Práctica Final: Letras

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

Estructuras de Datos

Grado en Ingeniería Informática
Doble Grado en Ingeniería Informática y ADE
Doble Grado en Ingeniería Informática y Matemáticas

Índice de contenido

1. Introducción

El objetivo de esta práctica es manejar varios tipos de datos para la realización de búsquedas.

2. Ejercicios

Se deberán implementar varios programas. Uno de ellos simulará parte del juego “Cifras y Letras”. En este caso solamente deberá generar la sección de Letras. Este juego consiste en dada una serie de letras escogidas de forma aleatoria obtener la palabra existente en el lista_palabras formada a partir de ellas de mayor longitud.

Por ejemplo dadas las siguientes letras:

O D Y R M E T

una de las mejores soluciones sería METRO.

2.1. Tareas a realizar

Se deberán llevar a cabo las siguientes tareas:

1. Definir los TDA que vea necesario para la solución del problema propuesto.
2. Probar los módulos con programas test.
3. Construir los programas que a continuación se detallan.

A continuación se detallan los programas que se deberán desarrollar.

2.1.1. Test lista_palabras

Este programa permitirá comprobar el buen funcionamiento del TDA lista_palabras, representado como un set. Para ello el código fuente “testlista_palabras.cpp” deberá funcionar con los tipos (ver sección 3) desarrollados.

```
1.  #include <fstream>
2.  #include <iostream>
3.  #include <string>
4.  #include <vector>
5.  #include <set>
6.  int main(int argc, char * argv[]){
7.   if (argc!=2){
8.            cout<<"Los parametros son:"<<endl;
9.            cout<<"1.- El fichero con las palabras";
10.           return 0;
11.  }
12.  ifstream f(argv[1]);
13.  if (!f){
14.           cout<<"No puedo abrir el fichero "<<argv[1]<<endl;
15.   return 0;
16.  }
17.  lista_palabras D;
18.  cout<<"Cargando lista_palabras...."<<endl;
19.  f>>D;
20.  cout<<"Leido el lista_palabras..."<<endl;
21.  cout<<D;
22.
23.  int longitud;
24.   cout<<"Dime la longitud de las palabras que quieres ver";
25.  cin>>longitud;
26.                                   vector<string> v=D.palabras_longitud(longitud);
27.
28.           cout<<"Palabras de Longitud "<<longitud<<endl;
29.  for (unsigned int i=0;i<v.size();i++)
30.   cout<<v[i]<<endl;
31.
32.  string p;
33.  cout<<"Dime una palabra: ";
34.  cin>>p;
35.  if (D.Esta(p))   cout<<"Sí esa palabra existe";
36.  else   cout<<"Esa palabra no existe";
37.  }
```

En este código, se lee de un fichero una lista_palabras y luego se imprime en la salida estándar. A continuación se muestran todas las palabras de lista_palabras de una longitud dada. Y finalmente dada una palabra por el usuario, el programa indica si existe tal palabra en la lista_palabras o no. El estudiante creará por lo tanto el programa testlista_palabras, que se deberá ejecutar en la línea de órdenes de la siguiente manera:

```
prompt%  testlista_palabras spanish
```

El único parámetro que se da al programa es el nombre del fichero donde se almacena lista_palabras.

2.1.2. Letras

Este programa construye palabras de longitud mayor (o de puntuación mayor) a partir de una serie de letras seleccionadas de forma aleatoria. El programa letras se deberá ejecutar en la línea de órdenes de la siguiente manera:

```
prompt%  letras spanish letras.txt 8 L
```

Los parámetros de entrada son los siguientes:

1. El nombre del fichero con lista_palabras.
2. El nombre del fichero con las letras.

3. El número de letras que se deben generar de forma aleatoria.
4. Modalidad de juego:
    - Longitud: Si el parámetro es *L* se buscará la palabra más larga.
    - Puntuación: Si el parámetro es *P* se buscará la palabra de mayor puntuación.

Tras la ejecución en pantalla aparecerá lo siguiente:

```
Las letras son: T       I       E       O       I       T       U       S
Dime tu solución:tieso
tieso   Puntuación: 5

Mis soluciones son:
otitis  Puntuación: 6
tiesto  Puntuación: 6
Mejor Solución:tiesto
¿Quieres seguir jugando[S/N]?N
```

En primer lugar el programa genera 8 letras. Estas letras se escogen, de forma aleatoria, entre las dadas en el fichero letras (ver sección 2.2.2). Una vez generadas las letras, el programa pide al usuario su solución, en el ejemplo la solución dada es "tieso". A continuación se muestra la solución del usuario junto con su puntuación. Y finalmente se muestran las soluciones dadas por el programa. Para generar de forma aleatoria las letras con las que construir la palabra, el estudiante creará una *Bolsa de Letras* (contenedor de letras que se disponen de forma aleatoria) en la que el número de veces que aparece cada letra, en la *Bolsa de Letras*, viene dado por la columna *Cantidad* del fichero de letras.

