

## **Taller 2**

**Douglas Santiago Diaz**

## Introducción

Este informe documenta el desarrollo e implementación de un programa en C++ que permite buscar palabras dentro de un archivo de texto, utilizando una subcadena ingresada por el usuario. Se explican los Tipos Abstractos de Datos (TADs) utilizados, la estructura del código y los resultados obtenidos con diferentes archivos de prueba.

El objetivo del programa es procesar un archivo de texto y realizar búsquedas de palabras que cumplan con tres condiciones específicas:

- Comienzan con la subcadena.
- Contienen la subcadena en cualquier posición.
- Contienen la versión invertida de la subcadena.

## Funcionamiento del código

El código está estructurado en varios archivos que implementan clases para manejar la lógica de búsqueda. A continuación, se describe su funcionamiento:

1. **Archivo (archivo.h y archivo.cxx)**
  - Se encarga de leer el archivo de entrada y almacenar su contenido.
  - Implementa funciones para buscar palabras que comiencen con la subcadena y que la contengan en cualquier posición.
2. **Palabra (palabra.h y palabra.cxx)**
  - Define la estructura de una palabra y almacena su número de línea.
3. **Búsqueda (busqueda.h y busqueda.cxx)**
  - Utiliza una cola para almacenar los resultados en orden de aparición.
  - Utiliza una pila para almacenar palabras con la subcadena invertida.
  - Implementa métodos para realizar las búsquedas y mostrar los resultados.
4. **Archivo Principal (main.cpp)**
  - Lee el archivo de entrada y obtiene la subcadena de búsqueda.
  - Llama a las funciones de búsqueda y muestra los resultados en la consola.

## Repositorio en GitHub

El código fuente de este proyecto está disponible en el siguiente repositorio de GitHub:

<https://github.com/SoyDouglas/BuscarSubcadenas>

## Especificación del problema

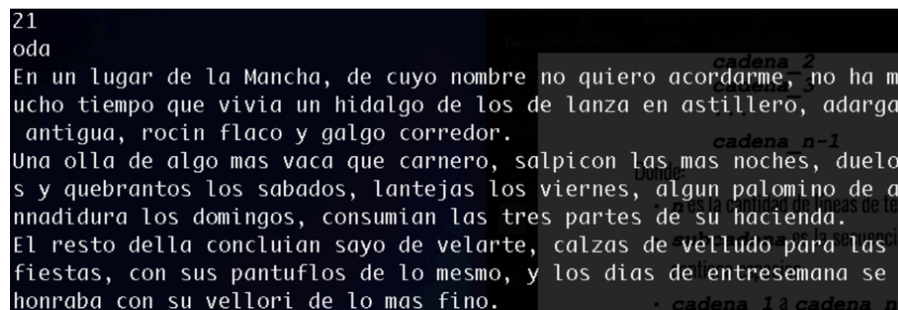
Se requiere diseñar e implementar un programa que busque subcadenas dentro de un archivo de entrada proporcionado por el usuario. Este archivo tiene un formato específico:

- La primera línea contiene un número entero **n**, que indica la cantidad de líneas de texto en el archivo (sin contar esta línea).
- La segunda línea contiene la **subcadena** a buscar.
- De la tercera línea en adelante, se encuentran **n** cadenas de caracteres que representan el texto en el que se realizarán las búsquedas.

Estructura:

```
n
subcadena (a buscar)
cadena_1
cadena_2
cadena_3
...
cadena_n-1
```

Ejemplo:



```
21
oda
En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha m
ucho tiempo que vivia un hidalgo de los de lanza en astillero, adarga
antigua, rocin flaco y galgo corredor.
Una olla de algo mas vaca que carnero, salpicon las mas noches, duelo
s y quebrantos los sabados, lantejas los viernes, algun palomino de a
nnadidura los domingos, consumian las tres partes de su hacienda.
El resto della concluian sayo de velarte, calzas de velludo para las
fiestas, con sus pantuflos de lo mismo, y los dias de entresemana se
honraba con su vellori de lo mas fino.
```

El programa debe procesar este archivo y proporcionar la cantidad y lista de palabras que cumplen con cada uno de los tres criterios de búsqueda.

---

## Diseño TADs

### TAD Palabra

#### Datos mínimos

- **palabra**: cadena de caracteres. Representa la palabra a almacenar.
- **n\_linea**: entero que representa el número de línea en el que se encuentra la palabra.

