**EXPRESIONES REGULARES EN PROGRAMACIÓN**

Iván Espinosa Fourquet

Raúl García Pablos

Ruth Hernández Quitián

1ºC DAM

Contenido

[Ejercicio 1 3](#_Toc101262716)

[Ejercicio 2 3](#_Toc101262717)

[Ejercicio 3 3](#_Toc101262718)

[Ejercicio 4 3](#_Toc101262719)

[Ejercicio 5 3](#_Toc101262720)

[Ejercicio 6 3](#_Toc101262721)

[Ejercicio 7 4](#_Toc101262722)

[Ejercicio 8 5](#_Toc101262723)

[Ejercicio 9 5](#_Toc101262724)

[Ejercicio 10 6](#_Toc101262725)

[Referencias 8](#_Toc101262726)

# Ejercicio 1

1. Crea una expresión regular que sirva para validar un DNI
2. **Solución/Expresión regular**:

**^[0-9]{8}[T|R|W|A|G|M|Y|F|P|D|X|B|N|J|Z|S|Q|V|H|L|C|K|E]$**

**[0-9]{8}** 🡪 8 dígitos del 0 al 9.

**[T|R|W|A|G|M|Y|F|P|D|X|B|N|J|Z|S|Q|V|H|L|C|K|E]** 🡪 Una de la letras posibles del DNI.

# Ejercicio 2

1. Crea una expresión regular que sirva para validar una fecha (dd/mm/aaa)
2. **Solución/Expresión regular**:

**^(0?[1-9]|[12][0-9]|[3][01])[\\/](0?[1-9]|1[012])[\\/]([12][0-9])\\d{2}$**

**0?** 🡪 Tenemos en cuenta si la fecha tiene o no un 0 delante de los 01,02,03...

**[1-9]|[12][0-9]|[3][01]** 🡪 Un número del 1 al 31.

**[\\/]** 🡪 Escapamos con doble barra atrás la barra del formato fecha.

**(0?[1-9]|1[012])** 🡪 Meses desde el 1 o 01 al 12.

**([12][0-9])\\d{2}** 🡪 Año desde el 1000 al 2999 (10 al 29 más dos dígitos cualquiera).

# Ejercicio 3

1. Crea una expresión regular que sirva para validar un número real
2. **Solución/Expresión regular**:

**^-?[0-9]+([\\.,][0-9]+)?$**

**-?** 🡪 Tenemos en cuenta si el número va a tener o no signo negativo.

**[0-9]+** 🡪 Uno o más números del 0 al 9. Después de la coma lo mismo pero con el símbolo de interrogación por si el número tiene decimales o no.

**[\\.,]** 🡪 Escapamos con doble barra atrás el punto (símbolo especial) y añadimos la coma para tenerla en cuenta también.

# Ejercicio 4

1. Crea una expresión regular que sirva para buscar en el texto todas las fechas (“día” de “mes” de “año”)

Texto: [cuenta fechas BOE.txt](cuenta%20fechas%20BOE.txt)

**Expresión regular**:

1. **([1-9]|[12][0-9]|[3][01])\sde\s(enero|febrero|marzo|abril|mayo| junio|julio|agosto|septiembre|octubre|noviembre|diciembre) de ([12][0-9])\\d{2})**

**([1-9]|[12][0-9]|[3][01])** 🡪 Número del mes del 1 al 31.

**\sde\s ó de** 🡪 Los espacios podemos introducirlos con \s o directamente el espacio en blanco. En esta expresión he usado las dos formas.

**mes1|mes2|…** 🡪 Introducimos entre tuberías los meses correspondientes que queremos en la expresión regular.

**([12][0-9])\\d{2}** 🡪 El año XXXX en este caso de 1000 al 2999.

**Resultado**:

Fechas encontradas: 45

# Ejercicio 5

1. Crea una expresión regular que sirva para buscar en el texto todas las palabras acentuadas

Texto: [cuenta fechas BOE.txt](cuenta%20fechas%20BOE.txt)

**Expresión regular**:

**[A-Za-z]([a-z]+)?(á|é|í|ó|ú)([a-z]+)?**

**[A-Za-z]** 🡪 Que la palabra empiece por mayúscula o minúscula (por hacer la expresión más simple no tendré en cuenta palabras acentuadas en su primera letra [ÁÉÍÓÚ]).

**([a-z]+)?** 🡪 Tanto antes o después de una vocal acentuada (esta expresión solo sirve para palabras del español) podrá haber o no una o más vocales tanto delante como detrás.

**Resultado**:

Palabras acentuadas encontradas: 135

# Ejercicio 6

1. Dado el siguiente listado de nombres de usuarios, habrá que contabilizar cuántos de ellos son válidos. Cada elemento del listado estará separado del siguiente por un punto y coma y un salto de línea.

Los nombres de usuario válidos tendrán las siguientes características:

* Los nombres de usuario estarán compuestos por caracteres alfanuméricos.
* También estará permitido el uso de la barra baja (\_) y el guion (-).
* El nombre de usuario tendrá que empezar y terminar por un carácter alfanumérico.
* El nombre de usuario tendrá que tener entre 5 y 20 caracteres.

