

# Práctica 03: Expresiones regulares

Computabilidad y Algoritmia  
Grado en Ingeniería Informática  
Universidad de La Laguna

Alejandro Rodríguez Rojas  
[alu0101317038@ull.edu.es](mailto:alu0101317038@ull.edu.es)

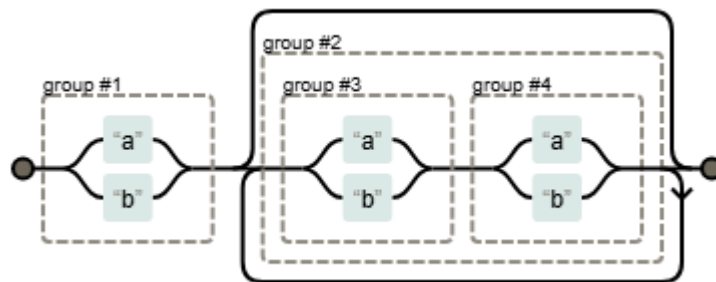
# Índice

<b>1. Ejercicios sobre operadores básicos</b>	<b>3</b>
1.1. Cadenas sobre el alfabeto {a, b} con longitud impar	3
1.2. Cadenas sobre el alfabeto {a, b} con longitud igual a 5.	4
1.3. Cadenas sobre el alfabeto {a, b, c} con una "a" en la antepenúltima posición.	5
1.4. Cadenas sobre el alfabeto {a, b} con número de "a's" par o número de "b's" impar.	6
1.5. Cadenas w sobre el alfabeto {0, 1} tales que $2 \leq  w  \leq 5$	7
1.6. Cadenas sobre el alfabeto {0, 1} con longitud múltiplo de 3	8
1.7. Cadenas sobre el alfabeto {0, 1} con una longitud que no sea múltiplo de 3.	9
1.8. Cadenas w sobre el alfabeto {0, 1} tal que $w = 0^n 1^m$ con $n + m$ impar.	10
1.9. Cadenas sobre el alfabeto {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} que tengan como máximo dos ceros.	11
1.10. Cadenas sobre el alfabeto {x, y, z} que no contengan dos símbolos x consecutivos.	13
<b>2. Ejercicios sobre operadores extendidos</b>	<b>14</b>
2.1. Direcciones de correos electrónicos de estudiantes de la Universidad de La Laguna.	14
2.2. Palabras que terminen por una vocal.	15
2.3. Números enteros.	15
2.4. Texto que se encuentre entre paréntesis.	16
2.5. Código postal en España.	17
2.6. Palabras que contienen sólo letras mayúsculas.	17
2.7. Números de teléfono en formato prefijo XXX-XXX-XXX, donde el prefijo del país puede indicarse empezando por 00 o bien con un símbolo +; por ejemplo, 0034 o +34 para España.	18
2.8. Fecha en formato DD/MM/AAAA	18
2.9. Palabras de al menos 10 letras de longitud.	19
2.10. Palabras que terminen con "ing" o "ed".	19
<b>3. Modificación</b>	<b>20</b>
3.1. Sentencias class en C++	20
3.2. Lenguaje binario con número par de 1's	20

# 1. Ejercicios sobre operadores básicos

## 1.1. Cadenas sobre el alfabeto {a, b} con longitud impar

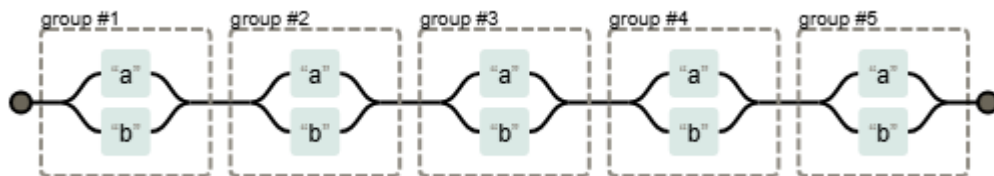
- Expresión regular:  $(a|b)((a|b)(a|b))^*$



- Cadenas que pertenecen al lenguaje:
  - $w_1 = a$
  - $w_2 = aba$
  - $w_3 = babaa$
  - $w_4 = aaaaaaa$
  - $w_5 = bbabbba$
- Cadenas que no pertenecen al lenguaje:
  - $w_6 = ba$
  - $w_7 = abab$
  - $w_8 = bbaabb$
  - $w_9 = aa$
  - $w_{10} = bababa$

