**1. Introducción**

**1.1 Propósito del Programa**

El programa tiene como objetivo principal gestionar productos en una lista doblemente enlazada. Mediante una interfaz de consola, se permiten varias operaciones esenciales sobre los productos, tales como su inserción, eliminación, ordenamiento por distintos criterios y la posibilidad de persistir estos datos a través de archivos. Este tipo de gestión es particularmente útil para sistemas de inventarios, bases de datos sencillas o sistemas que requieren la manipulación eficiente de grandes cantidades de información.

El uso de estructuras de datos como listas enlazadas dobles permite optimizar la memoria y los tiempos de acceso, lo que hace que este programa sea una solución efectiva y flexible para sistemas de control de inventarios o bases de datos pequeñas.

**1.2 Justificación**

Este proyecto se justifica por la necesidad de crear una solución sencilla y funcional para la gestión de productos de manera eficiente. En lugar de depender de bases de datos complejas, la implementación de una lista doblemente enlazada permite manejar un número razonable de productos con la flexibilidad de una estructura dinámica. La implementación de algoritmos de ordenamiento y la opción de persistir los datos en archivos, agregan funcionalidad para su uso en escenarios del mundo real.

El enfoque con listas doblemente enlazadas permite realizar operaciones de inserción, eliminación y modificación de elementos de manera rápida y eficiente, a diferencia de otras estructuras como arreglos estáticos o listas simplemente enlazadas, que podrían tener limitaciones en ciertos contextos.

**2. Requisitos**

**2.1 Librerías Necesarias**

**2.1.1 <iostream>**

La librería <iostream> es crucial para el programa ya que proporciona las funcionalidades de entrada y salida estándar. Mediante esta librería, se pueden realizar operaciones como mostrar mensajes al usuario y capturar datos desde la consola. Las funciones cin y cout son esenciales para interactuar con el usuario.

Ejemplo de uso:

cout << "Ingrese el nombre del producto: ";

cin >> nombre;

**2.1.2 <climits>**

La librería <climits> es utilizada para trabajar con los límites de los tipos de datos, especialmente enteros. Se usa en el programa para validar que los valores de identificación (ID) de los productos no excedan los límites definidos por el tipo int.

Ejemplo de uso:

if (id > INT\_MAX) {

std::cout << "ID no válido. Excede el valor máximo permitido.\n"; }

**2.1.3 <iomanip>**

La librería <iomanip> se utiliza principalmente para manipular el formato de la salida en consola. En este programa, se usa para alinear las columnas cuando se imprimen los productos, y para dar formato a las cadenas que contienen los valores de las cantidades.

Ejemplo de uso:

std::cout << std::setw(20) << "Producto" << std::setw(10) << "Cantidad" << "\n";

**2.1.4 <fstream>**

La librería <fstream> permite la lectura y escritura de archivos. Este programa la utiliza para cargar productos desde archivos y para guardar los productos en un archivo, asegurando que los datos persistan entre diferentes ejecuciones del programa.

std::ofstream archivo("productos.txt");

archivo << producto.nombre << "," << producto.id << "," << producto.cantidad << "\n";

**2.1.5 <sstream>**

La librería <sstream> se usa para convertir cadenas de texto en otros tipos de datos y viceversa. Es útil cuando se cargan los datos desde archivos o cuando se necesita procesar los datos en un formato estructurado.

std::stringstream ss;

ss << "100,Producto A,50";

ss >> id >> nombre >> cantidad;

**2.1.6 <string>**

La librería <string> es fundamental para manejar cadenas de texto. En este programa, se usa para almacenar el nombre de los productos y para manipular cadenas cuando se leen o escriben en archivos.

std::string nombre;

std::cout << "Ingrese el nombre del producto: ";

std::cin >> nombre;

**2.1.7 <conio.h>**

La librería <conio.h> se utiliza principalmente para operaciones de control de consola como capturar teclas sin esperar un enter. Esta librería es útil en entornos de consola para manejar entradas rápidas.

getch(); // Captura una tecla y la retorna

**2.1.8 <stdlib.h>**

La librería <stdlib.h> proporciona funciones útiles para la gestión del sistema, como system() para limpiar la pantalla o cambiar configuraciones de la consola. Esto mejora la presentación del programa y la experiencia del usuario.

