Programación II - I.T.I. Sistemas

Práctica del curso 2003/04

I. Objetivos

La práctica consiste en desarrollar, utilizando las técnicas descritas en la asignatura, una aplicación que permita crear un fichero con el texto introducido por un usuario pero con una característica especial: **Para la introducción del texto sólo se utilizarán 3 teclas**. La idea es permitir la introducción de texto sencillo en dispositivos que tengan un teclado limitado (televisores, lavadoras, teléfonos móviles, etc.)[1].

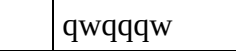
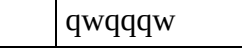
El tipo de texto que se va a permitir es muy simple: Consistirá únicamente en palabras separadas por espacios en blanco. Cada palabra consistirá en una secuencia de caracteres que pertenecen a un determinado alfabeto. No se plantea la posibilidad de estructurar el texto ni de darle formato: A todos los efectos el texto es una única línea formada por una secuencia de palabras.

El modo de operación será muy sencillo: El usuario introduce el **código** asociado a una palabra, la aplicación busca en un diccionario aquellas palabras que pueden corresponder al código y las lista al usuario, el usuario elige la palabra correcta o bien, si la palabra no está en el listado, la teclea para que se incorpore al diccionario. La palabra se incorpora al texto de salida y se vuelve a repetir el proceso hasta que el usuario indique que ha terminado.

Para introducir una palabra el usuario sólo utiliza dos teclas (las denominaremos T_1 y T_2). La tercera tecla (que para nosotros será la tecla de retorno) sólo sirve para indicar el final de la palabra.

El código asociado a cada palabra se obtiene de la siguiente forma: Supongamos que el alfabeto permitido está formado por los caracteres α_1, α_2, ..., α_n en ese orden. Se elige un índice, *m*, que divide el alfabeto en dos partes. Para introducir una palabra el usuario pulsa, por cada carácter de la palabra, la tecla T_1 si el carácter está en el conjunto α_1, ..., α_m o bien la tecla T_2 en caso contrario (el carácter pertenece a α_{m+1}, ..., α_n).

Por ejemplo, si las teclas T_1 y T_2 fueran  y  , respectivamente, y el alfabeto el español, eligiendo como **carácter separador** la letra *l* obtendríamos los siguientes códigos:

Palabra	Teclas pulsadas	Código
programa		wwwqwqwq
collar		qwqqqw
codigo		qwqqqw

[1] Esta práctica se inspira en la tecnología de texto predicitivo T9, usada en teléfonos móviles, pero llevada al extremo. Para más información sobre T9 podéis visitar la página http://www.tegic.com

Como se puede apreciar en el ejemplo a un mismo código pueden corresponder muchas palabras distintas (se pierde información al traducir una palabra a su código). La aplicación buscará en un diccionario las palabras que tienen el mismo código que el introducido por el usuario para que este elija la palabra adecuada (o introduzca una nueva).

La aplicación se desarrollará en Eiffel, y constará en varios ficheros (uno por cada clase). Todos los métodos/subprogramas *relevantes* deberán definir un método/subprograma adicional encargado de someterlo. Además, se deberann incluir precondiciones, postcondiciones, e invariantes y variantes de bucle.

Aparte del programa se incluirá documentación referente a la descomposición modular y a la documentación interna de la aplicación (ver el último apartado, presentación de la práctica).

II. Descripción detallada

La aplicación usará el siguiente alfabeto:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z

Sin embargo, se deberá diseñar de manera que se facilite su **extendibilidad** para la utilización de otros alfabetos (Ejemplo: añadir caracteres que representan la letra *ñ* y los acentos, usar alfabeto cirílico, etc.) Se debe tener en cuenta que al usar otros alfabetos el **orden** de los caracteres dentro de él puede ser distinto del que proporcionan las funciones de comparación de valores de tipo CHARACTER de Eiffel.