## 1.2. Cadenas sobre el alfabeto {a, b} con longitud igual a 5.

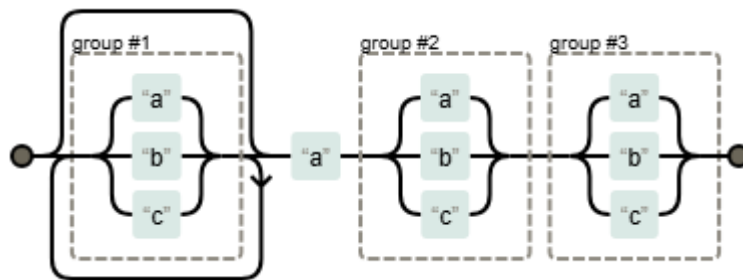
- Expresión regular:  $(a|b)(a|b)(a|b)(a|b)(a|b)$



- Cadenas que pertenecen al lenguaje:
  - $w_1 = aaaaa$
  - $w_2 = ababa$
  - $w_3 = bbabb$
  - $w_4 = bbbbb$
  - $w_5 = aabba$
- Cadenas que no pertenecen al lenguaje:
  - $w_6 = a$
  - $w_7 = bb$
  - $w_8 = bbb$
  - $w_9 = ababababab$
  - $w_{10} = aabb$

1.3. Cadenas sobre el alfabeto {a, b, c} con una “a” en la antepenúltima posición.

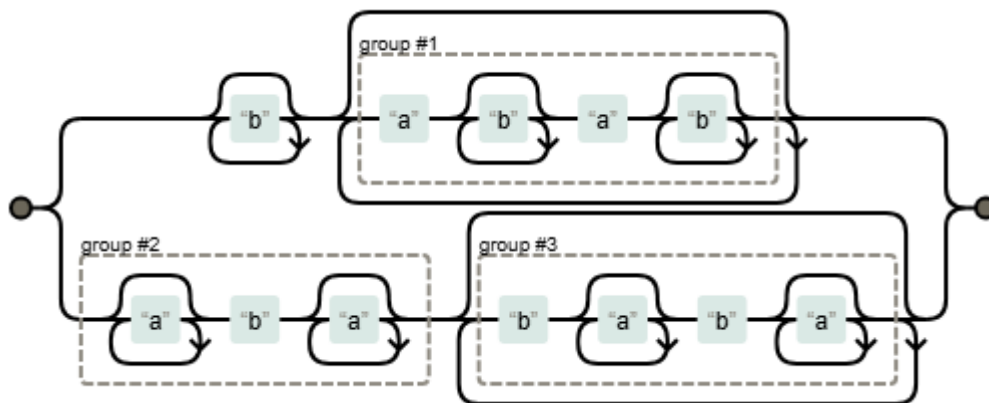
- Expresión regular:  $(a|b|c)^* a(a|b|c)(a|b|c)$



- Cadenas que pertenecen al lenguaje:
  - $w_1 = abc$
  - $w_2 = abbabb$
  - $w_3 = cabb$
  - $w_4 = aaa$
  - $w_5 = bacc$
- Cadenas que no pertenecen al lenguaje:
  - $w_6 = ccc$
  - $w_7 = bca$
  - $w_8 = a$
  - $w_9 = bbbca$
  - $w_{10} = aacccc$

