ED – PRÁCTICA 2 : ABSTRACCIÓN

SUBGRUPO C3, AUTOR : Jesús Ruiz Castellano DNI : 76439001 - L

INDICE

ED – PRÁCTICA 2 : ABSTRACCIÓN

SUBGRUPO C3, AUTOR : Jesús Ruiz Castellano DNI : 76439001 - L

TDA FRASE

1.ESPECIFICACIÓN

Una instancia c del tipo de dato abstracto Frase es un objeto que mantiene la información de una frase "hecha" en el idioma origen, y todas las posibles traducciones en el idioma destino.

Ejemplo: No worries;Sin problema;No te preocupes

OPERACIONES BÁSICAS

- Constructores, Destructor y Operador de asignación.
- Consultores :
  - **NumTraducciones :** Devuelve el número de traducciones de la frase.
  - **GetOrigen :** Devuelve la frase origen.
  - **GetDestino :** Devuelve la cadena de las frases traducciones.
- Operadores :
  - **Indexación e indexación constante / Consulta y consulta constante :** Devuelve la traducción en la posición i, pasada como parámetro.
  - **Comparación (==) :** Compara si las frases son iguales tanto la origen, como las traducciones, y en el mismo orden. Devuelve si son iguales, o no lo son.
  - **Operadores de E/S :** Se encargan de gestionar la lectura de una frase desde un flujo cualquiera, ya sea un fichero o el mismo teclado. Y de imprimirla en el mismo formato que tienen en los ficheros dados.

ED – PRÁCTICA 2 : ABSTRACCIÓN

SUBGRUPO C3, AUTOR : Jesús Ruiz Castellano DNI : 76439001 - L

2.DIFERENTES ED PARA EL *tipo Rep*

1. PRIMERA OPCIÓN

Meter todo en un único string. Demasiado complicado a la hora de implementar los métodos necesarios.

```
class Frase {
private:

  string frase;

public:

  ...
}
```

2. SEGUNDA OPCIÓN

Utilizar un vector estático para las traducciones. Error, ya que no sabemos, a priori, cuántas traducciones tiene cada frase.

```
class Frase {
private:

  string frase_origen;
  int NTraduc;
  string[Ntraduc] traducciones;

public:

  ...
}
```

3. TERCERA OPCIÓN (ELEGIDA)

Utilizar un vector dinámico para almacenar las traducciones

```
class Frase {
private:

  string origen;
  int nTraduc;
  string* cadTraducciones;

public:

  ...
}
```

ED – PRÁCTICA 2 : ABSTRACCIÓN

SUBGRUPO C3, AUTOR : Jesús Ruiz Castellano DNI : 76439001 - L

3.TIPO REP ELEGIDO

Además de éstos datos miembro privados, también tiene un método privado (bool Invariante() const;), que llamaré en los métodos de la clase que crea oportunos, para asegurarme que se cumple el Invariante.

```
class Frase {
private:

  string origen;
  int nTraduc;
  string* cadTraducciones;

public:

  ...
}
```

4. INVARIANTE DE LA REPRESENTACIÓN Y FUNCIÓN DE ABSTRACCIÓN

INVARIANTE DE LA REPRESENTACIÓN

- rep.nTraduc => 1.
- rep.cadTraducciones[i] != rep.cadTraducciones[j],
  Para todo i,j => $i \neq j$.
- rep.cadTraducciones.size() == nTraduc.

FUNCIÓN DE ABSTRACCIÓN

Un objeto valido rep del TDA Frase representa al valor:

fA(rep)=
rep.origen;rep.cadTraducciones[0];...;rep.cadTraducciones[i];...;rep.cadTraducciones[rep.nTraduc-1];

ED – PRÁCTICA 2 : ABSTRACCIÓN

SUBGRUPO C3, AUTOR : Jesús Ruiz Castellano DNI : 76439001 - L

TDA CONJUNTO DE FRASES

1.ESPECIFICACIÓN

Una instancia c del tipo de dato abstracto ConjuntoFrases es un objeto que contiene una sucesión de frases leídas desde un fichero, o cualquier entrada posible.

Ejemplo :

```
Not Bad;No esta mal
Not for nothing;No es por nada
Not half;Ya lo creo
Not half;Por supuesto
Not my business;No es asunto mio
...
```

OPERACIONES BÁSICAS

- Constructores, Destructor y Operador de asignación.
- Consultores :
  - **Esta :** Recorre el fichero buscando en las origen de cada frase y devuelve si el string pasado como parámetro coincide con alguna frase origen.
  - **GetTraducciones :** Dado el campo origen de una frase, que está pasado como parámetro, devuelve la frase completa a la que pertenece.
  - **Contenga :** Devuelve un conjunto de frases que, en cuya frase origen, contenga la subcadena dada como parámetro.
  - **Size :** Devuelve el número de frases que contiene el conjunto.

- Operadores :
  - **Indexación e indexación constante / Consulta y consulta constante :** Devuelve la frase en la posición i, pasada como parámetro.
  - **Operadores de E/S :** Se encargan de gestionar la lectura de un conjunto de frases desde un flujo cualquiera, ya sea un fichero o el mismo teclado. Y de imprimirlo en el mismo formato que tiene en los ficheros dados.

2. DIFERENTES ED PARA EL *tipo Rep*

1. PRIMERA OPCIÓN

Meter todo en un único vector estático de string. Mala opción, puesto que no sabemos cuantas frases tenemos. Nos restringiría mucho ésta implementación.

```
class ConjuntoFrases {
private:

  static const int TAM = 500;
  int utilizadas = 0;
  string[TAM] frases;

public:

  ...
}
```

2. SEGUNDA OPCIÓN

Utilizar un vector dinámico de Frases. Si no le indicamos un tamaño a priori, no sabemos cuánta memoria reservar, por lo cual sería un error.

```
class ConjuntoFrases {
private:

  Frase* Conjunto;

public:

  ...
}
```

3. TERCERA OPCIÓN (ELEGIDA)

Utilizar un vector dinámico de Frases. Definiendo, también, un tamaño inicial y un contador de las posiciones ocupadas.

```
class ConjuntoFrases {
private:

  int TAM;
  int ocupadas;
  Frase* ConjF;

public:

  ...
}
```

3.TIPO REP ELEGIDO

Además de éstos datos miembro privados, también tiene un método privado (bool Invariante() const;), que llamaré en los métodos de la clase que crea oportunos, para asegurarme que se cumple el Invariante, y otro, privado también (void MoreSize();), que se encargará de redimensionar, cuando sea necesario, el tamaño del conjunto de frases.

```
class ConjuntoFrases {
private:

  int TAM;
  int ocupadas;
  Frase* ConjF;

public:

  ...
}
```

4. INVARIANTE DE LA REPRESENTACIÓN Y FUNCIÓN DE ABSTRACCIÓN

INVARIANTE DE LA REPRESENTACIÓN

- rep.ConjF[i] != rep.ConjF[j], Para todo i,j => $i \neq j$
- rep.ocupadas >= 2
- rep.ocupadas < rep.TAM

FUNCIÓN DE ABSTRACCIÓN

Un objeto valido rep del TDA Conjunto de Frases representa al valor:

```
fA(rep) = rep.ConjF[0]
          rep.ConjF[1]
          …
          rep.ConjF[rep.ocupadas-1]
          …
          rep.ConjF[i]
          …
          rep.ConjF[rep.TAM-1]
```