENTAS_H // Definir el archivo ventas.h

#include <iostream> // Librería para el manejo de entrada y salida
#include <fstream> // Para operaciones de archivos
#include <cstdlib> // Para funciones generales de C
#include <cstring> // Para funciones de manipulación de cadenas
#include "inventario.h" // Incluir el archivo inventario.h

#define MAX_VENTAS 100 // Definir el número máximo de ventas

typedef struct VENTA // Estructura para almacenar la información de una venta
{
    int productoId; // Identificador del producto
    char productoVendido[MAX_NAME_LENGTH]; // Nombre del producto
    int cantidadVendida; // Cantidad vendida
    char cliente[MAX_NAME_LENGTH]; // Nombre del cliente
} VENTA;

/*Al declarar una variable o función como extern, le decimos al compilador que la variable o
función ya existe en otro archivo o módulo de programa.*/

extern VENTA ventas[MAX_VENTAS]; // Arreglo de ventas
extern int numVentas; // Número de ventas

void registrarVenta(); // Registrar una venta
void guardarVentas(); // Guardar las ventas en un archivo
int cargarVentas(); // Cargar las ventas de un archivo
void mostrarVentas(); // Mostrar las ventas
void menuVentas(); // Menú de ventas

#endif // Fin de la definición del archivo ventas.h
```

Descripción detallada de las partes del archivo ventas.h:

1. **Directiva de preprocesador #ifndef y #define**:
    - **Propósito**: Evita la inclusión múltiple del archivo ventas.h en el mismo programa.
    - **Funcionamiento**: Si VENTAS_H no está definido (es decir, no ha sido incluido previamente), se define VENTAS_H. Esto asegura que el contenido del archivo se procese solo una vez durante la compilación.
2. **Inclusión de librerías estándar**:
    - iostream: Para manejo estándar de entrada y salida.
    - fstream: Para operaciones de archivos.
    - cstdlib: Para funciones generales de C.
    - cstring: Para funciones de manipulación de cadenas.
3. **Inclusión del archivo de encabezado inventario.h**:

- #include "inventario.h": Permite acceder a las estructuras y funciones definidas en inventario.h, necesarias para la gestión de inventarios dentro de las funciones de ventas.

4. **Definición de constantes**:
   - #define MAX_VENTAS 100: Define el número máximo de ventas que pueden almacenarse en el arreglo ventas.
5. **Declaraciones externas**:
   - extern VENTA ventas[MAX_VENTAS];: Declara externamente el arreglo ventas, que almacenará las ventas realizadas.
   - extern int numVentas;: Declara externamente la variable numVentas, que llevará el conteo del número actual de ventas registradas.
6. **Definiciones de funciones**:
   - **registrarVenta()**: Prototipo de función para registrar una venta.
   - **guardarVentas()**: Prototipo de función para guardar las ventas en un archivo.
   - **cargarVentas()**: Prototipo de función para cargar las ventas desde un archivo.
   - **mostrarVentas()**: Prototipo de función para mostrar las ventas registradas.
   - **menuVentas()**: Prototipo de función para mostrar el menú de opciones relacionadas con ventas.
7. **Directiva de preprocesador #endif**:
   - **#endif**: Marca el final de la definición condicional de VENTAS_H. Todo el contenido entre #ifndef y #endif se incluirá solo si VENTAS_H no está definido previamente.

Estas estructuras y funciones permiten organizar y gestionar las ventas de productos de manera eficiente en un sistema más amplio de gestión de inventario y ventas.

Código en C++ para la gestión de un análisis de productos

Descripción detallada de las funciones

1. **productosMasVendidos()**

- **Descripción**: Esta función calcula y muestra el producto más vendido en base a las ventas registradas.
- **Proceso**:
  - Verifica si hay ventas registradas. Si no hay ventas, muestra un mensaje y retorna.
  - Inicializa un arreglo cantidadVendida para almacenar la cantidad vendida de cada producto.
  - Recorre todas las ventas y para cada venta busca el producto correspondiente en el inventario.
  - Incrementa la cantidad vendida del producto en cantidadVendida.
  - Encuentra el índice del producto con la mayor cantidad vendida.
  - Muestra el nombre y la cantidad vendida del producto más vendido.

2. **productoMenosVendido()**
   - **Descripción**: Calcula y muestra el producto menos vendido en base a las ventas registradas.
   - **Proceso**:
     - Verifica si hay ventas registradas. Si no hay ventas, muestra un mensaje y retorna.
     - Inicializa un arreglo ventasPorProducto para almacenar las ventas por producto.
     - Recorre todas las ventas y para cada venta busca el producto correspondiente en el inventario.
     - Incrementa las ventas del producto en ventasPorProducto.
     - Encuentra el índice del producto con la menor cantidad vendida.
     - Muestra el nombre y la cantidad vendida del producto menos vendido.
3. **calcularIngresosTotales()**
   - **Descripción**: Calcula y muestra los ingresos totales generados por todas las ventas registradas.
   - **Proceso**:
     - Inicializa una variable ingresos para acumular los ingresos.
     - Recorre todas las ventas y para cada venta busca el producto correspondiente en el inventario.

        - Calcula el ingreso de cada venta como el producto de la cantidad vendida y el precio del producto.
        - Suma el ingreso al total acumulado en ingresos.
        - Muestra el ingreso total calculado.
4. **clientesMasFrecuentes()**
    - **Descripción**: Encuentra y muestra al cliente que más veces ha realizado compras.
    - **Proceso**:
        - Inicializa variables para almacenar la frecuencia máxima (maxFrecuencia) y el nombre del cliente más frecuente (clienteMasFrecuente).
        - Para cada venta, cuenta cuántas veces aparece cada cliente.
        - Actualiza maxFrecuencia y clienteMasFrecuente si encuentra un cliente con mayor frecuencia.
        - Muestra el nombre del cliente más frecuente y la cantidad de compras realizadas.
5. **stockCritico()**
    - **Descripción**: Muestra los productos que tienen una cantidad en inventario igual o menor a 5.
    - **Proceso**:
        - Recorre todos los productos en el inventario.
        - Verifica si la cantidad en inventario de cada producto es igual o menor a 5.
        - Muestra los detalles del producto si se encuentra en stock crítico.
6. **menuAnalisis()**
    - **Descripción**: Presenta un menú interactivo para que el usuario seleccione qué análisis desea realizar.
    - **Proceso**:
        - Muestra un menú con opciones para realizar análisis específicos.
        - Lee la opción seleccionada por el usuario.
        - Llama a la función correspondiente según la opción seleccionada hasta que el usuario elija salir.

Consideraciones adicionales:

- **Uso de system("cls")**: Se utiliza para limpiar la pantalla de la consola y mostrar los resultados de manera organizada.
- **Interacción con datos globales (ventas y inventario)**: Estas funciones hacen uso de los arreglos globales ventas y inventario, que deben estar definidos y actualizados desde otros archivos para que funcionen correctamente.
- **Manejo de iteraciones y comparaciones**: Las funciones utilizan bucles anidados para comparar y calcular datos en base a las ventas y al inventario.

Estas funciones son esenciales para realizar análisis detallados sobre las ventas, ingresos, clientes y estado del inventario, proporcionando información clave para la toma de decisiones en la gestión de productos y ventas.

Archivo de encabezado (analisis.h)

El archivo de encabezado analisis.h proporciona las declaraciones de funciones necesarias para realizar análisis sobre las ventas y el inventario de productos.

D: > Descargas > C analisis.h > ...