#### 1.4. Cadenas sobre el alfabeto {a, b} con número de “a’s” par o número de “b’s” impar.

- Expresión regular:  $b^*(ab^*ab^*)^* \mid (a^*ba^*)(ba^*ba^*)^*$

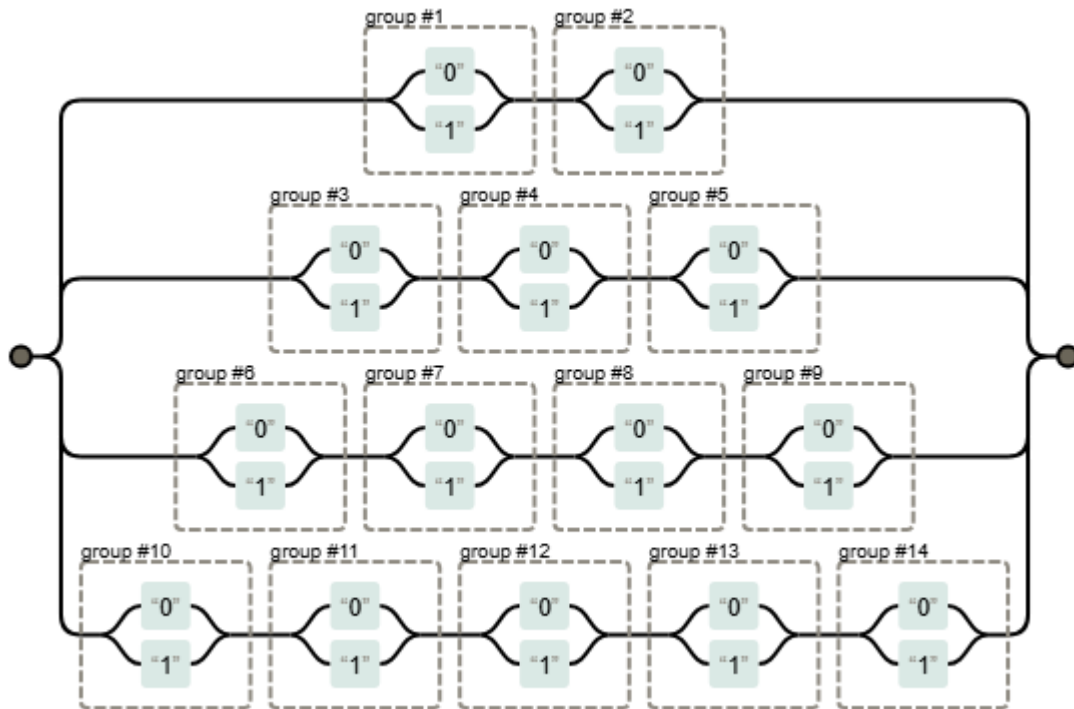


- Cadenas que pertenecen al lenguaje:
  - $w_1 = baa$
  - $w_2 = aabb$
  - $w_3 = bbba$
  - $w_4 = aa$
  - $w_5 = b$
- Cadenas que no pertenecen al lenguaje:
  - $w_6 = a$
  - $w_7 = bb$
  - $w_8 = abb$
  - $w_9 = aaaaabbbbbbb$
  - $w_{10} = bbbaabb$

### 1.5. Cadenas $w$ sobre el alfabeto $\{0, 1\}$ tales que $2 \leq |w| \leq 5$

- Expresión regular:

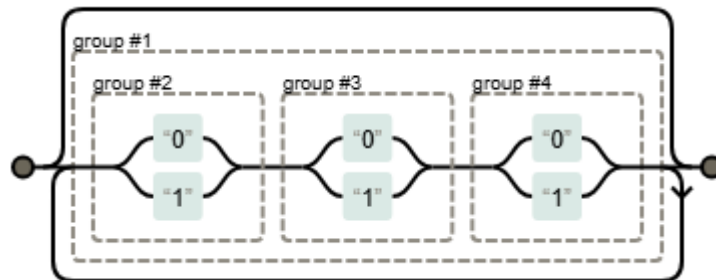
$(0|1)(0|1) \mid (0|1)(0|1)(0|1) \mid (0|1)(0|1)(0|1)(0|1) \mid (0|1)(0|1)(0|1)(0|1)(0|1)$



- Cadenas que pertenecen al lenguaje:
  - $w_1 = 01$
  - $w_2 = 111$
  - $w_3 = 0011$
  - $w_4 = 001$
  - $w_5 = 10011$
- Cadenas que no pertenecen al lenguaje:
  - $w_6 = 1$
  - $w_7 = 0$
  - $w_8 = 1111110000$
  - $w_9 = 0000000000$
  - $w_{10} = 101010101010110101010$

