UNIVERSIDAD DE MÁLAGA

PROGRAMACIÓN II

GRADO EN INGENIERÍA TELEMÁTICA
PRÁCTICA 4 DE LABORATORIO
ACTIVIDAD NO PRESENCIAL

1. Implementar el TAD *Cola* genérico siguiendo las especificaciones descritas en clase. Utilizar el espacio de nombre `tads`.

2. Implementar el TAD *Tabla Asociativa* que permita asociar una cadena de caracteres a otra utilizando una estructura de datos de tipo hash. Utilizar el espacio de nombre `tads`. Una tabla asociativa permite asociar una serie de valores llamados *clave* a otros llamados *valor asociado*. Por ejemplo, una tabla asociativa donde el valor clave es una cadena de caracteres y el valor asociado es otra cadena de caracteres podría almacenar una lista de países y sus capitales de forma que, dado el nombre de un país, la tabla asociativa devolvería el nombre de su capital.
Una forma de implementar una tabla asociativa consiste en utilizar una estructura de datos llamada *tabla hash dinámica*. Una *tabla hash dinámica* consiste en un array de listas enlazadas donde, en cada nodo de estas lista, se almacena un par (*clave, valor asociado*). Esta estructura utiliza una *función hash* que, a partir del valor de la clave, devuelve el número de la lista en que se almacenará esa clave.
La estructura de datos en C++ utilizada para implementar una tabla *hash dinámica* con el tipo `string` como clave y valor asociado es la siguiente:

```
const int TAM = 100;
struct TNodo{
     string clave, valor;
     TNodo* sig;
};
typedef TNodo* TLista;
typedet TLista THash[TAM];
```

Por ejemplo, utilizando la siguiente *función hash*:

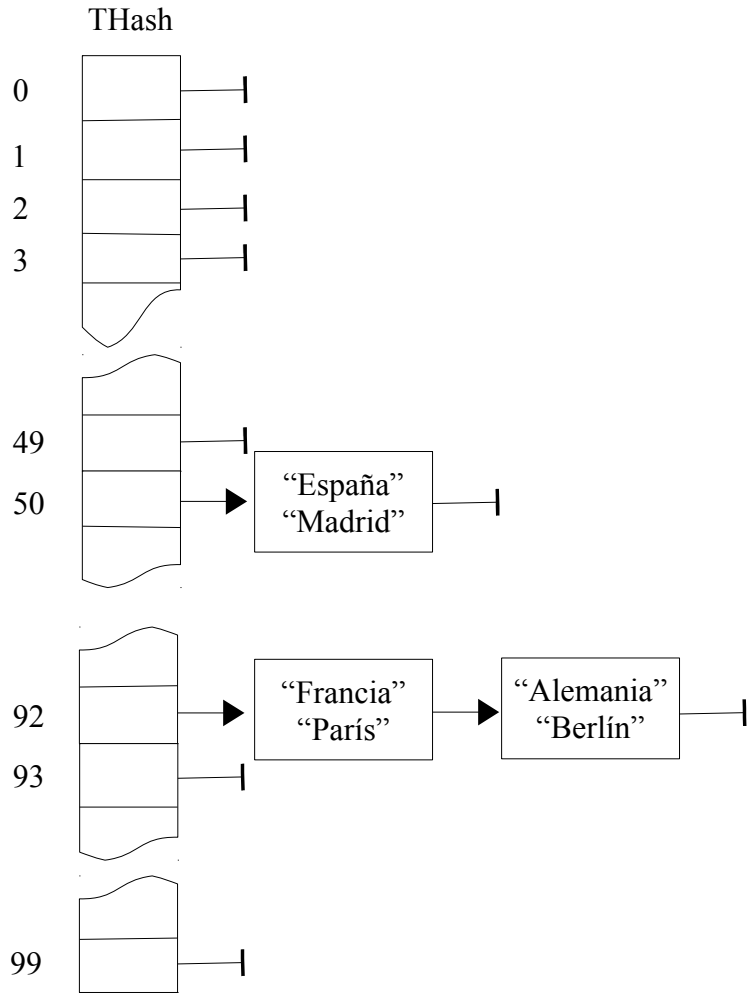
```
unsigned fHash( const string& c ){
        unsigned suma;

        suma = 0;

        for( unsigned k = 0; k < c.size(); k++ ){
                suma = suma + unsigned( c[k] );
        }

        suma = suma % TAM;
        return suma;
}
```

Almacenar los pares (“España”, “Madrid”), (“Francia”, “Paris”) y (“Alemania”, “Berlín”) en la tabla hash descrita anteriormente quedaría de la siguiente forma (NOTA: el valor que devuelve la función hash para “España” es 50; para “Francia” es 92 y para “Alemania” es 92):



Las operaciones del TAD *tabla asociativa* son:

- `void incluir( const std::string& cl, const std::string& va );` Añade la clave `cl` a la table con el valor asociado `va`. Si la clave se encuentra en la tabla sobreescribiría su valor asociado (No puede haber dos claves iguales en la tabla).
- `std::string obtener( const std::string& cl ) const;` Devuelve el valor asociado a la clave `cl`. Si la clave no esta en la tabla muestra un mensaje de error y devuelve la cadena vacía.
- `bool existe( const std::string& cl ) const;` Devuelve verdadero si la clave cl se encuentra en la tabla. Falso en caso contrario.
- `void borrar( const std::string& cl );` Elimina la clave cl y su valor asociado de la tabla. Si la clave no existe no hace nada.
- `void grabar( const std::string& f ) const;` Graba la tabla en el fichero cuyo nombre se le pasa como parámetro. El fichero contendrá dos líneas por cada pareja (clave, valor) almacenada en la tabla: una línea con la clave y la siguiente con su valor asociado. Por ejemplo, para la tabla del ejemplo anterior el fichero contendría las siguientes líneas:

  España
  Madrid
  Francia
  Paris
  Alemania
  Berlín

- `void cargar( const std::string& f );` Carga la tabla del fichero cuyo nombre se le pasa como parámetro. Deberá eliminar cualquier asociación almacenada en la tabla antes de leer el fichero. El fichero contendrá el mismo formato que el método lectura. Si el fichero no existe mostrará un mensaje de error y dejará la tabla vacía.

3. Implementar los siguientes subalgoritmos sobre cadenas de caracteres:

- `string consecuentes( const string& s, const string& p );` Devuelve una cadena con todos los caracteres que aparecen en la cadena s inmediatamente detrás la subcadena p. En la cadena devuelta no debe haber caracteres repetidos. Por ejemplo, dada s = “ZAABCAAADAABDDAHAA” y p = “AA” el subalgoritmo devolvería la cadena “BAD” ya que después de la cadena “AA” en la cadena s aparecen solamente los caracteres 'B', 'A' y 'D'.
- `string antecedentes( const string& s, const string& p );` Devuelve una cadena con todos los caracteres que aparecen en la cadena s inmediatamente delante de la subcadena p. En la cadena devuelta no debe haber caracteres repetidos. Por ejemplo, dada s = “AXBCAXDAXBDDAHAX” y p = “AX” el subalgoritmo devolvería la cadena “CDH” ya que delante de la cadena “AX” en la cadena s aparecen los caracteres 'C', 'D' y 'H'.