Universidad Nacional del Altiplano
Facultad de Ingeniería Estadística e Informática
Docente: Fred Torres Cruz
Autor : .....

Trabajo Encargado - N° 001
[12pt]article [utf8]inputenc geometry xcolor
a4paper, margin=1in

Código: Búsqueda de Productos

Este programa realiza una búsqueda lineal en un archivo de texto llamado `producto.txt`. Busca cualquier dato relacionado con un producto (nombre, precio, cantidad, etc.) y muestra los resultados coincidentes.

Código fuente:

```
\documentclass[12pt]{article}
\usepackage[utf8]{inputenc}
\usepackage{geometry}
\usepackage{xcolor}

\geometry{a4paper, margin=1in}

\begin{document}

\section*{Código: Búsqueda de Productos}

\textbf{Descripción:} Este programa realiza una búsqueda lineal en un archivo de texto l

\textbf{Código fuente:}

\begin{verbatim}
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include <algorithm>
#include <ctime>
#include <iomanip>

using namespace std;

int main() {
```

```
ifstream inData("producto.txt");

if (!inData) {
    cout << "No se pudo abrir el archivo de productos." << endl;
    return 1;
}

string linea;
string datoBuscado;
int numeroDeLinea = 0;
int pasos = 0;

cout << "Introduce el dato que deseas buscar (nombre de producto, precio, cantidad,
getline(cin, datoBuscado);

bool encontrado = false;
transform(datoBuscado.begin(), datoBuscado.end(), datoBuscado.begin(), ::tolower);

cout << "Algoritmo de búsqueda: Búsqueda Lineal (recorrido de archivo línea por líne

clock_t start = clock();

cout << "\n" << left << setw(20) << "Producto"
     << setw(10) << "Precio"
     << setw(10) << "Cantidad"
     << setw(20) << "Ubicación"
     << setw(15) << "Estante" << endl;

cout << string(75, '-') << endl;

while (getline(inData, linea)) {
    numeroDeLinea++;
    pasos++;

    string lineaLower = linea;
    transform(lineaLower.begin(), lineaLower.end(), lineaLower.begin(), ::tolower);

    if (lineaLower.find(datoBuscado) != string::npos) {
        stringstream ss(linea);
        string producto, precio, cantidad, ubicacion, estante;

        getline(ss, producto, '\t');
        getline(ss, precio, '\t');
        getline(ss, cantidad, '\t');
        getline(ss, ubicacion, '\t');
```

```
            getline(ss, estante, '\t');

            cout << left << setw(20) << producto
                 << setw(10) << precio
                 << setw(10) << cantidad
                 << setw(20) << ubicacion
                 << setw(15) << estante << endl;

            encontrado = true;
        }
    }

    clock_t end = clock();
    double duration = double(end - start) / CLOCKS_PER_SEC;

    cout << "Tiempo de ejecución de la búsqueda: " << duration << " segundos." << endl;
    cout << "Número de pasos (líneas procesadas): " << pasos << endl;

    if (!encontrado) {
        cout << "No se encontró el dato: " << datoBuscado << endl;
    }

    inData.close();

    return 0;
}


\documentclass[12pt]{article}
\usepackage[utf8]{inputenc}
\usepackage{geometry}
\usepackage{xcolor}
\usepackage{listings}

\geometry{a4paper, margin=1in}

% Configuración para el entorno listings
\lstset{
    language=C++,
    basicstyle=\ttfamily\small,
    keywordstyle=\color{blue}\bfseries,
    stringstyle=\color{red},
    commentstyle=\color{gray}\itshape,
    numbers=left,
    numberstyle=\tiny,
```

```
    frame=single,
    breaklines=true,
    tabsize=4,
    showstringspaces=false
}

\begin{document}

\section*{Código: Búsqueda Binaria en Productos}

\textbf{Descripción:} Este programa lee un archivo de productos \texttt{producto.txt}, o

\textbf{Código fuente:}

\begin{lstlisting}[language=C++]
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include <algorithm>
#include <ctime>
#include <iomanip>
#include <vector>

using namespace std;

struct Producto {
    string nombre;
    string precio;
    string cantidad;
    string ubicacion;
    string estante;
};

bool compararProductos(const Producto& p1, const Producto& p2) {
    return p1.nombre < p2.nombre;
}

int busquedaBinaria(const vector<Producto>& productos, const string& datoBuscado) {
    int inicio = 0;
    int fin = productos.size() - 1;

    while (inicio <= fin) {
        int medio = inicio + (fin - inicio) / 2;
        if (productos[medio].nombre == datoBuscado) {
```

```
            return medio;
        }
        if (productos[medio].nombre < datoBuscado) {
            inicio = medio + 1;
        } else {
            fin = medio - 1;
        }
    }
    return -1;  // No encontrado
}

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

    if (!inData) {
        cout << "No se pudo abrir el archivo de productos." << endl;
        return 1;
    }

    vector<Producto> productos;
    string linea;

    while (getline(inData, linea)) {
        stringstream ss(linea);
        Producto producto;

        getline(ss, producto.nombre, '\t');
        getline(ss, producto.precio, '\t');
        getline(ss, producto.cantidad, '\t');
        getline(ss, producto.ubicacion, '\t');
        getline(ss, producto.estante, '\t');

