Ejercicios Expresiones Regulares

Descargar estos ejercicios

Índice

✅ Ejercicio 1

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

1. Una **fecha larga**. Formato válido DD<sep>MM<sep>AAAA
   <sep> → Separadores ‘ ‘, ‘/‘ o ‘-‘;
   Se permitirán separadores diferentes, en una misma fecha.
2. Una **matrícula**. Formatos válidos LL<sep>DDDD<sep>LL y DDDD<sep>LLL
   <sep> → Separadores ‘ ‘ o ‘-‘;
3. Un número **real con exponente**.
   Correctos: 12,34E12 → -, 34e-1 → 0,34E+22

📌 **Nota:** Comprobar sólo el formato y no la corrección. IMPORTANTE, no crear la expresión que solo sea válida para los ejemplos planteados.

✅ Ejercicio 2

Se nos pide hacer un programa en C# que compruebe el formato de entrada de un **número de cuenta** por teclado, utilizando expresiones regulares.
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. Comprueba además, 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 3

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** donde **<sep>** podrá ser **' ', '-' 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, U, V y W**.
- **PP**: Código provincial numérico.
- **NNNNN**: 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 4

Partiendo del **ejercicio 3 amplíalo** definiendo un método que extraiga los grupos de datos que componen el CIF y 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.

Las operaciones para calcular este dígito de control se realizan sobre **los siete dígitos centrales** y son las siguientes:

1. Sumar los dígitos de las **posiciones pares**. Suma = **A**
2. Para cada uno de los dígitos de las **posiciones impares**, multiplicarlo por 2 y sumar los dígitos del resultado.
   Ej: **8 * 2 = 16 → 1 + 6 = 7**
   Acumular el resultado. Suma = **B**
3. Calcular la suma **A + B = C**
4. Tomar sólo el **dígito de las unidades** de **C** y restárselo a **10**. Esta resta nos da **D**.
5. A partir de **D** ya se obtiene el dígito de control.
   Si ha de ser **numérico** es directamente **D** y si se trata de una letra se realizará la siguiente correspondencia: **A = 1, B = 2, C = 3, D = 4, E = 5, F = 6, G = 7, H = 8, I = 9, J = 10 ó 0**
   Deberemos tener en cuenta pues, que las **organizaciones** con la letra **N, P, Q, R, S o W** tendrán como dígito de control una letra.

Ejemplo: para el C.I.F.: **A58818501**. Utilizamos los siete dígitos centrales = 5881850

1. Sumamos los dígitos pares: **A** = 8 + 1 + 5 = **14**
2. Posiciones impares:
   5 * 2 = 10 → 1 + 0 = **1**
   8 * 2 = 16 → 1 + 6 = **7**
   8 * 2 = 16 → 1 + 6 = **7**
   0 * 2 = **0**
   Sumamos los resultados: **B** = 1 + 7 + 7 + 0 = **15**
3. Suma parcial: **C** = A + B = 14 + 15 = **29**
4. El dígito de las unidades de C es 9. Se lo restamos a 10 y nos da: **D** = 10 - 9 = **1**
5. Como el tipo de organización es **A** será un número **1**, en caso de haber sido letra el DC hubiera sido la letra **A** también.

Ejercicio 5

> 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 6

> Programa que compruebe con una expresión regular, si un número introducido es un **número complejo**.
>
> 1. Para que hacer este ejercicio, deberías usar el siguiente la patrón para identificar un número real.
>
>    ```
>    string patrolReal = @"[+-]?((\d+)|(\d*[.,]\d+))([eE][+-]?\d+)?";
>    ```
> 2. Un numero complejo en su forma binomial se representará como `a + bi` o `a + bj` siendo `a` y `b` números reales.
>
> **Ejemplos** de de entrada de números reales correctos:
>
> ```
> -2,3 + 5e-2j
> 7i
> 2E+5 + 2,3i
> 3 - 5i
> ```
>
> ✋ **Importante:** Para ser considerado el ejercicio como correcto se debería dividir el patrón en varias partes evitando repetir el patrón de número real del ejercicio 1.

Ejercicio 7

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}`**