Índice

Ejercicios Unidad 11

Descargar estos ejercicios

> **Antes de empezar**
>
> Estos ejercicios están pensados para trabajar de forma avanzada el manejo de cadenas y objetos mutables como `StringBuilder` en C#. Se recomienda revisar los apuntes de la unidad antes de resolverlos. Para realizar estos ejercicios, deberás descargar los recursos del enlace de proyecto_cadenas. Como puedes ver, la solución está compuesta de varios proyectos. Cada uno de ellos corresponde con un ejercicio, deberás implementar todo el código, tanto de la Main como de los métodos que se piden en cada ejercicio. Cada proyecto contiene el test correspondiente, que deberás pasar para comprobar que has hecho el ejercicio correctamente.

Ejercicio 1. Nivel de seguridad de contraseña

Escribe una función que devuelva el nivel de una contraseña teniendo en cuenta que:

- Muy débil: solo números o < 8 caracteres.
- Débil: solo letras y ≥ 8.
- Fuerte: letras y números y ≥ 8.
- Muy fuerte: letras/números + especiales y ≥ 8.

Salida por consola:

```
Introduce una contraseña: 123456
Nivel de seguridad: Muy débil

Introduce una contraseña: abcdabcd
Nivel de seguridad: Débil

Introduce una contraseña: abc12345
Nivel de seguridad: Fuerte

Introduce una contraseña: abc12345!
Nivel de seguridad: Muy fuerte
```

Requisitos:

- Usar métodos de la clase `string` y/o `StringBuilder`.
- Crear los métodos SoloNumeros y otro SoloLetras que retorne un bool según se cumplan las condiciones.
- Crear los método ContieneLetras, ContieneNumeros y ContieneEspeciales que retorne un bool según se cumplan las condiciones.
- Crear un método NivelSeguridad que reciba la contraseña y retorne "Muy débil" , "Débil", "Fuerte" o "Muy fuerte" según el caso. Haz uso de los métodos anteriores para determinar el nivel.
- El programa debe funcionar correctamente con cadenas vacías o nulas.

Ejercicio 2. Palíndromos

Crea una función que determine si una frase es palíndroma, ignorando espacios, signos de puntuación y mayúsculas. No puedes utilizar el método Reverse para invertir la frase.

Salida por consola:

```
Introduce una frase: Anita lava la tina
Es palíndroma

Introduce una frase: Hola mundo
No es palíndroma
```

Requisitos:

- Ignorar espacios, signos de puntuación y mayúsculas.
- No usar el método `Reverse`.
- Usar bucles y métodos de la clase `string`.
- Crea un método EsPalindroma que reciba la cadena y devuelva bool según el caso.

Ejercicio 3. Normalizador con reglas personalizadas

Implementa una función que reciba una cadena y la normalice aplicando las siguientes reglas:

- Sustituir acentos y diéresis por su vocal base.
- Eliminar cualquier carácter no alfabético.
- Pasar todo a minúsculas.

Salida por consola:

```
Introduce una cadena: ¡Árboles y pingüinos!
Normalizada: arbolesypinguinos
```

Requisitos:

- Crear método Normaliza que reciba la cadena y la retorne modificada.
- Usar `StringBuilder` para construir la cadena.
- Sustituir acentos y diéresis por su vocal base.
- Eliminar caracteres no alfabéticos.
- Convertir todo a minúsculas.

Ejercicio 4. Reescritor en código Tiko extendido

Desarrolla una primera versión de un "traductor al lenguaje Tiko extendido" que transforme una frase de entrada aplicando una serie de reglas básicas. En esta fase solo podrás utilizar clases y métodos de manipulación de cadenas (string y StringBuilder), no se permite el uso de expresiones regulares.

El programa debe realizar las siguientes transformaciones:

1. Sustitución de números por texto.Sustituye cada dígito numérico (del 0 al 9) por su nombre equivalente en texto (por ejemplo, "2" se convierte en "dos").
2. Eliminación de letras repetidas consecutivas. Si una misma letra aparece repetida varias veces seguidas, deja solo una. Por ejemplo: "holaaaaaa" se convierte en "hola". Hasta que se

pueda hacer una mejora, esto tendrá el problema de que eliminará letras repetidas correctas (Por ejemplo, perro lo convertirá en pero). En futuras ampliaciones esto quedará corregido.

3. Traducción básica de onomatopeyas. Cambia algunas combinaciones típicas de risa como "jaja", "jeje", "jiji" o "jojo" por emojis sencillos como 😂 , 😄 , 😆 o 🤣 . La sustitución se hará solo si coinciden exactamente esas combinaciones, sin variaciones.Sustituye cada grupo de onomatopeya por un emoji, por ejemplo jaja -> 😂😂

Salida por consola:

```
Introduce una frase: jaja holaaaa 123
Tiko extendido: 😂 hola uno dos tres
```

Requisitos:

- Crear método TraducirTiko que reciba la cadena y la retorne modificada.
- El método TraducirTiko debe hacer uso de los métodos auxiliares SustituyeNumeros, EliminaLetras y TraduceOnomatopeyas.
- Usar solo métodos de `string` y `StringBuilder`.
- No usar expresiones regulares.
- Implementar las tres reglas indicadas.

Ejercicio 5. Juego del ahorcado modular

Desarrolla una versión modular del juego del ahorcado. El programa pedirá al usuario la palabra secreta y el máximo de fallos permitidos. El usuario irá introduciendo letras hasta que acierte la palabra o supere el número de fallos. En cada turno se mostrarán los huecos/aciertos de la palabra y las letras falladas.

Salida por consola:

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


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
```

Requisitos:

- Utiliza `StringBuilder` para la palabra parcialmente adivinada y las letras falladas.
- Todas las letras se tratarán en mayúsculas.
- 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 AñadeLetraALetrasPalabraAMostrar o AñadeLetraALetrasFalladas (aunque se tenga que volver a recorrer las cadenas).
- No se permite usar LINQ ni colecciones dinámicas, solo cadenas y `StringBuilder`.
- Modulariza siguiendo los siguientes métodos para mayor claridad y robustez, sigue el siguiente DEM a la hora de diseñar tus módulos o funciones, podrás ver que no están definidos todos los del DEM, solo es una guía para saber los parámetros de entrada, pero se deberán crear todos los del DEM:

```
string PidePalabraAAdivinar()
int PideMaximoFallos()
bool EstaLetraEnLetras(char letra, string letras)
char PideLetraNoRepetida(string palabraParcialmenteAdivinada, string letrasFalladas)
void MuestraEstadoJuego(string palabraParcialmenteAdivinada, string letrasFalladas)
void AñadeLetraALetrasPalabraAMostrar(string palabraAAdivinar, in char letra, StringB
bool FinDeJuego(int numFallos, int maxFallos, string palabraAAdivinar, string palabra
void Jugar(string palabraAAdivinar, int maximoFallos)
```