

1 Objetivo

Diseñar e implementar una aplicación que usa estructuras lineales para buscar palabras a partir de subcadenas. En especial, se desea evaluar las habilidades del estudiante en el desarrollo y uso de las operaciones de recorrido en secuencias, y uso de pilas y colas como apoyo algorítmico; así como el uso de operaciones de lectura de archivos de texto y de los contenedores proveídos por la STL.

2 Recordatorio: compilación con g++

La compilación con g++ (compilador estándar que será usado en este curso para evaluar y calificar las entregas) se realiza con los siguientes pasos:

1. **Compilación:** de todo el código fuente compilable (**ÚNICAMENTE LOS ARCHIVOS CON EXTENSIONES** *.c, *.cpp, *.cxx)
`g++ -std=c++11 -c *.c *.cxx *.cpp`
2. **Encadenamiento:** de todo el código de bajo nivel en el archivo ejecutable
`g++ -std=c++11 -o nombre_de_mi_programa *.o`

Nota: Estos dos pasos (compilación y encadenamiento) pueden abreviarse en un sólo comando:

```
g++ -std=c++11 -o nombre_de_mi_programa *.c *.cxx *.cpp
```

3. **Ejecución:** del programa ejecutable anteriormente generado
`./nombre_de_mi_programa`

ATENCIÓN: Los archivos de encabezados (*.h, *.hpp, *.hxx) **NO SE COMPILAN**, se incluyen en otros archivos (encabezados o código). Así mismo, los archivos de código fuente (*.c, *.cpp, *.cxx) **NO SE INCLUYEN**, se compilan. Si el programa entregado como respuesta a este Taller no atiende estas recomendaciones, automáticamente se calificará la entrega sobre un 25% menos de la calificación máxima.

2.1 Recordatorio: lectura de archivos de texto

La librería `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. A continuación se presenta un ejemplo de lectura de un archivo por líneas (tomado de <http://www.cplusplus.com/doc/tutorial/files/>):

```
// reading a text file
#include <iostream>
#include <fstream>
#include <string>
int main () {
    std::string line;
    std::ifstream myfile ("example.txt");
    if (myfile.is_open()) {
        while ( getline (myfile,line) ) {
            std::cout << line << '\n';
        }
        myfile.close();
    }
    else std::cout << "Unable to open file";
    return 0;
}
```

3 Desarrollo del taller

El objetivo de este taller es diseñar (utilizando la plantilla de diseño de TADs vista en clase) e implementar un programa que permita buscar palabras dentro de un archivo de texto utilizando subcadenas (palabras más cortas contenidas dentro de las palabras buscadas). En particular, para una subcadena dada, el programa debe:

- imprimir en pantalla la cantidad de palabras que comienzan por la subcadena, y listar cada palabra encontrada indicando su número de línea en el texto.
- imprimir en pantalla la cantidad de palabras que contienen la subcadena. La subcadena puede aparecer en cualquier posición de la palabra (no sólo al principio de la palabra), y debe estar contenida completamente dentro de la palabra (no pueden sobrar letras de la subcadena en la palabra). Se debe listar cada palabra encontrada indicando su número de línea en el texto.
- imprimir en pantalla la cantidad de palabras que contienen la subcadena invertida (leída de derecha a izquierda). La subcadena invertida puede aparecer en cualquier posición de la palabra (no sólo al principio o al final de la palabra), y debe estar contenida completamente dentro de la palabra (no pueden sobrar letras de la subcadena en la palabra). Se debe listar cada palabra indicando su número de línea en el texto.

Nota: El diseño del programa debe considerar el uso de al menos dos tipos diferentes de estructuras lineales (listas, pilas o colas). La implementación debe utilizar los contenedores correspondientes de la STL (Standard Template Library) de C++.

Como entrada, el programa debe recibir por línea de comandos el nombre del archivo que contiene la información necesaria. Esto quiere decir que, una vez compilado el programa, debe ejecutarse por la línea de comandos de la siguiente forma:

```
./nombre_de_mi_programa archivo.txt
```

La estructura del archivo de entrada será la siguiente:

```
n
subcadena
cadena_1
cadena_2
cadena_3
...
cadena_n-1
```

Donde:

