Estructuras de Datos

Taller 02

Juan Esteban Bello Durango

Instrucciones de uso

Una vez ubicados en el directorio en el que se encuentra el código fuente y las diferentes entradas de texto, use el comando (g++ -o nombre_del_programa taller02.cpp).

Una vez compilado, ejecute el programa con el siguiente comando (./nombre_del_programa entrada_de_texto.txt)

note que “nombre_del_programa” puede ser cambiado por cualquier nombre de su preferencia siempre y cuando sea el mismo en ambos comandos. Por otro lado “entrada_de_texto.txt” debe ser reemplazado por el nombre de uno de los archivos de texto existentes en el directorio o uno personalizado siempre y cuando siga la siguiente estructura:

n (correspondiente a la cantidad de líneas de texto en el archivo)

subcadena (es la secuencia de letras sin espacios que se utilizará para buscar las palabras requeridas)

cadena_1

cadena_2

…

cadena_n

Informe del programa

Lectura de ficheros

En este programa la lectura de ficheros se realiza por medio de la librería fstream utilizado el método ifstream que recibe como argumento de entrada el nombre del archivo indicado por consola

```
ifstream archivo(argv[1]);                    //abre el archivo indicado por el segundo argumento
```

Separación de string en palabras

Para esta función se utilizó un stack como estructura de apoyo en la que se van almacenando cada una de las palabras de la línea separadas por espacios, luego retorna este stack de strings para que el resto del programa pueda procesarlas individualmente

```
//funcion que separa las lineas del archivo por palabra y las almacena en un stack
stack<string> separaPalab(string linea){
        stack<string> palabras;
        size_t ini = 0, fin;                                    //variables de posicion inicial y final

        while((fin = linea.find(' ',ini)) != string::npos){
                palabras.push(linea.substr(ini, fin-ini));
                ini = fin + 1;
        }
        palabras.push(linea.substr(ini, fin-ini));
        return palabras;
}
```

Reversión de la sub-cadena

Para esta función también se utilizo un stack pues como solo permite leer el ultimo elemento ingresado simplemente se lleno este stack y luego se volvió a llenar el string por lo que queda invertido

```
//funcion que revierte un string
string reverso(string s){
        stack<char> st;

        for(char c : s)
                st.push(c);

        s.clear();

        while(!st.empty()){
                s.push_back(st.top());
                st.pop();
        }

        return s;
}
```

El funcionamiento de la parte de búsqueda se apoya en el uso de tres listas (resini, rescon, y resrev) que almacenaran los resultados de búsqueda correspondientes a las tres funciones solicitadas. Consiste en un primer bucle while que hace que las siguientes operaciones se repitan por cada una de las líneas del archivo a excepción de las primeras dos que definen los parámetros de la cantidad de líneas y la sub-cadena que se desea buscar

Dentro de este primer bucle se encuentra un segundo bucle while que recorre cada uno de los strings en el stack de palabras y busca si la sub-cadena se encuentra en alguna parte de la palabra. Si encuentra que coincide al inicio de la palabra lo almacena en la lista resini junto con el número de línea. Si encuentra que coincide en otra parte que no sea el inicio lo almacena en la lista rescon junto con el número de línea. Si encuentra que coincide con la subcadena invertida lo almacena en la lista resrev junto con el número de línea. Por último, si no coincide de ninguna manera con la sub-cadena simplemente se desecha (usando el método pop() del stack de palabras) y se evalúa la siguiente palabra hasta que el stack este vacío

```
//determina si las palabras almacenadas en el stack inician, contienen o contienen en reverso la subcadena
while(!palabras.empty()){
	//determima si inicia o contiene la sub cadena
	int res = palabras.top().find(sub);
	if(res!=string::npos){
		if(res == 0){
			resini.push_back("Linea " + to_string(n) + ": " + palabras.top());	//guarda linea y palabra en resini
		}else
			rescon.push_back("Linea " + to_string(n) + ": " + palabras.top());	//guarda linea y palabra en rescon
	}

	//determina si contiene la subcadena invertida
	res = palabras.top().find(reverso(sub));
	if(res!=string::npos){
		resrev.push_back("Linea " + to_string(n) + ": " + palabras.top());		//guarda linea y palabra en resrev
	}

	palabras.pop();

}
n++;					//aumeta el contador de linea
```

Una vez se han recorrido todas las líneas del archivo se usan bucles for para recorrer y mostrar por terminal cada una de las listas con los resultados de búsqueda del proceso anterior

```
//imprime el contenido de las listas
cout<<"se encontraron "<<resini.size()<<" palabras que inician por "<<sub<<endl;
for(string s : resini)
	cout<<s<<endl;
cout<<"\nse encontraron "<<rescon.size()<<" palabras que contienen "<<sub<<endl;
for(string s : rescon)
	cout<<s<<endl;
cout<<"\nse encontraron "<<resrev.size()<<" palabras que contienen "<<reverso(sub)<<endl;
for(string s : resrev)
	cout<<s<<endl;
```

Resultados

Después de compilar el programa utilizando el comando (g++ -o t02 taller02.cpp) , y ejecutarlo con el argumento entrada4.txt de la siguiente forma : (./t02 entrada4.txt), podemos observar un resultado coherente con el esperado en el ejemplo de la guía

```
vboxuser@Ubuntu:~/Estructuras_D/taller02$ g++ -o t02 taller02.cpp
vboxuser@Ubuntu:~/Estructuras_D/taller02$ ./t02 entrada4.txt
se encontraron 0 palabras que inician por oda

se encontraron 3 palabras que contienen oda
Linea 4: podadera.
Linea 13: acomodada
Linea 15: toda

se encontraron 6 palabras que contienen ado
Linea 2: sabados,
Linea 6: madrugador
Linea 12: curado,
Linea 13: graduado
Linea 17: encantado,
Linea 18: criado.
vboxuser@Ubuntu:~/Estructuras_D/taller02$
```