Texto: [usuarios regex.txt](usuarios%20regex.txt)

**Expresión regular**:

**(^|\\n)[a-zA-Z0-9]([\_-]|[a-zA-Z0-9]){3,18}[a-zA-Z0-9](;|$)**

**(^|\\n)** 🡪 Nos aseguramos de coger el primer elemento del listado, el resto de elementos darán comienzo en el salto de línea.

**[a-zA-Z0-9]** 🡪 El primer carácter ha de ser un carácter alfanumérico.

**([\_-]|[a-zA-Z0-9]){3,18}** 🡪 Los siguientes caracteres han de ser caracteres alfanuméricos o bien un guion o una barra baja. Habrá un número variable de caracteres entre 3 y 18, para ajustarlo a lo indicado teniendo en cuenta que el primer y el último se determinan de forma independiente.

**[a-zA-Z0-9]** 🡪 El último carácter ha de ser un carácter alfanumérico.

**(;|$)**🡪 Nos aseguramos de coger el último elemento del listado y de dar por terminado el resto de elementos al llegar al punto y coma separador.

**Resultado**:

Hay 49 nombres de usuario válidos en el listado.

# Ejercicio 7

1. Dado el siguiente listado de contraseñas, contabilizar cuántas de ellas son seguras. Cada elemento del listado estará separado del siguiente por un punto y coma y un salto de línea.

Las contraseñas seguras tendrán las siguientes características:

* Al menos un número.
* Al menos una letra minúscula.
* Al menos una letra mayúscula.
* Al menos uno de los caracteres especiales [\*.!@#$%^&(){}[]:;<>,.?/~\_+-=|\]
* Al menos 8 caracteres, pero menos de 32.

Texto: [contraseñas regex.txt](contraseñas%20regex.txt)

**Expresión regular**:h

**(^|\\n)(?=.\*[0-9])(?=.\*[a-z])(?=.\*[A-Z])(?=.\*[\*.!@$%^&(){}\\[\\]:;<>,.?/~\_+-=|\\\\]).{8,32}(;|$)**

**(^|\\n)** 🡪 Nos aseguramos de coger el primer elemento del listado, el resto de elementos darán comienzo en el salto de línea.

**(?=.\*[0-9])** 🡪 Al menos un carácter ha de ser un dígito.

**(?=.\*[a-z])** 🡪 Al menos un carácter ha de ser una letra minúscula.

**(?=.\*[A-Z])** 🡪 Al menos un carácter ha de ser una letra mayúscula.

**(?=.\*[\*.!@$%^&(){}\\[\\]:;<>,.?/~\_+-=|\\\\]).** 🡪 Al menos un carácter ha de ser un carácter especial.

.**{8,32}** 🡪 Cualquier carácter se repetirá entre 8 y 32 veces

**(;|$)**🡪 Nos aseguramos de coger el último elemento del listado y de dar por terminado el resto de elementos al llegar al punto y coma separador.

**Resultado**:

Hay 5 contraseñas seguras en el listado.

# Ejercicio 8

1. En el siguiente texto, contabilizar cuántas veces aparece una palabra con la misma raíz que la palabra “oler” (¡tener en cuenta que es un verbo irregular!).

**Expresión regular**:

**"(^|\\W)(ol|huel|Ol|Huel)([a-z]+)(**[**\\W|$**](file:///\\W|$)**)"**

**(^|\\W)** 🡪 Nos aseguramos de coger la primera palabra de la línea, el resto de palabras estarán siempre precedidas por algún signo de puntuación, espaciados…

**(ol|huel|Ol|Huel)**🡪 Insertamos la raíz de la palabra, al ser irregular tendrá dos posibles formas, tenemos en cuenta que puede aparecer en mayúscula y también sería válida.

**([a-z]+)** 🡪 La raíz irá seguida de una o más letras minúsculas.

**(**[**\\W|$**](file:///\\W|$)**)**🡪 Nos aseguramos de coger la última palabra de la línea, el resto de palabras finalizarán al aparecer algún signo de puntuación, espaciados…

**Resultado:**

Hay 21 palabras que comparten raíz con la palabra “oler” en el texto dado.

# Ejercicio 9

1. Contabilizar el número de palabras sin repetir que aparecen en la búsqueda del ejercicio anterior y elaborar una lista con las mismas.

**Resultado:**

Hay 7 palabras en total que comparten raíz con la palabra “oler” si eliminamos las repetidas. Son las siguientes:

1. oler

2. olores

3. olor

4. huelo

5. huele

6. olfato

7. oliendo

# Ejercicio 10

1. En el siguiente texto, extraer el precio total de cada una de las facturas.
2. Contabilizar los ingresos totales en cada una de las divisas (dólar $, euro €, libra esterlina £). ¿Cuál de los tres mercados es el más rentable?
3. Contabilizar el número de operaciones realizadas en cada una de las divisas.

