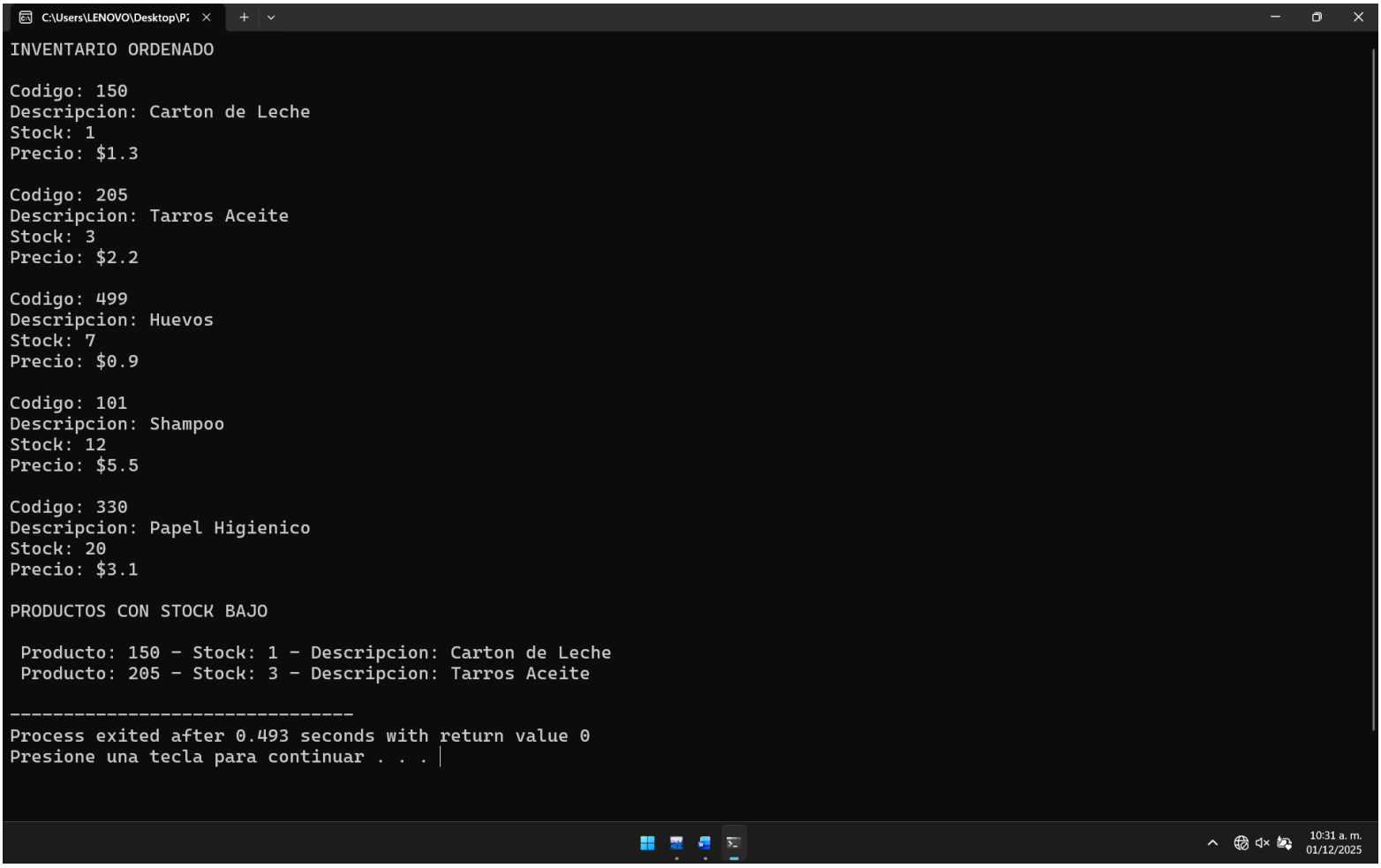
Código

```
#include <iostream>
#include <string>
using namespace std;
typedef int Codigo;
typedef string Descripcion;
typedef int Cantidad;
typedef float Precio;
struct Producto {
    Codigo codigo;
    Descripcion descripcion;
    Cantidad cantidadDisponible;
    Precio precioUnitario;
};
struct NodoSimple {
    Producto dato;
    NodoSimple* siguiente;
};
void insertarSimple(NodoSimple*& cabeza, Producto prod) {
    NodoSimple* nuevo = new NodoSimple();
    nuevo->dato = prod;
    nuevo->siguiente = cabeza;
    cabeza = nuevo;
}
void mostrarInventario(NodoSimple* cabeza) {
    cout << "INVENTARIO ORDENADO\n" << endl;
    while (cabeza != nullptr) {
        cout << "Codigo: " << cabeza->dato.codigo << endl;
        cout << "Descripcion: " << cabeza->dato.descripcion << endl;
        cout << "Stock: " << cabeza->dato.cantidadDisponible << endl;
        cout << "Precio: $" << cabeza->dato.precioUnitario << "\n"<< endl;
        cabeza = cabeza->siguiente;
    }
}
// Algoritmo Selection Sort
void ordenarPorStock(NodoSimple* cabeza) {
    for (NodoSimple* i = cabeza; i != nullptr; i = i->siguiente) {
        NodoSimple* menor = i;
        for (NodoSimple* j = i->siguiente; j != nullptr; j = j->siguiente) {
            if (j->dato.cantidadDisponible < menor->dato.cantidadDisponible) {
                menor = j;
            }
        }
        // Swap
        Producto temp = i->dato;
        i->dato = menor->dato;
        menor->dato = temp;
    }
}
void mostrarAlertas(NodoSimple* cabeza, int limite = 5) {
    cout << "PRODUCTOS CON STOCK BAJO\n" << endl;
    bool hayAlertas = false;
    while (cabeza != nullptr) {
        if (cabeza->dato.cantidadDisponible < limite) {
            cout << " Producto: " << cabeza->dato.codigo << " - Stock: " << cabeza->dato.cantidadDisponible << " - Descripcion:
" << cabeza->dato.descripcion << endl;
            hayAlertas = true;
        }
        cabeza = cabeza->siguiente;
    }
    if (!hayAlertas) {
        cout << "No hay productos con stock bajo" << endl;
    }
}
int main() {
    NodoSimple* lista = nullptr;
    Producto p1 = {101, "Shampoo", 12, 5.50f};
    Producto p2 = {205, "Tarros Aceite", 3, 2.20f};
    Producto p3 = {330, "Papel Higienico", 20, 3.10f};
    Producto p4 = {150, "Carton de Leche", 1, 1.30f};
    Producto p5 = {499, "Huevos", 7, 0.90f};
    insertarSimple(lista, p1);
    insertarSimple(lista, p2);
```

```
    insertarSimple(lista, p3);
    insertarSimple(lista, p4);
    insertarSimple(lista, p5);
    // Ordenar por stock
    ordenarPorStock(lista);
    // Mostrar inventario ordenado
    mostrarInventario(lista);
    // Mostrar alertas de stock bajo
    mostrarAlertas(lista, 5);
    return 0;
}
```