En el caso de que el usuario haya escogido jugar en modo "*Puntuación*", como resultado se

```
******Puntuaciones Letras*******       Z       10
A       1
B       3                             Las letras son:
C       3                             N       S       A       O       T       O       A       I
D       2                             Dime tu solucion:sonata
E       1
F       4
G       2                             sonata  Puntuacion: 6
H       4                             Mis Soluciones son:
I       1                             asiano  Puntuacion: 6
J       8                             atonia  Puntuacion: 6
L       1                             ostion  Puntuacion: 6
M       3                             sonata  Puntuacion: 6
N       1                             sotana  Puntuacion: 6
O       1                             sotani  Puntuacion: 6
P       3                             sotano  Puntuacion: 6
Q       5                             tisana  Puntuacion: 6
R       1                             toison  Puntuacion: 6
S       1                             Mejor Solucion:toison
T       1                             ¿Quieres seguir jugando[S/N]?N
U       1
V       4
Y       4
```

obtendrán las palabras que acumulen una mayor puntuación. Para obtener la puntuación de la palabra simplemente tenemos que sumar las puntuaciones de las letras en la palabra (en el fichero de Letras viene descrita en la columna Puntos).

En ambas versiones, se le preguntará al usuario si quiere seguir jugando. En caso afirmativo se generará una nueva secuencia de letras aleatorias para jugar de nuevo. En otro caso el programa termina.

En la sección 2.2 se detallan los formatos de los ficheros de entrada al programa.

2.1.3. Cantidad de Letras

El programa cantidad_letras obtiene la cantidad de instancias de cada letra (ver fichero letras en la sección 2.2.2). El programa se deberá ejecutar en la línea de órdenes de la siguiente manera:

```
prompt%  cantidad_letras spanish letras.txt salida.txt
```

Los parámetros de entrada son los siguientes:

1. El nombre del fichero con lista_palabras
2. El nombre del fichero con las letras
3. El fichero donde escribir el conjunto de letras con la cantidad de apariciones calculada.

Este programa una vez haya cargado el fichero lista_palabras en memoria y el conjunto de letras, obtiene para cada letra en el conjunto el número de veces que aparece en lista_palabras, es decir encuentra la frecuencia de aparición. Finalmente obtiene el tanto por ciento, sobre el total de las frecuencias, del número de veces que aparece cada letra. Este valor aparecerá reflejado con la etiqueta cantidad.

A modo de ejemplo para la anterior ejecución, el fichero salida.txt podría ser:

```
#Letra Cantidad
A       14.03
B       1.75
C       5.26
D       4.38
E       8.77
F       1.75
G       1.75
H       0.88
I       7.89
J       0.88
L       4.38
M       3.51
N       6.14
O       8.77
P       2.63
Q       0.88
R       8.77
S       4.38
T       5.26
U       3.51
V       1.75
X       0.88
Y       0.88
Z       0.88
```

2.2. Ficheros

2.2.1. Fichero lista_palabras

El fichero lista_palabras se compone de un conjunto de palabras, cada una en un línea. Este conjunto de palabras serán las que se consideren como válidas.

Un ejemplo de fichero lista_palabras es el siguiente:

```
a
aaronica
aaronico
ab
abab
ababillarse
ababol
abaca
abacera
abaceria
abacero
abacial
abaco
abad
abada
abadejo
abadenga
abadengo
```

2.2.2. Fichero Letras

Un ejemplo de fichero de letras es el que se muestra a la derecha.

