Ejercicios Cadenas

Descargar estos ejercicios

Índice

Ejercicio 1

Realiza un programa que se repita hasta pulsar **ESC**. En el aparecerá un menú con 8 opciones que corresponderán a los siguientes ejercicios. Utiliza un **switch** para seleccionar cada opción y los métodos necesarios para que el programa quede claro y **no se repita código**:

1. Lea una cadena y a continuación muestra en pantalla su longitud.
2. Introduce dos cadenas **cadena1** y **cadena2**. Muestra en que posición comienza la última aparición de la **cadena2** en la **cadena1** o un texto indicando que una no contiene a la otra.
3. Introduce una cadena y luego muestra la cadena pero con todas las letras en mayúsculas.
4. Introduciremos una cadena por teclado y encadenaremos la cadena “LA CADENA FIN HA SIDO AÑADIDA A NUESTRA CADENA” **`(utiliza un método de la clase String para resolverlo, no el operador +)`** . Visualizaremos el resultado por pantalla.
5. Introduce una cadena y elimina los N caracteres a partir de la posición P indicada **`(estos dos números también se introducirán por teclado)`** . Mostrando posteriormente el resultado.
6. Introduce una cadena y elimina todos los carácteres de espaciado, saltos de línea o puntos que se puedan encontrar al final de ella (utiliza triming para resolverlo).
7. Introduce una cadena y un número. Añade al final de la cadena tantos carácteres **!** como el número introducido (utiliza Padding para resolverlo).

8. Introduce un texto y posteriormente dos cadenas. Sustituye las apariciones de la primera cadena en el texto por la segunda cadena. Mostrando posteriormente el resultado.

Nota: Para no repetir código a la hora de recoger la cádena para los distintos puntos, puedes crear un método encargado de esa función.

☑ Ejercicio 2

Programa para practicar con **StringBuilder** (obligatorio su uso). El programa deberá recoger una cadena con un texto acabado en '**.**' y que pasaremos a codificar u ofuscar. Para ello crearemos tres tipos de codificación, cada una la realizaremos en un método diferente y se usará StringBuilder para desarrollarla. Consisten en lo siguiente:

1. Codificación **especular**, es la famosa manera que tenía Leonardo Da Vinci de escribir volviendo las palabras del revés, de forma que es más sencillo su lectura mediante un espejo.

👉**Pista:** puedes leer el texto hasta un espacio en blanco (fin de palabra), guardando la posición de inicio de palabra (carácter después de espacio o primero). Y hacer la sustitución entre los dos indices.

2. Codificación **ofuscación cambiando carácteres de puntuación**, esta manera de ofuscar la frase será mediante el cambio de los carácteres de puntuación (, : . ; ? ¿ ! ¡), por algunos carácteres especiales generados aleatoriamente.

👉**Pista:** puedes usar una cadena para guardar los carácteres especiales y el método Contains para saber si un caracter está entre ellos. Para generar los carácteres aleatorios puedes usar el rango de ASCII entre 224-238.

3. Codificación **ofuscación quitando espacios en blanco**, en este caso solamente se eliminarán los espacios en blanco que hayan en la frase.

Un ejemplo de ejecución sería el siguiente:

```
Introduce la frase a ofuscar:
La verdadera felicidad cuesta poco; si es cara, no es de buena clase.
aL aredadrev dadicilef atseuc ;ocop is se ,arac on se ed aneub esalc
aL aredadrev dadicilef atseuc ëocop is se íarac on se ed aneub esalc
aLaredadrevdadicilefatseucâocopisseçaraconseedaneubesalc
```

✅ Ejercicio 3

El programa a realizar es una versión del famoso **juego del ahorcado**. Para ello el programa en primer lugar, pedirá al usuario la palabra secreta y el máximo de fallos permitido..

```
Introduce la palabra a adivinar: BUCLE
Introduce el número máximo de fallos: 3
```

- Tras introducir estos datos, la pantalla se borrará y comenzaremos con el juego. Mostrando en cada iteración los huecos o aciertos de la palabra secreta, así como las letras falladas hasta el momento.
- El usuario irá introduciendo letras hasta que acierte la palabra o supere el número de fallos.
- Tras introducir estos datos, la pantalla se borrará y comenzaremos con el juego. Mostrando en cada iteración los huecos o aciertos de la palabra secreta, así como las letras falladas hasta el momento.
- El usuario irá introduciendo letras hasta que acierte la palabra o supere el número de fallos.