## 1.6. Cadenas sobre el alfabeto {0, 1} con longitud múltiplo de 3

- Expresión regular:  $((0|1)(0|1)(0|1))^*$

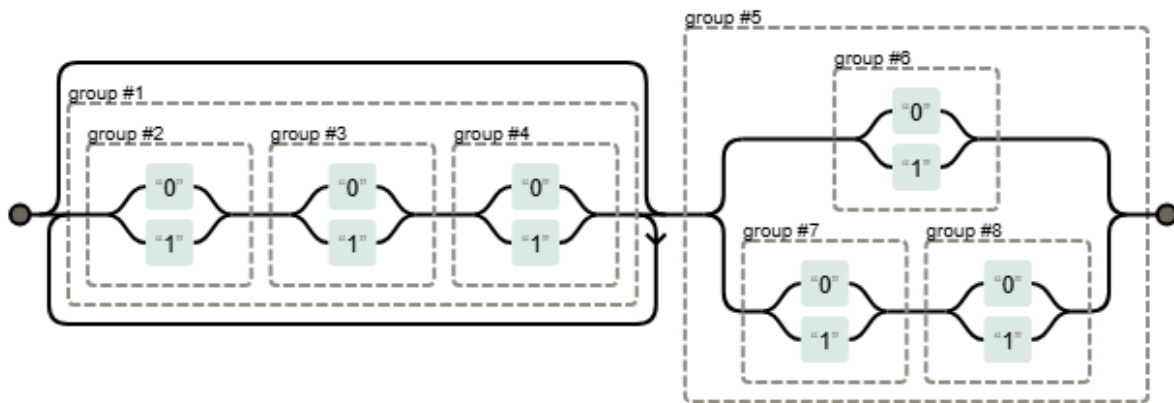


- Cadenas que pertenecen al lenguaje:
  - $w_1 = 000$
  - $w_2 = 111000$
  - $w_3 = 101010101$
  - $w_4 = 001100111000$
  - $w_5 = 110$
- Cadenas que no pertenecen al lenguaje:
  - $w_6 = 1$
  - $w_7 = 11$
  - $w_8 = 1111$
  - $w_9 = 0110011$
  - $w_{10} = 0111$



1.7. Cadenas sobre el alfabeto  $\{0, 1\}$  con una longitud que no sea múltiplo de 3.

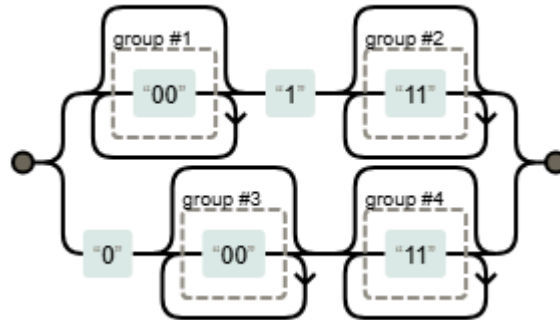
- Expresión regular:  $((0|1)(0|1)(0|1))^* ((0|1)|(0|1)(0|1))$



- Cadenas que pertenecen al lenguaje:
  - $w_1 = 0$
  - $w_2 = 1$
  - $w_3 = 11$
  - $w_4 = 0011$
  - $w_5 = 0000110$
- Cadenas que no pertenecen al lenguaje:
  - $w_6 = 000$
  - $w_7 = 101010$
  - $w_8 = 11111111$
  - $w_9 = 111$
  - $w_{10} = 101$