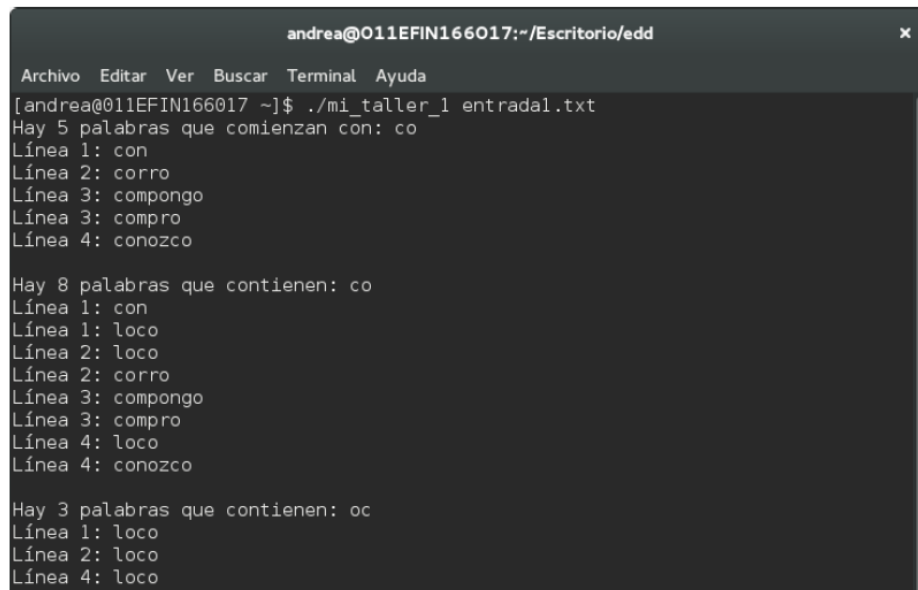
- *n* es la cantidad de líneas de texto que contiene el archivo (sin contar la línea donde aparece este valor).
- *subcadena* es la secuencia de letras que se utilizará para buscar las palabras requeridas. Se garantiza que esta subcadena no contiene espacios.
- *cadena_1* a *cadena_n-1* son cadenas de caracteres (una por línea) que representan el texto donde deben buscarse las palabras. Estas cadenas de caracteres pueden ser desde palabras individuales hasta frases completas de longitud indeterminada.

3.1 Ejemplo

Para el archivo de entrada `entrada1.txt`, que contiene la siguiente información:

```
5
co
solo con porro pongo loco
loco monto moto, corro, todo borro
borro todo tonto, no compongo, solo compro
yo, mono loco, no conozco otro modo!
```

La salida esperada del programa se presenta en la siguiente imagen:



```
andrea@O11EFIN166017:~/Escritorio/edd
Archivo Editar Ver Buscar Terminal Ayuda
[andrea@O11EFIN166017 ~]$ ./mi_taller_1 entradal.txt
Hay 5 palabras que comienzan con: co
Línea 1: con
Línea 2: corro
Línea 3: compongo
Línea 3: compro
Línea 4: conozco

Hay 8 palabras que contienen: co
Línea 1: con
Línea 1: loco
Línea 2: loco
Línea 2: corro
Línea 3: compongo
Línea 3: compro
Línea 4: loco
Línea 4: conozco

Hay 3 palabras que contienen: oc
Línea 1: loco
Línea 2: loco
Línea 4: loco
```

4 Evaluación

La entrega se hará a través de la correspondiente asignación de BrightSpace, antes de la medianoche del martes 15 de agosto. Se debe entregar un único archivo comprimido (único formato aceptado: .zip), nombrado con los apellidos de los integrantes del grupo. Este comprimido debe contener, dentro de un mismo directorio (sin estructura de carpetas interna), el documento de diseño (.pdf) y el código fuente (.h, .hxx, .cxx, .cpp). Si la entrega contiene archivos en cualquier otro formato, será descartada y no será evaluada, es decir, la nota definitiva de la entrega será de 0 (cero) sobre 5 (cinco).

La evaluación del taller tendrá la siguiente escala:

- **Excelente (5.0/5.0):** El estudiante diseñó correctamente (siguiendo la plantilla) e implementó una solución que ofrece las 3 funcionalidades (permite buscar la subcadena al principio de la palabra, en cualquier posición de la palabra y de forma invertida).
- **Muy bueno (4.2/5.0):** El estudiante diseñó correctamente (siguiendo la plantilla) e implementó una solución que ofrece 2 de las 3 funcionalidades requeridas.
- **Bueno (3.5/5.0):** El estudiante diseñó correctamente (siguiendo la plantilla) e implementó una solución que ofrece sólo 1 de las 3 funcionalidades requeridas.
- **No fue un trabajo formal de ingeniería (3.0/5.0):** El estudiante implementó una solución completa o parcial, pero no la diseñó correcta o completamente.
- **Necesita mejoras sustanciales (2.0/5.0):** El estudiante diseñó y/o implementó una solución, pero no es completa o no soluciona lo pedido.
- **Malo (1.0/5.0):** El código entregado por el estudiante no compila en el compilador g++ (mínimo versión número 4.5).
- **No entregó (0.0/5.0).**

IMPORTANTE: Por "diseño de la solución" se entiende la especificación de los TAD en el formato (plantilla) visto en clase.