Ejemplo de ejecución alcanzando el máximo de fallos:

```
Palabra: _ _ _ _ _
Fallos:
Introduce una letra: M

Palabra: _ _ _ _ _
Fallos: M
Introduce una letra: O

Palabra: _ _ _ _ _
Fallos: M O
Introduce una letra: L

Palabra: _ _ _ L _
Fallos: M O
Introduce una letra: T

Palabra: _ _ _ L _
Fallos: M O T
Lo siento has llegado al máximo de fallos permitido.
La palabra a adivinar era: BUCLE.
```

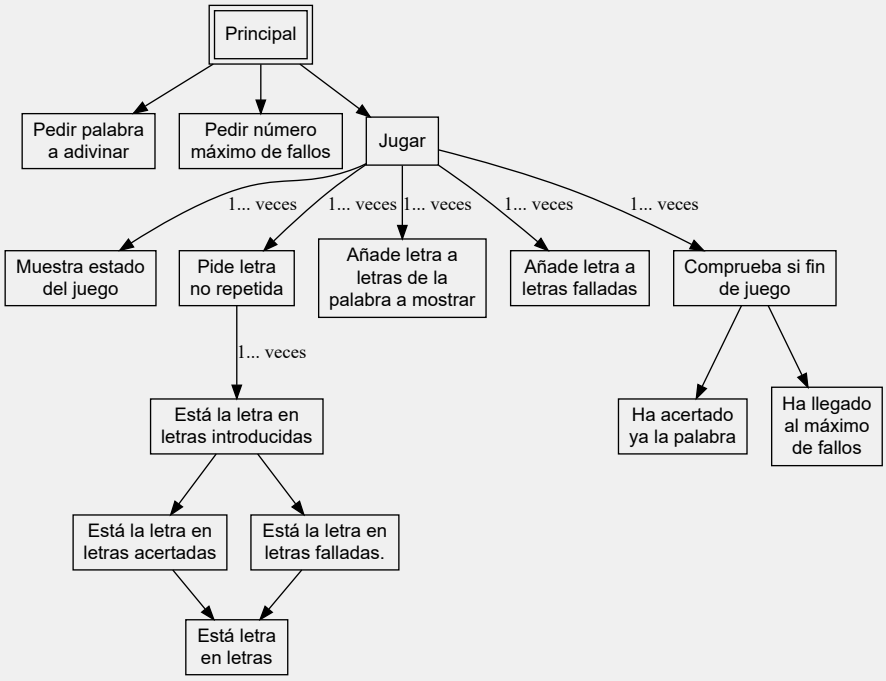
Ejemplo de ejecución acertando:

```
Palabra: _ _ _ _ _
Fallos:
Introduce una letra: U

Palabra: _ U _ _ _
Fallos:
Introduce una letra: C

Palabra: _ U C _ _
Fallos:
Introduce una letra: B

Palabra: B U C _ _
Fallos:
Introduce una letra: L

Palabra: B U C L _
Fallos:
Introduce una letra: E

Palabra: B U C L E
Fallos:
ENHORABUENA LO HAS CONSEGUIDO
```

👉**Pista:** la rama del DEM que se encarga de **pedir letra no repetida**, solamente estará recogiendo carácteres al usuario hasta que introduzca uno no repetido. Ese carácter no lo guardará en ningún sitio. Los métodos que se encargan de añadir la letra a los StringBuilders correspondientes son AñadeLetraALetrasParaMostrar o AñadeLetraALetrasFalladas (aunque se tenga que volver a recorrer las cadenas).

Tips a seguir en la resolución del ejercicio:

1. Tanto la palabra secreta, como las letras introducidas, las pasaremos a mayúsculas para no tener problemas a la hora de buscarlas y compararlas.
2. Utiliza StringBuilder para guardar las letras acertadas y las falladas.
3. Sigue el siguiente DEM a la hora de diseñar tus módulos o funciones:



Una posible propuesta de algunos interfaces para modularizar el DEM de arriba puede ser…

```
string PidePalabraAAdivinar()
int PideMaximoFallos()
bool EstaLetraEnLetras(char letra, string letras) //Puedes usar IndexOf
char PideLetraNoRepetida(
         string palabraParcialmenteAdivinada,
         string letrasFalladas)
void MuestraEstadoJuego(
         string palabraParcialmenteAdivinada,
         string letrasFalladas)
void AñadeLetraALetrasPalabraAMostrar(
         string palabraAAdivinar,
         in char letra,
         StringBuilder palabraParcialmenteAdivinada)
bool FinDeJuego(
         int numFallos, int maxFallos,
         string palabraAAdivinar, string palabraParcialmenteAdivinada,
         out string mensajeSiFin)
void Jugar(string palabraAAdivinar, int maximoFallos)
```