La aplicación comienza su ejecución con los siguientes valores iniciales: diccionario vacío (sin ninguna palabra), las teclas usadas para introducir texto son  y  y el carácter separador la letra *l*. Al comienzo de su ejecución la aplicación presenta un menú parecido al siguiente:

```
Menu Principal
  [1] Cargar Diccionario
  [2] Guardar Diccionario
  [3] Definir teclas
  [4] Cambiar caracter separador (actual: l)
  [5] Optimizar carácter separador
  [6] Introducir texto
  [7] Mostrar texto
  [0] Salir del programa
Elija Accion: _
```

Fig. 1: Menú principal

- La acción “Cargar diccionario” pedirá al usuario el nombre de un fichero de texto. El fichero constará de una serie de líneas, cada una de ellas conteniendo una sola palabra. Estas palabras no tienen porqué estar ordenadas. Todas las palabras contendrán únicamente caracteres pertenecientes al alfabeto. Se puede suponer que no existen palabras con más de 20 caracteres y que el número de palabras del diccionario es menor de 50000.
- La acción “Guardar diccionario” pedirá al usuario el nombre del fichero donde se guardará el diccionario[2], con el formato indicado en el párrafo anterior.
- La acción “Definir teclas” pedirá al usuario los caracteres que definen a las dos teclas que se utilizan para introducir texto.
- La acción “Cambiar carácter separador” pedirá al usuario el carácter que sirve para dividir el alfabeto en dos partes, una asignada a la primera tecla y la otra a la segunda.

[2] Realmente no es un diccionario sino un *corpus*, puesto que sólo almacena un listado de palabras, no sus definiciones.

- La acción “Optimizar carácter separador” escoge el mejor valor para el carácter separador dado el diccionario actual. El mejor valor es aquel que minimiza el número de palabras que pueden dar lugar al mismo código. Para encontrarlo se debe asignar al carácter separador cada uno de los caracteres del alfabeto, y para cada carácter se mide el número de códigos **distintos** (llamémosle *p*) que se obtienen de todas las palabras del diccionario. El mejor carácter será aquel que hace máximo el valor de *p*.
- La acción “Mostrar texto” muestra por pantalla el texto introducido por el usuario.

Acción Introducir Texto

Nota: En los ejemplos siguientes se supondrá que las teclas permitidas son  y  , y que el carácter separador es la letra *l*.

En este acción es donde el usuario introduce el texto utilizando únicamente 3 teclas (en las acciones anteriores no se aplicaba ésta restricción). La introducción de texto se realiza palabra por palabra. La aplicación pide que se introduzca una palabra, y el usuario introduce su código utilizando sólo dos teclas, pulsando retorno cuando finaliza la palabra. A partir del código obtenido, el programa obtiene del diccionario las *k* palabras cuyo código es idéntico.

El usuario debe ahora indicar cual de las *k* palabras es la correcta o si ninguna de ellas coincide con la que quería introducir. Si no tuviéramos la restricción de utilizar únicamente dos teclas podríamos presentar un listado de éste tipo:

```
Elija palabra o pulse retorno si no es ninguna de ellas:
  [0] palabra1
  [1] palabra2
  [2] palabra3
  ...
  [k-1] palabrak
Escoja Palabra: 5
```

y pedir que introdujera el número correspondiente a la palabra elegida. Sin embargo, como sólo se pueden usar dos teclas, en lugar de identificar cada palabra de la lista por un número normal (en base diez) lo identificaremos **por un número en base dos**, donde la primera tecla se identificará con el cero y la segunda tecla con el uno.

```
Introduzca palabra (pulse retorno para finalizar): wwwwqq
Existen 14 palabras en diccionario con ese código.
Elija palabra o pulse retorno si no es ninguna de ellas:
  [qqqq] mortal
  [qqqw] motril
  [qqwq] normal
  [qqww] nutria
  [qwqq] otorga
  [qwqw] portal
  [qwwq] postal
  [qwww] propia
  [wqqq] provee
  [wqqw] puntal
  [wqwq] rompia
  [wqww] sonrei
  [wwqq] sonrie
  [wwqw] tropel
Elija Palabra: _
```

Fig. 2 : Menú de selección de palabra

En la Fig. 2 se puede ver un ejemplo de cómo aplicar ésta técnica. El usuario introduce el código wwwwqq , y en el diccionario existen 14 palabras con ese mismo código. Se listan las 14 palabras numeradas de 0 (0000 en binario) a 13 (1101 en binario), cambiando los ceros por la primera tecla (*q*) y los unos por la segunda (*w*). De esta forma el usuario sólo necesita introducir una cadena de texto formada únicamente por *q*'s y *w*'s para indicar la palabra correcta, la cual se añade al texto, y se vuelve a pedir la siguiente palabra, hasta que el usuario introduzca una cadena vacía y en ese caso se da por terminada la introducción de texto y se muestra el menú principal.

