Práctica 4: STL e Iteradores

Dpto. Ciencias de la Computación e Inteligencia Artificial
E.T.S. de Ingenierías Informática y de Telecomunicación
Universidad de Granada

DECSAI

Estructuras de Datos

Grado en Ingeniería Informática C

Índice de contenido

1. Introducción

Los objetivos de este guión de prácticas son los siguientes:

1. Practicar con T.D.A implementados en la STL
2. Resolver un problema eligiendo la mejor estructura de datos para las operaciones que se solicitan
3. Seguir asimilando los conceptos de documentación de un tipo de dato abstracto (T.D.A)

Los requisitos para poder realizar esta práctica son:

1. Haber estudiado el Tema 1: Introducción a la eficiencia de los algoritmos
2. Haber estudiado el Tema 2: Abstracción de datos. Templates.
3. Haber estudiado el Tema 3: T.D.A. Lineales.
4. Haber estudiado el Tema 4: STL e Iteradores.
5. Aunque no es un requisito indispensable, haber realizado la práctica 2 ayudará a entender mejor el problema.

2. STL

EL objetivo en esta práctica es hacer uso de la librería STL en C++ http://www.cplusplus.com/reference/stl/. Con tal objetivo vamos a centrarnos en resolver un problema con dicha librería. Previo al uso de esta librería, como en la anterior práctica se requiere que el alumno diseñe nuevos T.D.A eficientes para resolver el problema.

3. Ejercicio

En una academia de lenguajes queremos obtener una aplicación que nos permita traducir un refrán o dicho (o frase) de un determinado idioma a otro en un idioma destino. Es decir un refrán de un idioma puede tener varias traducciones. Con tal fin vamos a desarrollar varios tipos de datos abstractos:

- Frase: se compone de una frase en el idioma origen y se transforma en una o más frases en el idioma destino. Las frases no contendrán acentos, tildes ni el carácter ñ. Si existe más de un vocablo se separa por un espacio.
- Traductor: Es un conjunto de frases ordenadas por la frase en el idioma origen.

Se pide desarrollar el T.D.A. Frase definido en la práctica 2. Para el T.D.A Frase se debe dar una representación usando los contenedores de la STL. Por ejemplo se recomienda al alumno valorar la posibilidades de usar ***pair, vector, list, set*** o ***multiset.***

Para el T.D.A Traductor se le recomienda a alumno que analice el uso de los contenedor ***map*** *o* ***multimap*** de la STL. Puede que sea necesario usar incluso más de un contenedor para acelerar las operaciones que se quieren implementar.

Será indispensable que el T.D.A Traductor tenga definido un ***iterador*** para poder iterar sobre las frases del traductor. Y sobre el T.D.A Frase también definiremos un ***iterador*** que nos permita recorrer las frases en el idioma destino asociada a una frase en el idioma origen.

3.1. Traductor Inverso.

El objetivo es construir un programa que dado un traductor obtenga el traductor inverso.

Así por ejemplo si la entrada es el traductor de español-inglés, este programa obtendrá el traductor de inglés-español.

El comando podrá ejecutarse de dos formas. Una dando un único parámetro

```
prompt%  traductor_inverso spanish_english
```

En esta ejecución el programa recibe el traductor origen y su objetivo es obtener el traductor inverso (en este ejemplo english_spanish). La salida se hará en la salida estándar.

Si el programa se ejecuta dándole un parámetro más, este será el nombre del fichero donde dejaremos el traductor inverso.

```
prompt%  traductor_inverso spanish_english english_spanish
```

El T.D.A Traductor propuesto tiene que servir para poderlo usar con el siguiente código :

```
1.  #include "traductor.h"
2.  #include <fstream>
3.  #include <iostream>
4.  using namespace std;
5.  void GetTraductorInverso(const Traductor & t_origen, Traductor & t_destino){
6.
7.  }
8.  void ImprimeTraductor(const Traductor &T,ostream &os){
9.    Traductor::const_iterator it;
10.   for (it=T.begin(); it!=T.end();++it){
11.     os<<(*it).first<<";";
12.     vector<string>::const_iterator it_d;
13.     for (it_d=(*it).second.begin();
14.          it_d!=(*it).second.end();  ++it_d)
15.               os<<(*it_d)<<";";
16.  }
17. }
18. int main(int argc, char * argv[]){
19.  if (argc!=2 || argc!=3){
20.     cout<<".-Dime el nombre de fichero
21.              del traductor origen"<<endl;
22.    cout<<".-[opcionalmente] El nombre
23.              de fichero del    traductor destino"<<endl;
24.     return 0;
25.  }
26.  ifstream f (argv[1]);
27.  if (!f){
28.      cout<<"No puedo abrir el fichero "
29.     <<argv[1]<<endl;    return 0;
30.  }
31.  Traductor t_ori;
32.  f>>t_ori; //Cargamos en memoria, en el traductor.
33.
34.  Traductor t_des;
35.   GetTraductorInverso(t_ori,t_des);
36.   if (argc==2)
37.    ImprimeTraductor(t_des,cout);
38.  else{
39.     ofstream fout(argv[2]);
40.     if (!fout){
41.    cout<<"No puedo crear el fichero "
42.              <<argv[2]<<endl;
43.     return 0;
44.    }
45.    ImprimeTraductor(t_des,fout);
46.  }
    }
```

3.2. Concatenación de Traductores

Dados dos traductores, cada uno con su idioma origen e idioma destino, el objetivo de este programa es obtener el traductor que surge del idioma origen del primer traductor al idioma destino del segundo traductor. Así por ejemplo si los dos traductores de partida son ***frases_ingles_español y frases_español_frances*** al concatenar obtendremos el traductor de ***frases_ingles_frances***

El comando podrá ejecutarse desde la línea de órdenes de la siguiente manera:

```
prompt%  concatenar frases_ingles_espanhol.txt frases_español_frances.txt
frases_ingles_frances.txt
```

Los parámetros son los siguientes:

1. El nombre del fichero con el primer traductores a concatenar
2. El nombre del fichero con el segundo traductor a concatenar
3. El nombre del fichero del traductor salida.