system("cls"); // Limpia la pantalla de la consola

**2.1.9 <windows.h>**

La librería <windows.h> se usa para realizar manipulaciones más complejas de la consola, como mover el cursor a posiciones específicas para mejorar la disposición visual de la información. Esta función es especialmente útil para crear una presentación más profesional.

gotoxy(10, 5); // Mueve el cursor a la posición (10, 5)

**3. Descripción del Programa**

**3.1 Estructura de Datos**

El programa está basado en una estructura de lista doblemente enlazada. Cada nodo de la lista contiene la siguiente información:

* **Nproducto**: El nombre del producto, que se almacena como una cadena de caracteres (std::string).
* **id**: Un identificador único para cada producto, utilizado para realizar búsquedas y eliminaciones de productos.
* **cantidad**: La cantidad disponible del producto en inventario.
* **sig**: Un puntero al siguiente nodo de la lista.
* **ant**: Un puntero al nodo anterior, lo que permite la navegación en ambas direcciones.

Esta estructura permite operaciones rápidas de inserción y eliminación, ya que no es necesario mover los elementos de la lista como en un arreglo, simplemente se ajustan los punteros.

**3.2 Funciones Principales**

**3.2.1 Insertar Producto**

La función **insertarProducto** permite agregar un nuevo producto a la lista. El usuario ingresa el nombre, el ID y la cantidad del producto. El nuevo nodo se crea dinámicamente y se agrega al final de la lista doblemente enlazada.

**3.2.2 Agregar Nodo**

Esta es una función auxiliar que realiza la inserción del nuevo producto dentro de la lista. Administra el enlace entre los nodos para mantener la integridad de la lista.

**3.2.3 Eliminar Producto**

Esta función permite eliminar un producto específico de la lista. El usuario proporciona el ID del producto que desea eliminar, y el programa recorre la lista hasta encontrar el nodo correspondiente. Si lo encuentra, se elimina ajustando los punteros del nodo anterior y siguiente, evitando la pérdida de información en la lista.

**3.2.4 Imprimir Productos**

La función **imprimirProductos** recorre la lista de productos e imprime en la consola el nombre, ID y cantidad de cada producto. Además, muestra gráficamente la cantidad de cada producto usando asteriscos (\*), lo que proporciona una representación visual intuitiva de la cantidad de productos.

**3.3 Funciones de Ordenamiento**

**3.3.1 Intercambiar Nodos**

La función **intercambiarNodos** se encarga de intercambiar dos nodos de la lista. Es útil durante el proceso de ordenamiento para reorganizar los productos según el criterio de orden elegido (ID o cantidad).

**3.3.2 Partición**

La función **particion** se utiliza en el algoritmo de Quicksort para dividir la lista en dos partes. Los productos con un valor menor al pivote quedan en una parte y los productos con un valor mayor quedan en la otra. Esta función es esencial para que Quicksort divida correctamente la lista en cada iteración.

**3.3.3 Quicksort**

El algoritmo Quicksort es uno de los más eficientes para ordenar listas. En este programa, se utiliza para ordenar los productos por ID o por cantidad. La función **quicksort** implementa la lógica recursiva que divide la lista y luego la reorganiza hasta que esté completamente ordenada.

**3.3.4 Ordenar por ID**

La función **ordenarPorID** usa el algoritmo Quicksort para ordenar los productos según su identificador. El criterio de orden es ascendente, por lo que los productos con IDs más bajos se colocan al principio de la lista.

**3.3.5 Ordenar por Cantidad**

Similar a la ordenación por ID, la función **ordenarPorCantidad** utiliza Quicksort para ordenar los productos según la cantidad disponible. Este algoritmo asegura que los productos con mayor cantidad aparezcan al final de la lista.

**3.4 Funciones de Archivo**

**3.4.1 Guardar Productos en Archivo**

La función **guardarProductos** permite persistir los productos en un archivo. Los datos se guardan en un archivo de texto con cada producto representado en una línea separada por comas. Este archivo puede ser cargado nuevamente en ejecuciones futuras del programa, asegurando la continuidad del inventario.

