Ejercicios Expresiones Regulares

Índice

1. Ejercicio 1
2. Ejercicio 2
3. Ejercicio 3
4. Ejercicio 4
5. ✅ Ejercicio 5
6. ✅ Ejercicio 6
7. ✅ Ejercicio 7
8. Ejercicio 8

Ejercicio 1

Crea expresiones regulares para comprobar el formato de las siguientes entradas.

Nota: Comprobar sólo el formato y no la corrección.

1. Una **fecha larga**. Formato válido DD<sep>MM<sep>AAAA
   <sep> → Separadores ‘ ‘, ‘/‘ o ‘-‘;
   Se permitirán separadores diferentes, en una misma fecha.
2. Un **número entero**.
   Correctos: 12 → +12 → -12 → -0
3. Una **matrícula**. Formatos válidos LL<sep>DDDD<sep>LL y DDDD<sep>LLL
   <sep> → Separadores ‘ ‘ o ‘-‘;
4. Un número **real**, Con la coma Española.
   Correctos: 12 → +12 → -12 → -0 → 12,34 → 0,34 → +0,34 → ,34 → -, 34
5. Un número **real con exponente**.
   Correctos: 12,34E12 → -, 34e-1 → 0,34E+22
6. Un número de **cuenta**. Formato válido
   EEEE<sep>SSSS<sep>C1C2<sep>NNNNNNNNNN
   <sep> → Separadores ‘ ‘ o ‘-‘;
   El numero de cuenta tiene los siguientes subgrupos que deberás etiquetar:
   - E → Dígitos Entidad
   - S → Dígitos Sucursal
   - C1 → Dígito de Control 1
   - C2 → Dígito de Control 2
   - N → Número de cuenta

Ejercicio 2

> Crea la expresión regular para comprobar el formato del **Código de Identificación Fiscal** (C.I.F.)
> Tendrá el siguiente formato: T<sep>PPNNNNN<sep>C
> <sep>: ‘ ‘, ‘-‘ o nada.
>
> - T: Letra de tipo de Organización, una de las siguientes: A, B, C, D, E, F, G, H ,K, L, M, N, P, Q, S.
> - P: Código provincial numérico.
> - N: Numeración secuencial dentro de la provincia.
> - C: Dígito de control, un número ó letra:
>   Aó1, Bó2, Có3, Dó4, Eó5, Fó6, Gó7, Hó8, Ió9, Jó0

Ejercicio 3

> Se nos pide hacer un programa en C# que compruebe el formato de entrada de un **número de cuenta** por teclado, utilizando expresiones regulares (del ejercicio 1 parte 6).
> Además debe indicar, tras la entrada, que dígitos corresponden a la entidad, cuales a la sucursal, los dígitos de control y el número de cuenta, para esto utilizaremos la captura con grupos.
>
> **Opcional:** Puedes comprobar si el número de cuenta es válido calculando los dígitos de control que debería tener, y comprobando si coinciden con los de la introducida. Puedes buscar por Internet como se calcula el dígito de control de una cuenta bancaria.

Ejercicio 4

> Se nos pide hacer un programa en C# que compruebe el formato de entrada y **etiquete cada grupo del C.I.F.**, de una cadena introducida por teclado.

☑ Ejercicio 5

Para el ejercicio anterior calcula, además, el **dígito de control para el C.I.F.**
Deberás comprobar si el C.I.F. es correcto teniendo en cuenta dos cosas: el formato y la corrección del C.I.F. a partir del dígito de control.

Nota: Buscar por Internet como se calcula el dígito de control de un C.I.F.

☑ Ejercicio 6

Escribe un programa con los métodos necesarios para no repetir código, y que sirva para validar las siguientes entradas de texto:

1. Una o más letras sueltas separadas por espacios. Por ejemplo, "a c é" es válida, pero "a c de" no o "a c d " tampoco.
2. Una o más palabras (sólo letras inglesas minúsculas, separadas por uno o varios espacios).
3. una única palabra en mayúsculas.
4. Contraseña (al menos seis caracteres, puede contener letras, números y los caracteres * + . - _, pero no espacios u otros caracteres).

Nota: Salvo que se indique lo contrario, las letras pueden ser minúsculas o mayúsculas. Si el enunciado dice letras inglesas, quiere decir que no se aceptan vocales acentuadas, ñ, ç, etc.

☑ Ejercicio 7

Programa que compruebe con una expresión regular, si un número introducido es un **número complejo**.

Nota: Un numero complejo en su forma binomial se representará como a + bi o a + bj siendo a y b números reales.
Correctos: -2,3 + 5e-2j → 7i → 2E+5 + 2,3i

Ejercicio 8

Expresión regular que encuentre definiciones de **tipos enumerados**, en una cadena de consumo.
Para simplificar supondremos las siguiente sintaxis para los tipos enumerados, respetando espacios:

```
enum Nombre {Valor1,Valor2,Valor3,…,ValorN}
```

...con las siguientes restricciones:

- Las definiciones irán todas en la misma línea, esto es, no hay saltos de línea.
- Todos los textos deben ir en PascalCasing, no pueden comenzar por número y como deben contener una letra.
- No se podrán inicializar los valores enumerados.

Posibles entradas:

- ✔ **`enum Ejemplo {Valor1,Valor2,Valor3}`**
- ✘ **`enum Ejemplo {Valor1,Valor2,Valor3,}`**
- ✘ **`enum Ejemplo{Valor1,Valor2, Valor3}`**
- ✘ **`enum ejemplo {Valor1,valor2,Valor3}`**
- ✘ **`enum 1ejemplo {Valor1,2Valor,Valor3}`**