1.8. Cadenas  $w$  sobre el alfabeto  $\{0, 1\}$  tal que  $w = 0^n 1^m$  con  $n + m$  impar.

- Expresión regular:  $(00)^*1(11)^* \mid 0(00)^*(11)^*$

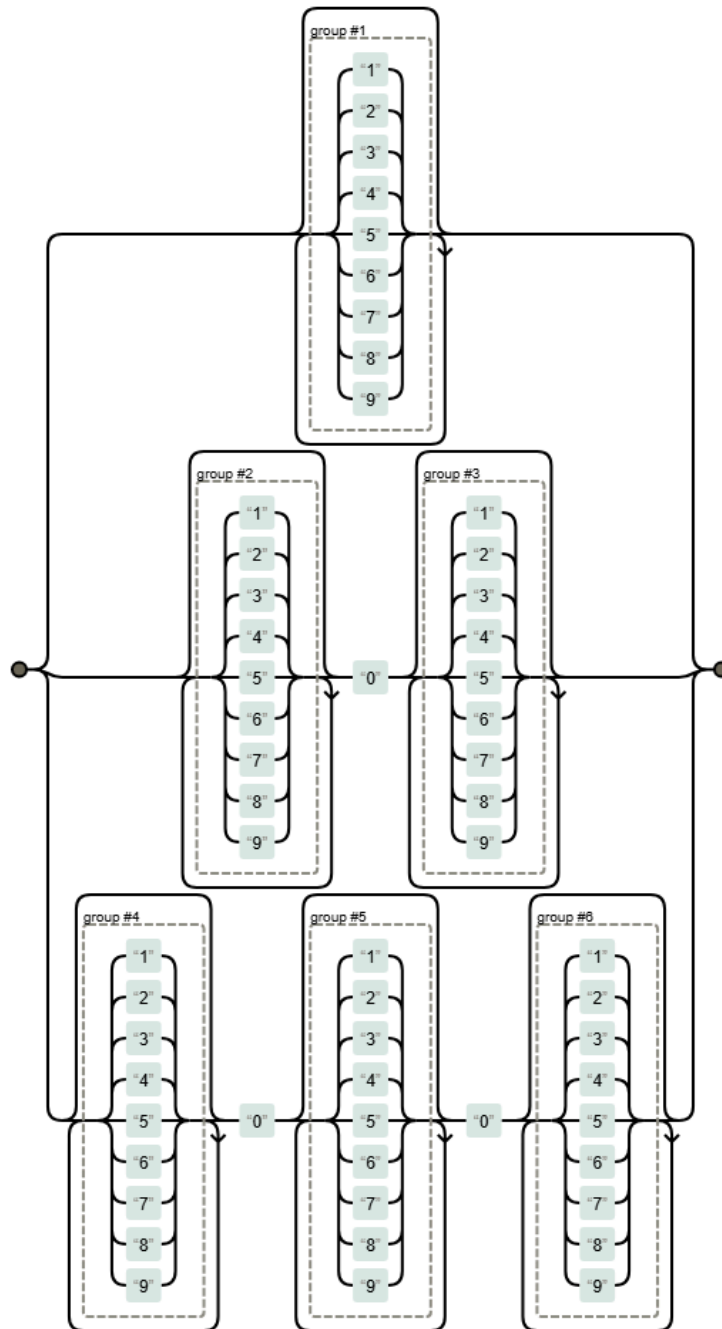


- Cadenas que pertenecen al lenguaje:
  - $w_1 = 0$
  - $w_2 = 1$
  - $w_3 = 0001111$
  - $w_4 = 00111$
  - $w_5 = 011$
- Cadenas que no pertenecen al lenguaje:
  - $w_6 = 0011$
  - $w_7 = 0111$
  - $w_8 = 0101$
  - $w_9 = 1111$
  - $w_{10} = 001111$

1.9. Cadenas sobre el alfabeto  $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$  que tengan como máximo dos ceros.

- Expresión regular:

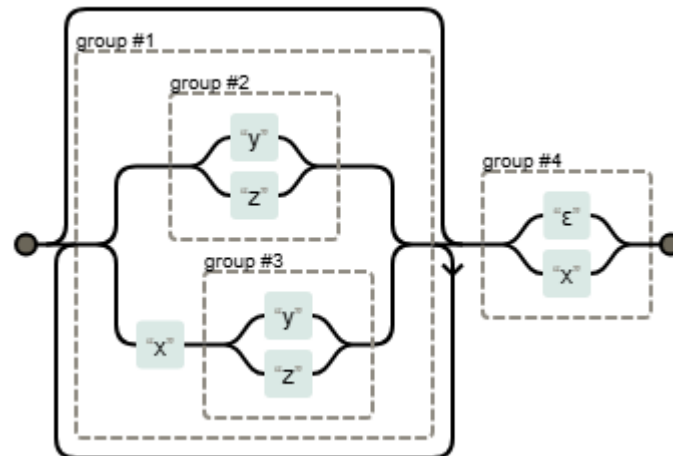
$(1|2|3|4|5|6|7|8|9)^* | (1|2|3|4|5|6|7|8|9)^* 0 (1|2|3|4|5|6|7|8|9)^* |$   
 $(1|2|3|4|5|6|7|8|9)^* 0 (1|2|3|4|5|6|7|8|9)^* 0 (1|2|3|4|5|6|7|8|9)^*$



- Cadenas que pertenecen al lenguaje:
  - $w_1 = 12345$
  - $w_2 = 54216023$
  - $w_3 = 00$
  - $w_4 = 10103234$
  - $w_5 = 4443$
  
- Cadenas que no pertenecen al lenguaje:
  - $w_6 = 000$
  - $w_7 = 800709$
  - $w_8 = 02220220$
  - $w_9 = 01001$
  - $w_{10} = 003333300$