**3.4.2 Cargar Productos desde Archivo**

La función **cargarProductos** lee los productos almacenados en un archivo y los inserta en la lista. Cada línea del archivo es procesada para extraer el nombre, el ID y la cantidad del producto. Esta operación permite que el programa mantenga el estado del inventario entre sesiones.

**3.5 Funciones Auxiliares**

**3.5.1 Carátula**

La **carátula** es la sección inicial que muestra información sobre el programa, los autores, la asignatura y otros detalles relevantes. Esta información aparece en pantalla cuando el programa se ejecuta.

**3.5.2 gotoxy()**

La función **gotoxy()** mueve el cursor de la consola a una posición específica, lo que permite una presentación más organizada de la salida. En este programa, se utiliza para centrar el texto de la carátula y mejorar la disposición de los productos listados.

**4. Interfaz de Usuario**

**4.1 Menú Principal**

El menú principal es el punto de entrada de la interacción con el programa. Permite al usuario elegir entre las diversas operaciones disponibles:

1. **Agregar Producto**: Ingresar un nuevo producto a la lista.
2. **Eliminar Producto**: Eliminar un producto existente según su ID.
3. **Ordenar Productos por ID**: Organizar los productos según su identificador.
4. **Ordenar Productos por Cantidad**: Ordenar los productos por la cantidad disponible.
5. **Imprimir Lista de Productos**: Mostrar todos los productos en la consola.
6. **Cargar Productos desde Archivo**: Cargar los productos almacenados previamente desde un archivo.
7. **Guardar Productos en Archivo**: Guardar los productos actuales en un archivo.
8. **Salir del Programa**: Finalizar la ejecución del programa.

El usuario selecciona una opción digitando el número correspondiente. El programa luego solicita los datos necesarios y ejecuta la operación seleccionada.

**5. Capturas de Pantalla**

A continuación, se presentan las capturas de pantalla que ilustran el funcionamiento del programa:

* **Captura 1**: Pantalla inicial con el menú principal, donde el usuario puede elegir la opción deseada.

Forma

Descripción generada automáticamente

* **Captura 2**: Pantalla de ingreso de datos para un nuevo producto.

Interfaz de usuario gráfica

Descripción generada automáticamente

* **Captura 3**: Pantalla mostrando los productos ordenados por cantidad con la representación gráfica.

Imagen que contiene Texto

Descripción generada automáticamente

* **Captura 4**: Pantalla de eliminación de un producto, confirmando la acción realizada.

Imagen que contiene Texto

Descripción generada automáticamente

**6. Recomendaciones**

**6.1 Mejora en la Persistencia de Datos**

Una recomendación para mejorar el programa sería la utilización de bases de datos, como SQLite o MySQL, en lugar de archivos de texto para la persistencia de datos. Esto permitiría un manejo más eficiente de grandes volúmenes de datos y una mayor capacidad de consulta.

**6.2 Optimización de la Interfaz de Usuario**

Aunque la interfaz de texto es funcional, sería recomendable desarrollar una interfaz gráfica de usuario (GUI) utilizando herramientas como Qt o wxWidgets. Esto mejoraría la interacción del usuario, haciendo el programa más intuitivo y accesible para personas sin experiencia técnica.

**7. Referencias**

1. **Libros:**
   * "Estructuras de Datos y Algoritmos" por Mark Allen Weiss.
   * "Programación en C++" por Bjarne Stroustrup.
2. **Sitios Web:**
   * [cppreference.com](https://cppreference.com) – Referencia completa de C++.
   * [cplusplus.com](http://cplusplus.com) – Recursos y documentación de C++.
3. **Artículos:**
   * "Introducción al algoritmo Quicksort" en [GeeksforGeeks](https://www.geeksforgeeks.org).

**8. Conclusión**

Este proyecto proporciona una solución robusta para gestionar productos en un inventario usando una lista doblemente enlazada. Al integrar algoritmos eficientes de ordenamiento como Quicksort y funciones de persistencia en archivos, se ofrece un sistema funcional y escalable. La implementación modular permite fácilmente agregar nuevas funcionalidades en el futuro, como la opción de modificar productos o agregar más criterios de ordenamiento.