Si ninguna de las palabras listadas es la correcta, el usuario pulsa directamente retorno (se obtiene una cadena vacía) y en ese caso se pasaría a la parte de la aplicación donde se introducen nuevas palabras en el diccionario.

```
Introduzca la nueva palabra utilizando el siguiente código
para cada carácter:
[qqqqq]: a  [qqqqw]: b  [qqqwq]: c  [qqqww]: d
[qqwqq]: e  [qqwqw]: f  [qqwwq]: g  [qqwww]: h
[qwqqq]: i  [qwqqw]: j  [qwqwq]: k  [qwqww]: l
[qwwqq]: m  [qwwqw]: n  [qwwwq]: o  [qwwww]: p
[wqqqq]: q  [wqqqw]: r  [wqqwq]: s  [wqqww]: t
[wqwqq]: u  [wqwqw]: v  [wqwwq]: w  [wqwww]: x
[wwqqq]: y  [wwqqw]: z

Palabra: qqwwwqwwwqqwqwwqqqqq

Ha escrito la palabra hola
  [q] Correcto
  [w] Volver a introducir
Escoja opcion: _
```

Fig.3 : Menú de introducción de palabra

En la Fig. 3 se puede ver cómo se introduce la nueva palabra: Cada carácter del alfabeto se identifica con un número binario usando la misma codificación que antes, y el usuario escribe la palabra sustituyendo cada carácter por el código binario asociado a él.

III. Consideraciones sobre el diseño de la aplicación

Uno de los objetivos de la práctica es que en su realización se utilicen las técnicas y criterios estudiados en la asignatura.

- **Extendibilidad**: La aplicación se debe diseñar de manera que sea posible adaptarla fácilmente al uso de un alfabeto distinto del indicado.
- **Robustez**: La aplicación debe ser *tolerante* respecto a los fallos en la introducción de datos por parte del usuario. Cuando detecte un dato erróneo debe indicarlo y volverlo a pedir al usuario. En otros aspectos, como por ejemplo que el fichero diccionario tenga el formato correcto (los caracteres pertenezcan al alfabeto, no existan palabras repetidas, no existan líneas vacías, etc) no se exige, aunque se tendrá en cuenta favorablemente, un comportamiento robusto por parte de la aplicación.
- **Eficiencia**: El programa deberá poder realizar las operaciones en un tiempo *razonable*. Por ejemplo, con el diccionario de 33000 palabras que se proporciona en la página de la asignatura la búsqueda de las palabras asociadas a un código debe tardar menos de 5 segundos, y la operación de optimizar el carácter separador menos de 5 minutos. En caso contrario se debería pensar en utilizar algoritmos o estructuras de datos diferentes. Respecto al uso de memoria, no existen restricciones señalables y en concreto es perfectamente posible almacenar el contenido del diccionario en memoria.

- **Corrección**: Los métodos deben incluir, en la mayor medida posible, precondiciones, poscondiciones e invariantes que garanticen su corrección.
- **Pruebas**: Todos los métodos *relevantes* deberán definir un método adicional (cuyo nombre sea test_X donde X es el nombre del método probado) encargado de someter a X a una batería de pruebas.

IV. Presentación de la práctica

La entrega de la práctica se realizará durante el examen teórico de la asignatura. La documentación que se debe entregar consiste en lo siguiente:

- El código fuente del programa en Eiffel que resuelve el problema planteado, es decir los ficheros de **texto** que contienen la definición de las clases que componen la aplicación. En la definición de las clases también deben aparecer los métodos utilizados para aplicar los casos de prueba.
- El diagrama modular de la aplicación. Se entiende por módulo cualquier método de cualquier clase que forme parte de la aplicación, **salvo** los métodos o clases predefinidos y los métodos utilizados para aplicar los casos de prueba.
- Un listado con la cabecera, precondición y postcondición de cada método, indicando si la postcondición es parcial o total, si se ha sometido el método a batería de pruebas y en caso afirmativo la(s) estrategia(s) de prueba utilizada(s).

El código fuente del programa se debe presentar en un disquete rotulado con el nombre del alumno. El resto de la documentación se puede presentar en papel o bien incluir en el disquete en alguno de los formatos siguientes: HTML, Word, T_EX (DVI), Acrobat PDF.

Si alguien desea presentar la documentación por correo electrónico debe hablar primero con el profesor.