Compilación



Breve Explicación de la solución propuesta

El ejercicio solicita 2 cosas puntuales:

1. Usar una lista.
2. Usar un algoritmo que "*ordene el stock de menos a mayor, recorriendo repetidamente la parte no ordenada del arreglo, buscando el elemento mas pequeño y haciendo un swap con la posición inicial disponible*", esta descripción es la del Algoritmo Selection Sort.

Teniendo claro los dos puntos principales desarrolle el código usando 2 estructuras:

1. Estructura Producto, esta almacena los datos del producto.
2. Estructura NodoSimple, esta sirve para crear la Lista Simple Enlazada.

Se implementa las funciones necesarias:

1. insertarSimple: Solo inserta un producto en la lista.
2. mostrarInventario: Solo muestra muestra los datos de producto desde el nodo.
3. ordenarPorStock: Uso el Algoritmo Selection sort para ordenarlo, y simplemente hago comparaciones extrayendo el dato stock desde el nodo cabeza.
4. mostrarAlertas: Pongo un límite arbitrario de 5 elementos en stock (lo inicializo en el parámetro), ese es el mínimo para considerarse "Bajo Stock", extraigo el dato del nodo cabeza otra vez y lo comparo cantidadDisponible con el límite.
5. Main: En el main simplemente inicializo el NodoSimple llamado lista como nulo y procedo a quemar los datos usando la función insertarSimple, luego solo llamo a el algoritmo de ordenamiento, y los dos métodos para mostrar la información.