1.10. Cadenas sobre el alfabeto {x, y, z} que no contengan dos símbolos x consecutivos.

- Expresión regular:  $((y|z)|x(y|z))^* (\epsilon | x)$

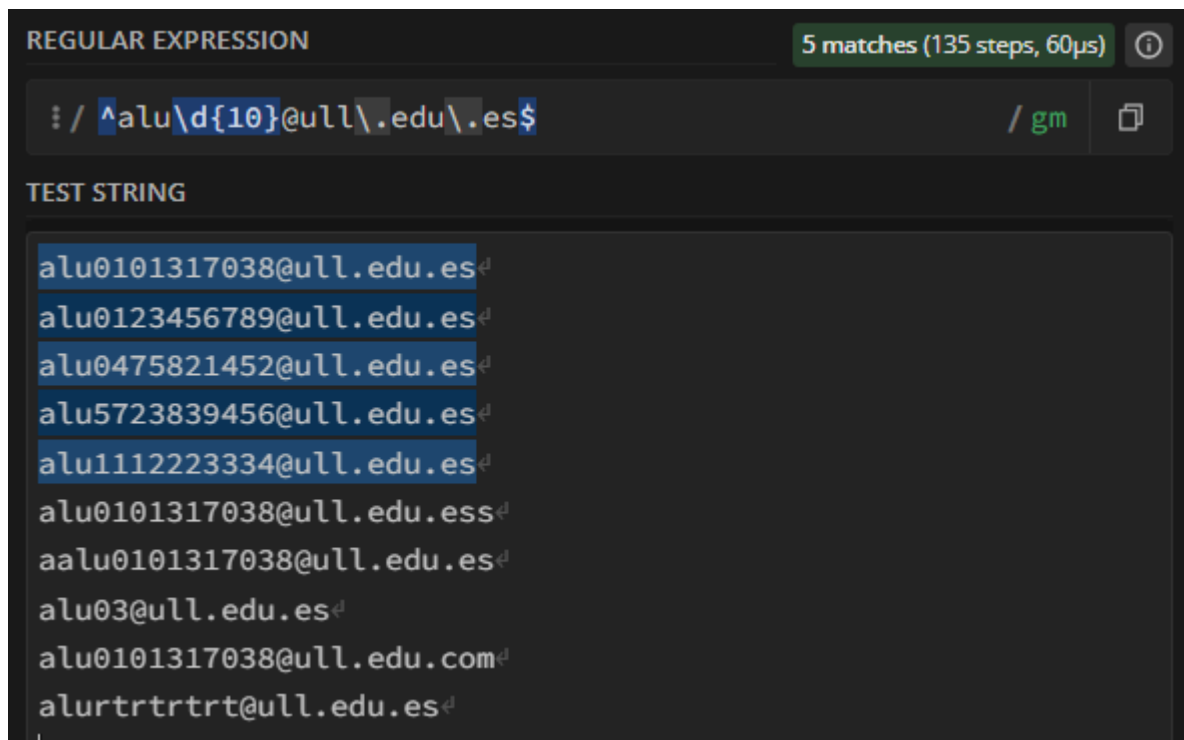


- Cadenas que pertenecen al lenguaje:
  - $w_1 = yxz$
  - $w_2 = zxyxz$
  - $w_3 = zzzxzzz$
  - $w_4 = yxzx$
  - $w_5 = zxz$
- Cadenas que no pertenecen al lenguaje:
  - $w_6 = xx$
  - $w_7 = zyxxxy$
  - $w_8 = zyyzxx$
  - $w_9 = xxyzx$
  - $w_{10} = zxxz$