Texto: [Regex Factura.xlsx](Regex%20Factura.xlsx)

**Expresión regular**:

Dólar:

**"\\$[0-9]\*,[0-9]\*\\r"**

**\\$**🡪 El símbolo dólar aparece antes de la cantidad, nos aseguramos de escapar el símbolo.

**[0-9]\*,**🡪 Habrá una cantidad de dígitos del 0 al 9 variable, y después la coma decimal.

**[0-9]\*** 🡪 Habrá una cantidad de dígitos del 0 al 9 variable tras la coma decimal.

[**\\r**](file:///\\W|$)🡪 Todas las cantidades estarán precediendo a un retorno de carro.

Euro:

**"€\\s[0-9]\*,[0-9]\*\\r"**

**€\\s**🡪 El símbolo euro aparece antes de la cantidad separado por un espacio.

**[0-9]\*,**🡪 Habrá una cantidad de dígitos del 0 al 9 variable, y después la coma decimal.

**[0-9]\*** 🡪 Habrá una cantidad de dígitos del 0 al 9 variable tras la coma decimal.

[**\\r**](file:///\\W|$)🡪 Todas las cantidades estarán precediendo a un retorno de carro.

Libra esterlina:

**"£[0-9]\*,[0-9]\*\\r"**

**£**🡪 El símbolo de la libra aparece antes de la cantidad.

**[0-9]\*,**🡪 Habrá una cantidad de dígitos del 0 al 9 variable, y después la coma decimal.

**[0-9]\*** 🡪 Habrá una cantidad de dígitos del 0 al 9 variable tras la coma decimal.

[**\\r**](file:///\\W|$)🡪 Todas las cantidades estarán precediendo a un retorno de carro.

**Resultado apartado A:**

El euro es el mercado más rentable.

Total ingresado en euros: 98126.04999999997 €.

Total ingresado en dólares: 91325.85999999997 $.

Total ingresado en libras: 16422.260000000002 £.

**Resultado apartado B:**

Hay 282 operaciones realizadas en dólares.

Hay 288 operaciones realizadas en euros.

Hay 47 operaciones realizadas en libras.

# Ejercicio 11

# Dada una lista de matriculas de coches, debes realizar un programa que identifique, cuente, guarde e imprima una lista con las matrículas Modernas (tipo 0000 XXX), las antiguas (tipo X-0000-XX) y una lista con todo lo que no sean matírculas.

**Solución/Expresión regular**:

**"^[0-9]{4}\s[A-Z]{3}$"**

Busca 4 caracteres numéricos, un espacio y tres letras mayúsculas al final.

**"^[A-Z]{1}\s[0-9]{4}\s[A-Z]{2}$"**

Busca una letra mayúscula, un espacio, 4 caracteres numéricos, otro espacio y dos letras mayúsculas al final.

# Ejercicio 12

Cogiendo el texto que quieras o el que se proporciona como ejemplo, desarrolla un programa que cuente cuantas palabras tiene y cuente cuantas veces se repiten. Se debe imprimir esa información.

**Solución/Expresión regular**:

**"\\s+"**

Busca los espacios en blanco entre palabras, con la función Split puedes separar el texto en varias palabras para contarlas.

# Ejercicio 13

Crea un programa en el que puedas meter una dirección de mail y te diga si es correcta o no.

**Solución/Expresión regular**:

**"^([\\w-]+\\.)\*?[\\w-]+@[\\w-]+\\.([\\w-]+\\.)\*?[\\w]+$"**

Busca primero cualquier carácter alfanumérico, después el @ seguido de más caracteres alfanuméricos seguidos de un punto y acabados en letras.

# Ejercicio 14

Intenta desarrollar el programa de censura de chat de videojuegos como el Lol. Se pedirán mensajes por Scanner y el programa detectará si se trata de malas palabras o insultos y los sustituirá por asteriscos (puta --> \*\*\*\* , gilipollas --> \*\*\*\*\*\*\*\*\*\*). Una vez comprobado el mensaje y censurado si hace falta se imprimirá en nuestro "chat".

**Solución/Expresión regular**:

**"put.|put..|coño|gilipoll..|cancer|nigga|nigger|subnormal|polla|retrasado|retard|sida|tumor"**

Aquí simplemente es meter una serie de insultos separados por el operador OR (“|”).

# Ejercicio 15

Crea un programa que genere combinaciones aleatorias de 4 letras y que no pare de generarlas hasta que por azar del universo se cree la palabra "hola", una vez se cree el programa parará y te mostrará cuantos intentos ha necesitado. Pista: Usa un long para el contador porque van a ser muchos intentos.

**Solución/Expresión regular**:

**"hola"**

Simplemente es meter la palabra que quieras buscar, la dificultad radica en la lógica.

# Referencias

<https://www.ocpsoft.org/tutorials/regular-expressions/password-regular-expression/>

<https://mkyong.com/regular-expressions/how-to-validate-username-with-regular-expression/#:~:text=Username%20consists%20of%20alphanumeric%20characters,the%20first%20or%20last%20character.&text=The%20number%20of%20characters%20must%20be%20between%205%20to%2020>.

<https://dle.rae.es/oler?m=form>

<https://dle.rae.es/olor?m=form>

<https://www.boe.es/boe/dias/1998/04/28/>

<https://www.youtube.com/channel/UCEzsgYrYzOCOcwmjhu_pngw>