Taller 2 – Estructuras lineales

Estructura de datos

Juan Manuel López Vargas

Sebastián Almanza Galvis

David León Velásquez

Juan Sebastián Méndez

Profesor: John Corredor

Fecha: 19 de agosto 2024

Índice:

1. Introducción
2. Objetivo
3. Descripción del problema
4. Diseño del programa
5. Implementación
6. Pruebas y Resultado
7. Conclusión
8. Bibliografía

***Solución del taller***

1. ***Introducción:***

La biblioteca “fstream” en C++ permite el uso de operaciones de lectura y escritura de archivos de texto, a través de un flujo que se conecta al archivo. Se va a diseñar e implementar una aplicación que use estructuras lineales para buscar palabras a partir de subcadenas. Se debe prestar especial atención en el desarrollo y uso de las operaciones de recorrido en secuencias y uso de pilas y colas como apoyo algorítmico. Adicional, el uso de operaciones de lectura de archivos de texto junto con los contenedores de STL.

1. ***Objetivo:***

• Diseñar una aplicación que use estructuras lineales usando contenedores STL

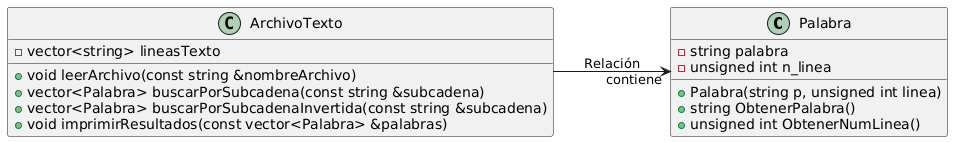
• Elaborar un informe descrito sobre el diseño e implementación.

1. ***Descripción del problema:***

Se deben hacer 3 tipos de búsquedas:

* Buscar palabras que comiencen con una subcadena.
* Buscar palabras que contengan una subcadena en cualquier posición.
* Buscar palabras que contengan la subcadena invertida.

1. ***Diseño del programa:***



TAD’s

TAD Palabra

Datos mínimos:

* palabra, cadena de caracteres, Representa la palabra que se está almacenando.
* n\_linea, entero sin signo, Representa el número de línea en el que se encuentra la palabra en el archivo de texto.

Operaciones:

* ObtenerPalabra(): Retorna la palabra almacenada.
* ObtenerNumLinea(): Retorna el número de línea en el que se encuentra la palabra.
* FijarPalabra(n\_palabra): Cambia la palabra actual a “n\_palabra”.
* FijarNumLinea(n\_num): Cambia el número de línea actual de “n\_num”

TAD ArchivoTexto

Datos mínimos:

* lineasTexto, lista de cadena de caracteres, Representa al archivo de texto contiene de texto que contiene las palabras.

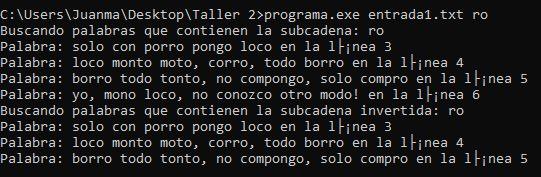
Operaciones:

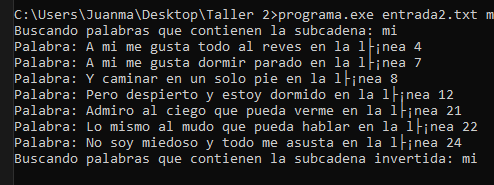
* ObtenerListaLineal(): Retorna la lista actual de líneas de texto en el archivo.
* ObtenerNumLineal(): Retorna la cantidad de líneas de texto en el archivo
* FijarListaLineas(n\_lista): Cambia la lista actual de líneas de texto a “n\_lista”.
* AgregarListaPals(n\_linea): Agrega una nueva línea de texto “n\_linea” a la lista de líneas de texto.
* BuscarPrincipio(subcadena): Busca palabras que comienzan con la subcadena en las líneas de texto.
* BuscarContiene(subcadena): Busca palabras que contienen la subcadena en cualquier posición dentro de las líneas de texto.
* BuscarInvertida(subcadena): Busca palabras que contienen la subcadena invertida en cualquier posición dentro de las líneas de texto.