## 2. Ejercicios sobre operadores extendidos

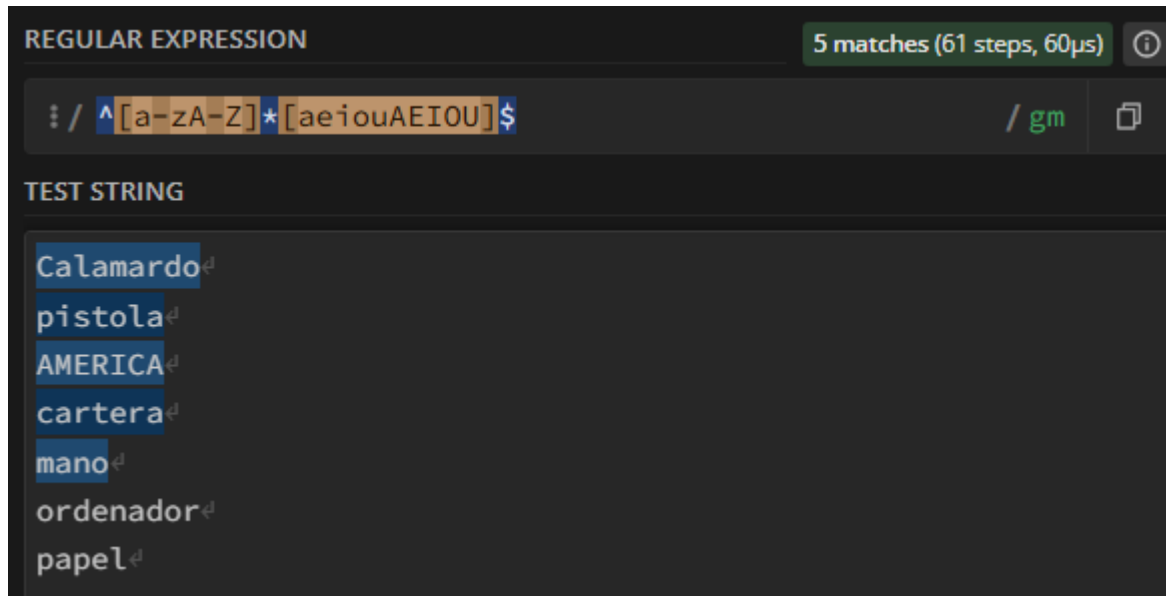
### 2.1. Direcciones de correos electrónicos de estudiantes de la Universidad de La Laguna.

- Expresión regular: `^alu\d{10}@ull\.edu\.es$`



## 2.2. Palabras que terminen por una vocal.

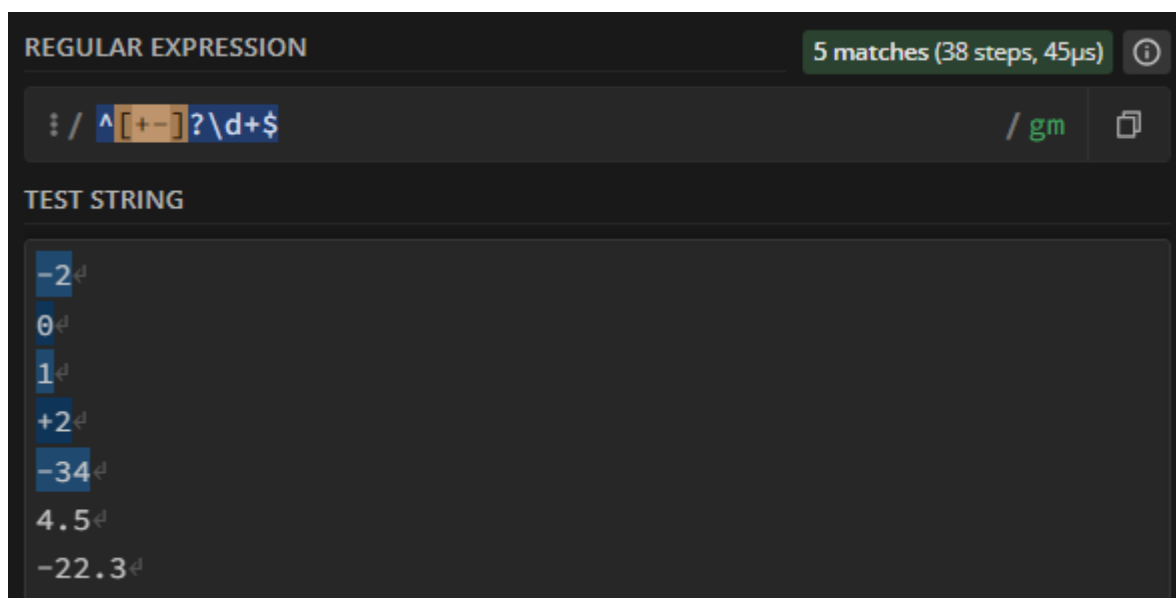
- Expresión regular: `^[a-zA-Z]*[aeiouAEIOU]$`



The screenshot shows a regular expression tester interface. At the top, the title is "REGULAR EXPRESSION" and the status bar indicates "5 matches (61 steps, 60µs)". The regular expression pattern is `^[a-zA-Z]*[aeiouAEIOU]$`. Below the pattern, the "TEST STRING" section lists several words: "Calamardo", "pistola", "AMERICA", "cartera", "mano", "ordenador", and "papel". The words "Calamardo", "pistola", "AMERICA", "cartera", and "mano" are highlighted in blue, indicating they match the pattern. The words "ordenador" and "papel" are not highlighted, indicating they do not match the pattern.