```
1   /*#ifndef en C++ se utiliza para verificar si un identificador no ha sido definido previamente
2    en el archivo o en un archivo incluido.*/
3
4   #ifndef ANALISIS_H // Para evitar errores si se llegase a definir dos veces la misma librería
5   #define ANALISIS_H // Definir la librería
6
7   #include <iostream> // Librería para el manejo de entrada y salida
8   #include <fstream> // Para operaciones de archivos
9   #include <cstdlib> // Para funciones generales de C
10  #include <cstring> // Para funciones de manipulación de cadenas
11  #include "inventario.h" // Incluir el archivo inventario.h
12  #include "ventas.h" // Incluir el archivo ventas.h
13
14  void productosMasVendidos(); // Función para mostrar los productos más vendidos
15  void productoMenosVendido(); // Función para mostrar el producto menos vendido
16  void calcularIngresosTotales(); // Función para calcular los ingresos totales
17  void clientesMasFrecuentes(); // Función para mostrar los clientes más frecuentes
18  void stockCritico(); // Función para mostrar los productos en stock crítico
19  void menuAnalisis(); // Función para mostrar el menú de análisis
20
21  #endif // Fin de la librería
```

Funciones del archivo analisis.h

1. **productosMasVendidos()**
    - **Descripción**: Esta función calcula y muestra el producto más vendido basándose en las ventas registradas.
    - **Proceso**: Utiliza los arreglos globales ventas e inventario para determinar la cantidad vendida de cada producto y encuentra el producto con la mayor cantidad vendida.
2. **productoMenosVendido()**
    - **Descripción**: Calcula y muestra el producto menos vendido basándose en las ventas registradas.
    - **Proceso**: Utiliza los arreglos globales ventas e inventario para determinar la cantidad vendida de cada producto y encuentra el producto con la menor cantidad vendida.
3. **calcularIngresosTotales()**
    - **Descripción**: Calcula y muestra los ingresos totales generados por todas las ventas registradas.
    - **Proceso**: Utiliza los arreglos globales ventas e inventario para calcular los ingresos totales sumando el producto de la cantidad vendida y el precio de cada producto.
4. **clientesMasFrecuentes()**
    - **Descripción**: Encuentra y muestra al cliente que más veces ha realizado compras.
    - **Proceso**: Utiliza el arreglo global ventas para contar la frecuencia de cada cliente y determinar cuál tiene la mayor cantidad de compras registradas.
5. **stockCritico()**
    - **Descripción**: Muestra los productos que tienen una cantidad en inventario igual o menor a 5.
    - **Proceso**: Utiliza el arreglo global inventario para encontrar los productos cuya cantidad en inventario es menor o igual a 5 y los muestra.
6. **menuAnalisis()**
    - **Descripción**: Presenta un menú interactivo para que el usuario seleccione qué análisis desea realizar.
    - **Proceso**: Utiliza la biblioteca estándar de C++ para manejar la entrada y salida, mostrando opciones al usuario y llamando a las funciones correspondientes según la opción seleccionada.

Consideraciones adicionales

- **Guardia de inclusión (#ifndef, #define, #endif)**: Evita problemas de inclusión múltiple al asegurarse de que el contenido del archivo analisis.h solo se incluya una vez en un archivo fuente.
- **Uso de funciones y arreglos globales**: Las funciones en analisis.h dependen de los arreglos globales ventas e inventario, que deben ser definidos y actualizados en otros archivos para que las funciones funcionen correctamente.
- **Interacción con la consola y manipulación de datos**: Se utiliza iostream para manejar la entrada/salida estándar y fstream para operaciones de archivos, además de cstring para manipular cadenas de caracteres.

Este archivo encapsula funcionalidades clave para analizar y gestionar datos relacionados con ventas y productos, facilitando el análisis y la toma de decisiones en la gestión empresarial.