3.3. Sumar dos Traductores.

Dados dos traductores, con la condición que sean o bien se correspondan con el mismo idioma origen y destino. Obtener el traductor suma, como la unión de los dos traductores.

El comando podrá ejecutarse desde la línea de órdenes de la siguiente manera:

```
prompt%  sumar frases_ingles_espanhol1.txt frases_ingles_espanhol2.txt
frases_ingles_español_suma.txt
```

Los parámetros son los siguientes:

4. El nombre del fichero con el primer traductores a sumar
5. El nombre del fichero con el segundo traductor a sumar
6. El nombre del fichero del traductor salida.

Como ejemplo para poder ejecutar este programa el alumno puede crear con los ficheros frases_español_frances.txt frases_frances_español.txt, puede en primer lugar crear el traductor inverso de frases_español_frances y a continuacion hacer la suma con frases_frances_español.txt.

3.4. Fichero prueba

Para poder probar nuestro programa usaremos un fichero compuesto de una serie de líneas. Cada línea se corresponde con una frase en el idioma origen y a continuación, separados por el carácter ';', las frases en el idioma destino. Por ejemplo un trozo del fichero inglés-español es el que se muestra a continuación:

…..

The apple doesn’t fall far from the tree;De tal palo tal astilla

The die is cast;La suerte esta echada

The early bird catches the worm;A quien madruga Dios lo ayuda

The grass is always greener on the other side of the fence;Lo mejor siempre lo tiene el otro

The shoemaker’s son always goes barefoot;En casa de herrero cuchara de palo

The squeaky wheel gets the oil;El que no llora no mama

There’s many a slip twixt cup and lip;Entre mano y boca desaparece la sopa;Del plato a la boca se pierde la sopa

Think twice act wise;Piensa dos veces antes de actuar

To call a spade a spade;Al pan pan y al vino vino

To err is human to forgive divine;Errar es humano perdonar es divino

Too many cooks spoil the broth;Demasiados cocineros estropean el caldo;Muchas manos en un plato hacen mucho garabato

Two in distress makes sorrow less;Las penas compartidas saben a menos

Variety is the spice of life;En la variedad esta el gusto

What a small world;El mundo es un panhuelo

What the eye doesn’t see the heart doesn’t grieve over;Ojos que no ven corazon que no siente

What goes around comes around;Se cosecha lo que se siembra

When in Rome do as the Romans do;Donde fueres haz lo que vieres

….

En el directorio **doc** tenéis ejemplos de ficheros traductores.

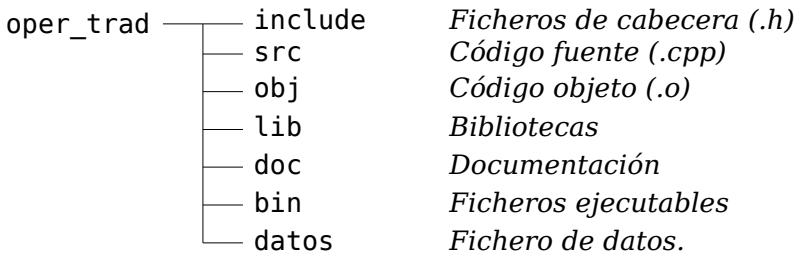
3.5. Módulos a desarrollar.

Habrá al menos dos módulos que deberéis desarrollar: 1) el módulo asociado a Frase (frase.cpp y frase.h) y 2) el módulo asociado a Traductor (traductor.cpp e traductor.h). Es posible añadir más módulos si lo estimáis necesario.

4. Práctica a entregar

El alumno deberá empaquetar todos los archivos relacionados en el proyecto en un archivo con nombre “oper_trad.*tgz*” y entregarlo antes de la fecha que se publicará en la página web de la asignatura. Tenga en cuenta que no se incluirán ficheros objeto, ni ejecutables, ni la carpeta datos. Es recomendable que haga una “limpieza” para eliminar los archivos temporales o que se puedan generar a partir de los fuentes.

El alumno debe incluir el archivo *Makefile* para realizar la compilación. Tenga en cuenta que los archivos deben estar distribuidos en directorios*:*



Para realizar la entrega, en primer lugar, realice la limpieza de archivos que no se incluirán en ella, y sitúese en la carpeta superior (en el mismo nivel de la carpeta "*oper_trad*") para ejecutar:

```
prompt% tar zcv oper_trad.tgz oper_trad
```

tras lo cual, dispondrá de un nuevo archivo oper_trad.tgz que contiene la carpeta oper_trad así como todas las carpetas y archivos que cuelgan de ella.