El formato del fichero es el siguiente:

1. En primer lugar aparece una línea encabezada con el carácter # donde se describe las columnas del fichero (Letra Cantidad Puntos)
2. A continuación cada línea se corresponde con la información de una letra:
   - Valor de la letra
   - Número de veces que aparece la letra en la Bolsa de Letras
   - Puntos asignados a la letra.

```
#Letra Cantidad Puntos
A      12      1
E      12      1
O      9       1
I      6       1
S      6       1
N      5       1
L      1       1
R      6       1
U      5       1
T      4       1
D      5       2
G      2       2
C      5       3
B      2       3
M      2       3
P      2       3
H      2       4
F      1       4
V      1       4
Y      1       4
Q      1       5
J      1       8
X      1       8
Z      1       10
```

3. Módulos a desarrollar

Módulo lista_palabras

Será necesario construir el TDA lista_palabras. Un objeto del TDA. lista_palabras almacena palabras de un lenguaje. El TDA lista_palabras será representado como un ***set*** *instanciado a string*. Así en líneas generales el módulo lista_palabras se detalla a continuación:

```
#ifndef __lista_palabras_h__
#define __lista_palabras_h__
#include <set>

class lista_palabras{
 public:
      /**
        @brief Construye un lista_palabras vacío.
      **/
      lista_palabras()

      /**
        @brief Devuelve el numero de palabras en el lista_palabras
      **/
      int size() const

      /**
        @brief Obtiene todas las palabras en el lista_palabras de una longitud dada
        @param longitud la longitud de las palabras de salida
        @return un vector con las palabras de longitud especificada en el parámetro de entrada
      **/
      vector<string> palabras_longitud(int longitud)

       /**
        @brief Indica si una palabra está en el lista_palabras o no
        @param palabra la palabra que se quiere buscar
        @return true si la palabra esta en el lista_palabras. false en caso contrario
      **/
      bool Esta(string palabra);

       /**
        @brief Lee de un flujo de entrada un lista_palabras
        @param is flujo de entrada
        @param D el objeto donde se realiza la lectura.
        @return el flujo de entrada
      **/
      friend   istream & operator>>(istream & is, lista_palabras  &D);

       /**
        @brief Escribe en un flujo de salida un lista_palabras
        @param os flujo de salida
        @param D el objeto lista_palabras que se escribe
        @return el flujo de salida
      **/
      friend ostream & operator<<(ostream & os, const lista_palabras  &D);
  private:
    set<string> datos;
};
#endif
```

Se debe incluir un iterador sobre el T.D.A lista_palabras que nos permita recorrer de manera ordenada todas las palabras del lista_palabras. La especificación y representación de este iterador es:

```
#ifndef __lista_palabras_h__
#define __lista_palabras_h__
#include <set>

class lista_palabras{
 public:
  …

 class iterator {
    public:
       iterator ();
       string operator *();
       iterator & operator ++();
       bool operator ==(const iterator &i)
       bool operator !=(const iterator &i)
       friend class lista_palabras;
     private:
       set<string>::iterator it;
 };
 iterator begin();

 iterator end();
 private:
   set<string> datos;
};

#endif
```

Módulo Letra y Conjunto de Letras

El T.D.A Letra almacena una letra. Una letra se especifica con tres valores:

1. El carácter de la propia letra
2. La cantidad de veces que puede aparecer.
3. La puntuación de una letra.

El T.D.A conjunto_letras permitirá tener en memoria un fichero Letras. Este T.D.A se define como una colección de letras, en las que no hay letras repetidas.

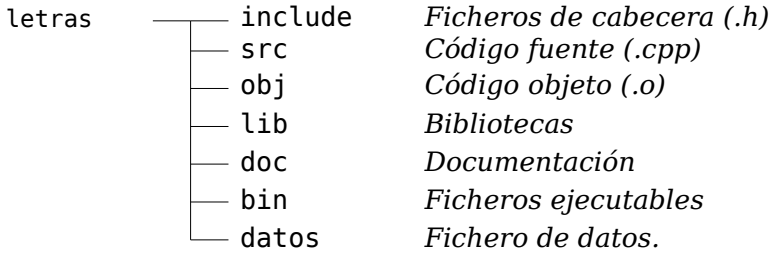
Módulo Bolsa de Letras

Este módulo será útil para el programa *letras.* El T.D.A bolsa_letras almacena caracteres correspondientes a una letra de un Conjunto de Letras. Este carácter aparece en la bolsa_letras repetido tantas veces como diga el campo cantidad de la letra. Por lo tanto en la Bolsa de Letras aparecen las letras de forma aleatoria. En el programa *"letras"* la secuencia de letras con las que se juega se cogen de la Bolsa de Letras.

4. Entrega.

Se deberá empaquetar todos los archivos relacionados en el proyecto en un archivo con nombre "*letras.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. Se debe realizar una "limpieza" para eliminar los archivos temporales o que se puedan generar a partir de los fuentes.

El estudiante 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 "*letras*") para ejecutar:

```
prompt% tar zcv letras.tgz letras
```

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

La práctica se puede realizar por parejas. Ambos miembros de la pareja deberán subir los ficheros a la plataforma docente a través del enlace correspondiente a la entrega de la práctica final