#### Operaciones

- **FijarPalabra(n\_palabra)**: cambia la palabra actual a n\_palabra.
  - **FijarNumLinea(n\_num)**: cambia el número de línea actual a n\_num.
  - **ObtenerPalabra()**: retorna la palabra almacenada.
  - **ObtenerNumLinea()**: retorna el número de línea actual de la palabra.
- 

### TAD Archivo

#### Datos mínimos

- **lineasTexto**: vector de vector de cadenas de caracteres. Representa el archivo de texto con líneas de texto que contienen palabras.
- **subcadena**: cadena de caracteres. Representa la subcadena que se utilizará en la búsqueda.

#### Operaciones

- **FijarListaLineas(n\_lista)**: cambia el vector actual de líneas de texto por n\_lista.
- **ObtenerListaLineas()**: retorna el vector de líneas de texto que contiene el archivo.
- **ObtenerNumLineas()**: retorna la cantidad de líneas de texto en el archivo.
- **AgregarListaPals(n\_lista)**: agrega una nueva línea de texto dada al vector que las contiene.
- **BuscarPrincipio(subcadena)**: busca la subcadena al principio de cada palabra de las líneas de texto del archivo.
- **BuscarContiene(subcadena)**: busca la subcadena dentro de cada palabra de las líneas de texto del archivo.
- **LeerArchivo(nombreArchivo)**: lee un archivo de texto y almacena su contenido en lineasTexto.

- **ObtenerSubcadena():** retorna la subcadena almacenada en el archivo.
- 

## TAD Búsqueda

### Datos mínimos

- **resultadosCola:** cola que almacena las palabras encontradas y su número de línea en orden de aparición.
- **resultadosPila:** pila que almacena las palabras encontradas con la subcadena invertida.

### Operaciones

- **BuscarPorPrefijo(archivo, subcadena):** busca las palabras que comienzan con la subcadena en el archivo.
- **BuscarPorSubcadena(archivo, subcadena):** busca las palabras que contienen la subcadena en cualquier posición.
- **BuscarPorSubcadenaInvertida(archivo, subcadena):** busca las palabras que contienen la subcadena invertida.
- **ImprimirResultadosCola():** imprime los resultados almacenados en la cola.
- **ImprimirResultadosPila():** imprime los resultados almacenados en la pila.

## Forma de compilar y ejecutar

Para ejecutar el programa usar el comando:

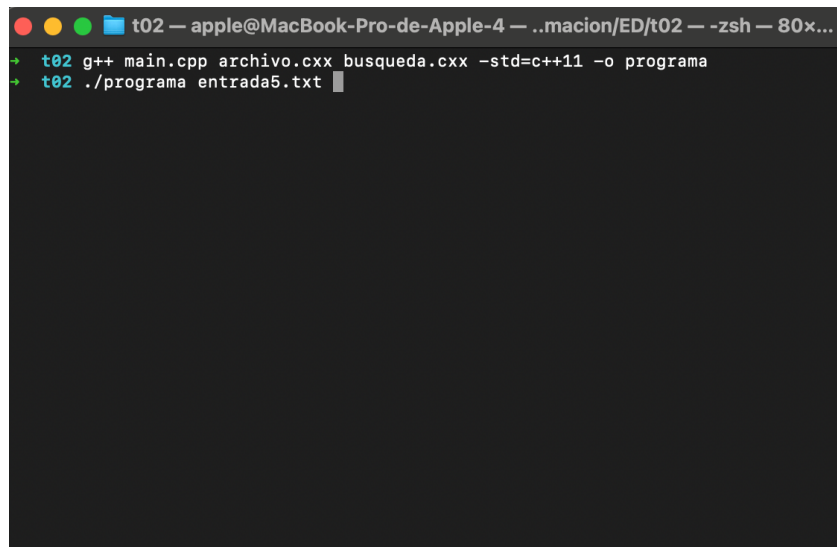
### Compilación:

```
g++ main.cpp archivo.cxx busqueda.cxx -std=c++11 -o programa
```