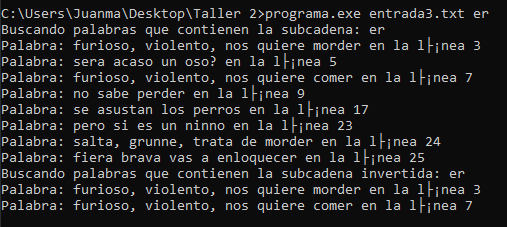
1. ***Implementación:***

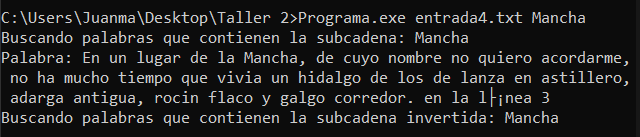
#include <iostream>  
#include <fstream>  
#include <vector>  
#include <string>  
  
using namespace std;  
  
struct Palabra {  
string palabra;  
unsigned int n\_linea;  
  
Palabra(string p, unsigned int linea) : palabra(p), n\_linea(linea) {}  
 };  
  
class ArchivoTexto {  
 private:  
 vector<string> lineasTexto;  
  
 public:  
 void leerArchivo(const string &nombreArchivo) {  
 ifstream archivo(nombreArchivo);  
 if (!archivo.is\_open()) {  
 cerr << "Error al abrir el archivo" << endl;  
 return;  
 }  
  
 string linea;  
 while (getline(archivo, linea)) {  
 lineasTexto.push\_back(linea);  
 }  
  
 archivo.close();  
 }  
  
 vector<Palabra> buscarPorSubcadena(const string &subcadena) {  
 vector<Palabra> resultado;  
 for (unsigned int i = 0; i < lineasTexto.size(); ++i) {  
 if (lineasTexto[i].find(subcadena) != string::npos) {  
 resultado.push\_back(Palabra(lineasTexto[i], i + 1));  
 }  
 }  
 return resultado;  
 }  
  
 vector<Palabra> buscarPorSubcadenaInvertida(const string &subcadena) {  
 string subcadenaInvertida(subcadena.rbegin(), subcadena.rend());  
 return buscarPorSubcadena(subcadenaInvertida);  
 }  
  
 void imprimirResultados(const vector<Palabra> &palabras) {  
 for (const auto &p : palabras) {  
 cout << "Palabra: " << p.palabra << " en la línea " << p.n\_linea << endl;  
 }  
 }  
};  
  
int main(int argc, char\* argv[]) {  
 if (argc < 3) {  
 cerr << "Uso: " << argv[0] << " <archivo> <subcadena>" << endl;  
 return 1;  
 }  
  
 string nombreArchivo = argv[1];  
 string subcadena = argv[2];  
  
 ArchivoTexto archivoTexto;  
 archivoTexto.leerArchivo(nombreArchivo);  
  
 cout << "Buscando palabras que contienen la subcadena: " << subcadena << endl;  
 vector<Palabra> palabras = archivoTexto.buscarPorSubcadena(subcadena);  
 archivoTexto.imprimirResultados(palabras);  
  
 cout << "Buscando palabras que contienen la subcadena invertida: " << subcadena << endl;  
 palabras = archivoTexto.buscarPorSubcadenaInvertida(subcadena);  
 archivoTexto.imprimirResultados(palabras);  
  
 return 0;  
}

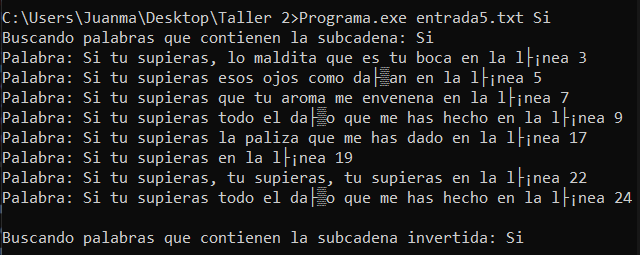
1. Pruebas y Resultado:

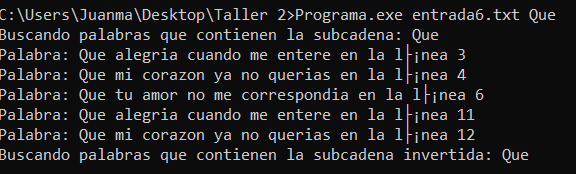


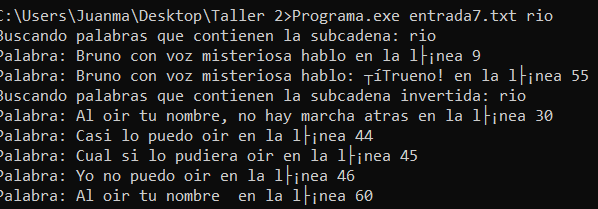


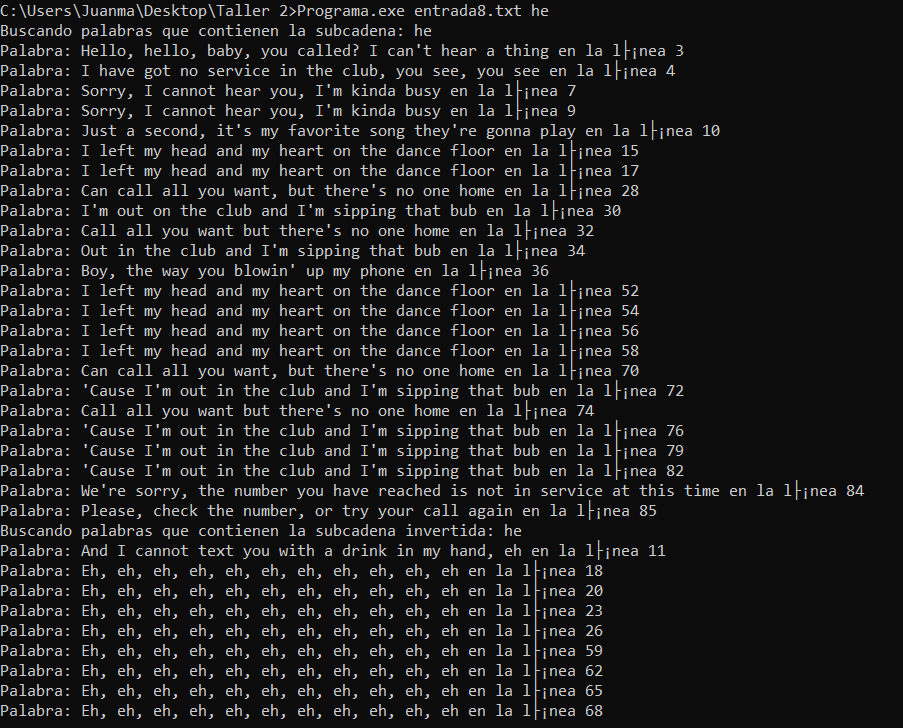












1. Conclusión:

El programa ha sido probado con diferentes archivos de texto y subcadenas en diferentes condiciones. Los resultados indican que el programa ejecuta sus operaciones esenciales, que incluyen examinar archivos, identificar subcadenas en texto y detectar subcadenas invertidas. Los casos de prueba han demostrado que el programa detecta y enumera correctamente las líneas de la subcadena dada, así como la subcadena misma (invertida) y ha funcionado. En consecuencia, se puede determinar que el programa está funcionando según lo previsto.