Informe sobre el Caso de Búsqueda de Productos en una Librería

1 Introducción

En este informe, se presenta un caso de uso real en el cual se implementa una búsqueda lineal para encontrar productos en una librería, usando como base un sistema informático para gestionar su inventario. En este caso particular, se utiliza un algoritmo de búsqueda lineal para localizar libros específicos dentro de un inventario de productos. El sistema permite que los empleados o clientes puedan consultar si un libro está disponible o no, además de ofrecer detalles sobre el producto (precio, cantidad, ubicación, etc.).

El contexto de la librería es típico en pequeños comercios que gestionan sus productos manualmente y no tienen una base de datos avanzada ni un sistema de búsqueda optimizado. La librería puede contener varios cientos de libros y desea implementar una forma rápida y efectiva de acceder a información clave sobre estos productos.

2 Objetivo

El objetivo de este sistema es permitir la búsqueda eficiente de productos (libros) por su nombre en un catálogo. Al ingresar el nombre del libro a buscar, el sistema debe ser capaz de mostrar si el producto está disponible, junto con detalles importantes como el precio, la cantidad disponible, la ubicación en la tienda, y el estante correspondiente.

3 Caso de Uso: Búsqueda de Productos

En este caso específico, se utiliza un archivo de texto (`producto.txt`) que contiene los datos de todos los productos de la librería, cada uno con los siguientes atributos:

- **Nombre del Producto**: El título del libro.
- **Precio**: El precio del libro.
- **Cantidad**: La cantidad disponible del libro en stock.
- **Ubicación**: La ubicación del libro en la librería.
- **Estante**: El estante específico donde se encuentra el libro.

4 Búsqueda Lineal

El algoritmo de búsqueda lineal es adecuado para este tipo de sistemas sencillos, donde no es necesario ordenar los productos previamente. El algoritmo realiza la búsqueda de manera secuencial, revisando uno por uno todos los elementos de la lista hasta encontrar una coincidencia exacta con el nombre del producto solicitado. Este método es especialmente útil en casos donde la cantidad de productos no es excesivamente grande y donde los productos no están ordenados.

El código que implementa este sistema es el siguiente:

```
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include <algorithm>
#include <ctime>
#include <iomanip>
#include <vector>
#include <cctype>  // Para la funci n tolower()

using namespace std;

// Estructura para representar un producto
struct Producto {
    string nombre;
    string precio;
    string cantidad;
    string ubicacion;
    string estante;
};

// Funci n de comparaci n para ordenar productos por nombre
bool compararProductos(const Producto& p1, const Producto& p2) {
    return p1.nombre < p2.nombre;
}

// Funci n para convertir una cadena a min sculas
string convertirAMinusculas(const string& str) {
    string resultado = str;
    transform(resultado.begin(), resultado.end(), resultado.begin(), ::tolower);
    return resultado;
}

// Funci n de b squeda lineal
int busquedaLineal(const vector<Producto>& productos, const string& datoBuscado) {
    for (int i = 0; i < productos.size(); i++) {
        // Comparamos el nombre en min sculas para hacer la b squeda insensible
    a may sculas
        if (convertirAMinusculas(productos[i].nombre) == datoBuscado) {
            return i;  // Se encontr  el producto
        }
    }
    return -1;  // No encontrado
}

int main() {
    ifstream inData("producto.txt");

    // Verificar si el archivo se abri  correctamente
    if (!inData) {
        cout << "No se pudo abrir el archivo de productos." << endl;
        return 1;
```

```
52      }
53
54      vector<Producto> productos;
55      string linea;
56
57      // Leer el archivo y almacenar los productos
58      while (getline(inData, linea)) {
59          stringstream ss(linea);
60          Producto producto;
61
62          // Leer los datos separados por tabulaciones
63          getline(ss, producto.nombre, '\t');
64          getline(ss, producto.precio, '\t');
65          getline(ss, producto.cantidad, '\t');
66          getline(ss, producto.ubicacion, '\t');
67          getline(ss, producto.estante, '\t');
68
69          productos.push_back(producto);
70      }
71
72      inData.close();
73
74      // Ordenar los productos por nombre
75      sort(productos.begin(), productos.end(), compararProductos);
76
77      // Solicitar al usuario el nombre del producto a buscar
78      string datoBuscado;
79      cout << "Introduce el nombre del producto que deseas buscar: ";
80      getline(cin, datoBuscado);
81
82      // Convertir el nombre ingresado a min sculas para que la b squeda sea
        insensible a may sculas
83      datoBuscado = convertirAMinusculas(datoBuscado);
84
85      // Medir el tiempo de ejecuci n
86      clock_t start = clock();
87
88      // Mostrar los encabezados de la tabla
89      cout << "\n" << left << setw(20) << "Producto"
90           << setw(10) << "Precio"
91           << setw(10) << "Cantidad"
92           << setw(20) << "Ubicaci n"
93           << setw(15) << "Estante" << endl;
94
95      // L nea divisoria
96      cout << string(75, '-') << endl;
97
98      // Realizar la b squeda lineal
99      int index = busquedaLineal(productos, datoBuscado);
100     if (index != -1) {
101         // Si el producto fue encontrado, mostramos su informaci n
102         const Producto& p = productos[index];
103         cout << left << setw(20) << p.nombre
104              << setw(10) << p.precio
105              << setw(10) << p.cantidad
106              << setw(20) << p.ubicacion
107              << setw(15) << p.estante << endl;
108     } else {
109         cout << "No se encontr  el producto: " << datoBuscado << endl;
110     }
111
112     clock_t end = clock();
```

```
113     double duration = double(end - start) / CLOCKS_PER_SEC;
114
115     // Mostrar el tiempo de ejecuci n
116     cout << "Tiempo de ejecuci n de la b squeda: " << duration << " segundos."
        << endl;
117
118     return 0;
119 }
```

5 Ventajas del Sistema

- **Simplicidad**: El sistema es fácil de entender y mantener, con un diseño sencillo que permite la búsqueda de productos de manera clara.
- **Eficiencia para Catálogos Pequeños**: La búsqueda lineal es adecuada cuando el número de productos no es excesivamente grande. Es simple y directa, sin la necesidad de estructuras de datos complejas.
- **Tiempo de Ejecución**: Aunque la búsqueda lineal tiene una complejidad de $\mathcal{O}(n)$, la eficiencia para un número moderado de productos sigue siendo aceptable. El tiempo de ejecución se mide para tener una referencia precisa.