### Ejecución:

```
./programa entrada#.txt
```

### Ejemplo de compilación y ejecución

A screenshot of a macOS terminal window. The title bar shows the window name 't02' and the user 'apple' on a 'MacBook-Pro-de-Apple-4' machine, with the current directory being '..macion/ED/t02'. The terminal has a dark background with light-colored text. Two commands are entered, each preceded by a green prompt character and the text 't02'. The first command is 'g++ main.cpp archivo.cxx busqueda.cxx -std=c++11 -o programa'. The second command is './programa entrada5.txt', followed by a cursor. The rest of the terminal window is empty.

```
t02 — apple@MacBook-Pro-de-Apple-4 — ..macion/ED/t02 — -zsh — 80x...
+ t02 g++ main.cpp archivo.cxx busqueda.cxx -std=c++11 -o programa
+ t02 ./programa entrada5.txt
```

## Ejemplo de salida

Si el archivo4.txt contiene:

```
21
oda
En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivia un
hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor.
Una olla de algo mas vaca que carnero, salpicon las mas noches, duelos y quebrantos los sabados,
lantejas los viernes, algun palomino de añadidura los domingos, consumian las tres partes de su
hacienda.
El resto della concluian sayo de velarte, calzas de velludo para las fiestas, con sus pantuflos
de lo mismo, y los dias de entresemana se honraba con su vellori de lo mas fino.
Tenia en su casa una ama que pasaba de los cuarenta y una sobrina que no llegaba a los veinte, y
un mozo de campo y plaza que así ensillaba el rocín como tomaba la podadera.
Frisaba la edad de nuestro hidalgo con los cincuenta años.
Era de complexion recia, seco de carnes, enjuto de rostro, gran madrugador y amigo de la caza.
Quieren decir que tenia el sobrenombre de Quijada, o Quesada, que en esto hay alguna diferencia
en los autores que deste caso escriben, aunque por conjeturas verisimiles se deja entender que
se llamaba Quijana.
Pero esto importa poco a nuestro cuento: basta que en la narracion del no se salga un punto de
la verdad.
Es, pues, de saber que este sobredicho hidalgo, los ratos que estaba ocioso, que eran los mas
del año, se daba a leer libros de caballerias, con tanta aficion y gusto, que olvido casi de
todo punto el ejercicio de la caza y aun la administracion de su hacienda; y luego a tanto su
curiosidad y desatino en esto, que vendio muchas hanegas de tierra de sembradura para comprar
libros de caballerias en que leer, y, así, llevo a su casa todos cuantos pudo haber dellos; y,
de todos, ningunos le parecian tan bien como los que compuso el famoso Feliciano de Silva,
porque la claridad de su prosa y aquellas entricadas razones suyas le parecian de perlas, y mas
cuando llegaba a leer aquellos requiebros y cartas de desafios, donde en muchas partes hallaba
escrito: La razon de la sinrazon que a mi razon se hace, de tal manera mi razon enflaquece, que
con razon me quejo de la vuestra fermosura.
Y tambien cuando leia: Los altos cielos que de vuestra divinidad divinamente con las estrellas
os fortifican y os hacen merecedora del merecimiento que merece la vuestra grandeza.
```

La salida es:

```
t02 — apple@MacBook-Pro-de-Apple-4 — ../Taller 2/t02 — zsh — 80x21
➤ t02 ./programa entrada4.txt
Subcadena de búsqueda: oda

Hay 0 palabras encontradas.
palabras que comiencen con: oda

Hay 3 palabras encontradas.
Linea 4: podadera.
Linea 13: acomodada
Linea 15: toda
palabras que contienen: oda

Hay 6 palabras encontradas.
Linea 18: criado.
Linea 17: encantado,
Linea 13: graduado
Linea 12: curado,
Linea 6: madrugador
Linea 2: sabados,
palabras que contienen la subcadena invertida: ado
➤ t02
```



## Conclusión

El desarrollo de este programa permitió aplicar estructuras lineales como listas, colas y pilas en la solución de un problema de búsqueda en texto. La implementación se realizó siguiendo un diseño modular utilizando clases para los TADs *Palabra*, *Archivo* y *Busqueda*, lo que permitió una organización clara del código.

La aplicación desarrollada demuestra la importancia del uso de estructuras de datos eficientes en el procesamiento de grandes volúmenes de texto. Además, el enfoque basado en colas y pilas ofrece una forma estructurada de almacenar y manipular los resultados de búsqueda.

Este proyecto refuerza la comprensión de estructuras lineales y su aplicabilidad en problemas prácticos de búsqueda y procesamiento de datos en archivos de texto. Asimismo, brinda una base sólida para la optimización y escalabilidad del programa en aplicaciones futuras.