        productos.push_back(producto);
    }

    inData.close();

    sort(productos.begin(), productos.end(), compararProductos);  // Ordenar productos p

    string datoBuscado;
    cout << "Introduce el dato que deseas buscar (nombre del producto): ";
    getline(cin, datoBuscado);

    transform(datoBuscado.begin(), datoBuscado.end(), datoBuscado.begin(), ::tolower);
```

```
    clock_t start = clock();

    cout << "\n" << left << setw(20) << "Producto"
         << setw(10) << "Precio"
         << setw(10) << "Cantidad"
         << setw(20) << "Ubicación"
         << setw(15) << "Estante" << endl;

    cout << string(75, '-') << endl;

    int index = busquedaBinaria(productos, datoBuscado);
    if (index != -1) {
        const Producto& p = productos[index];
        cout << left << setw(20) << p.nombre
             << setw(10) << p.precio
             << setw(10) << p.cantidad
             << setw(20) << p.ubicacion
             << setw(15) << p.estante << endl;
    } else {
        cout << "No se encontró el producto: " << datoBuscado << endl;
    }

    clock_t end = clock();
    double duration = double(end - start) / CLOCKS_PER_SEC;

    cout << "Tiempo de ejecución de la búsqueda: " << duration << " segundos." << endl;

    return 0;
}
\end{lstlisting}

\end{document}
por salto

#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include <algorithm>
#include <ctime>
#include <iomanip>
#include <vector>

using namespace std;
```

```
// Definir la estructura del producto
struct Producto {
    string nombre;
    string precio;
    string cantidad;
    string ubicacion;
    string estante;
};

// Función de comparación para ordenar productos por nombre
bool compararProductosPorNombre(const Producto& p1, const Producto& p2) {
    return p1.nombre < p2.nombre;
}

// Función de búsqueda Fibonacci
int busquedaFibonacci(const vector<Producto>& productos, const string& datoBuscado) {
    int n = productos.size();
    int fibMMm2 = 0;  // (m-2)'th Fibonacci No.
    int fibMMm1 = 1;  // (m-1)'th Fibonacci No.
    int fibM = fibMMm2 + fibMMm1;  // m'th Fibonacci No.

    // Encuentra el Fibonacci más cercano a n
    while (fibM < n) {
        fibMMm2 = fibMMm1;
        fibMMm1 = fibM;
        fibM = fibMMm2 + fibMMm1;
    }

    // Índice de comparación
    int offset = -1;

    // Comienza la comparación
    while (fibM > 1) {
        int i = min(offset + fibMMm2, n - 1);

        // Comparar el dato buscado con el elemento
        if (productos[i].nombre == datoBuscado) {
            return i;
        }

        // Si el dato es mayor, ignora la mitad inferior
        if (productos[i].nombre < datoBuscado) {
            fibM = fibMMm1;
            fibMMm1 = fibMMm2;
            fibMMm2 = fibM - fibMMm1;
```

```
            offset = i;
        }
        // Si el dato es menor, ignora la mitad superior
        else {
            fibM = fibMMm2;
            fibMMm2 = fibMMm1;
            fibMMm1 = fibM - fibMMm2;
        }
    }

    // Compara el último elemento
    if (fibMMm1 && productos[offset + 1].nombre == datoBuscado) {
        return offset + 1;
    }

    return -1;  // No encontrado
}

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

    if (!inData) {
        cout << "No se pudo abrir el archivo de productos." << endl;
        return 1;
    }

    vector<Producto> productos;
    string linea;

    // Leer el archivo y almacenar los productos
    while (getline(inData, linea)) {
        stringstream ss(linea);
        Producto producto;

        getline(ss, producto.nombre, '\t');
        getline(ss, producto.precio, '\t');
        getline(ss, producto.cantidad, '\t');
        getline(ss, producto.ubicacion, '\t');
        getline(ss, producto.estante, '\t');

        productos.push_back(producto);
    }

    inData.close();
```

```
    // Ordenar los productos por nombre
    sort(productos.begin(), productos.end(), compararProductosPorNombre);

    string datoBuscado;
    cout << "Introduce el nombre del producto que deseas buscar: ";
    getline(cin, datoBuscado);

    clock_t start = clock();

    cout << "\n" << left << setw(20) << "Producto"
         << setw(10) << "Precio"
         << setw(10) << "Cantidad"
         << setw(20) << "Ubicación"
         << setw(15) << "Estante" << endl;

    cout << string(75, '-') << endl;

    // Realizar la búsqueda Fibonacci
    int index = busquedaFibonacci(productos, datoBuscado);
    if (index != -1) {
        const Producto& p = productos[index];
        cout << left << setw(20) << p.nombre
             << setw(10) << p.precio
             << setw(10) << p.cantidad
             << setw(20) << p.ubicacion
             << setw(15) << p.estante << endl;
    } else {
        cout << "No se encontró el producto con nombre: " << datoBuscado << endl;
    }

    clock_t end = clock();
    double duration = double(end - start) / CLOCKS_PER_SEC;

    cout << "Tiempo de ejecución de la búsqueda: " << duration << " segundos." << endl;

    return 0;
}
```