## 2.3. Números enteros.

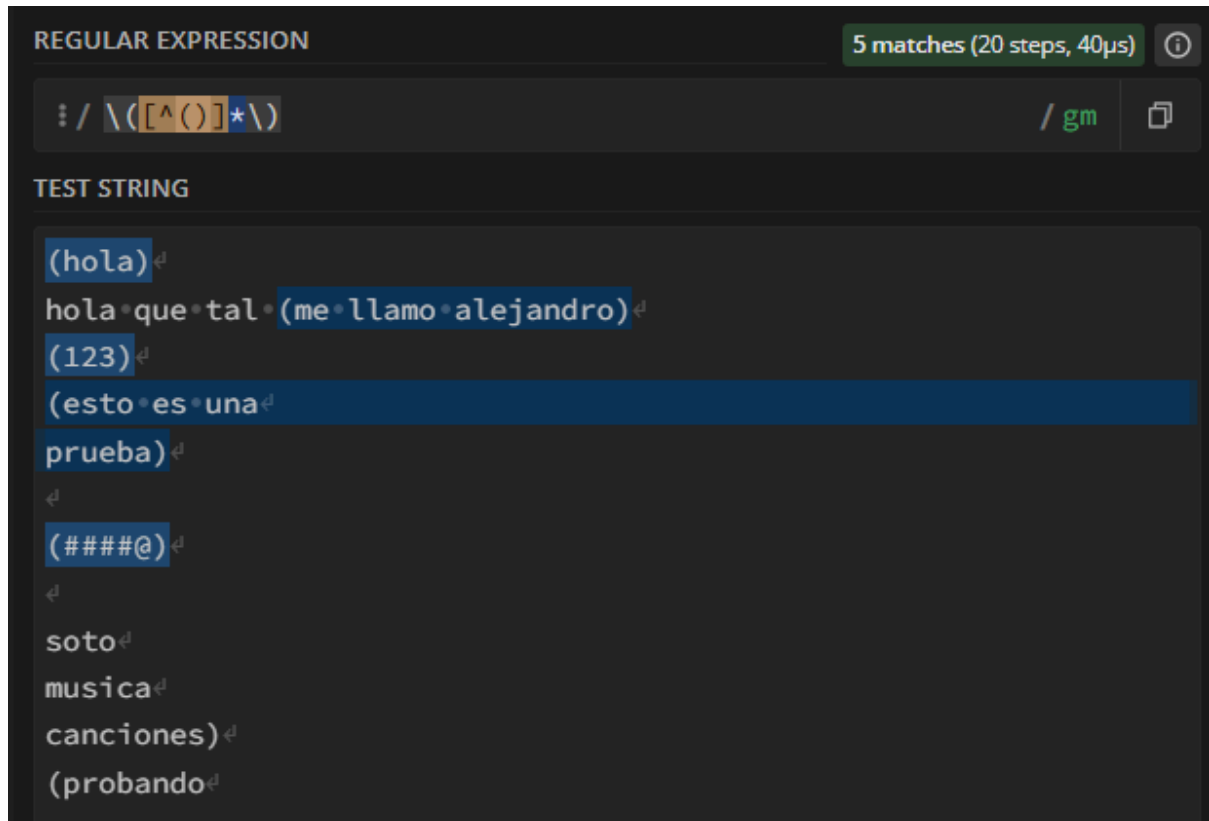
- Expresión regular: `^[+-]?\d+$`



The screenshot shows a regular expression tester interface. At the top, the title is "REGULAR EXPRESSION" and the status bar indicates "5 matches (38 steps, 45µs)". The regular expression pattern is `^[+-]?\d+$`. Below the pattern, the "TEST STRING" section lists several numbers: "-2", "0", "1", "+2", "-34", "4.5", and "-22.3". The numbers "-2", "0", "1", "+2", and "-34" are highlighted in blue, indicating they match the pattern. The numbers "4.5" and "-22.3" are not highlighted, indicating they do not match the pattern.

## 2.4. Texto que se encuentre entre paréntesis.

- Expresión regular: `\([^\)]*\)`

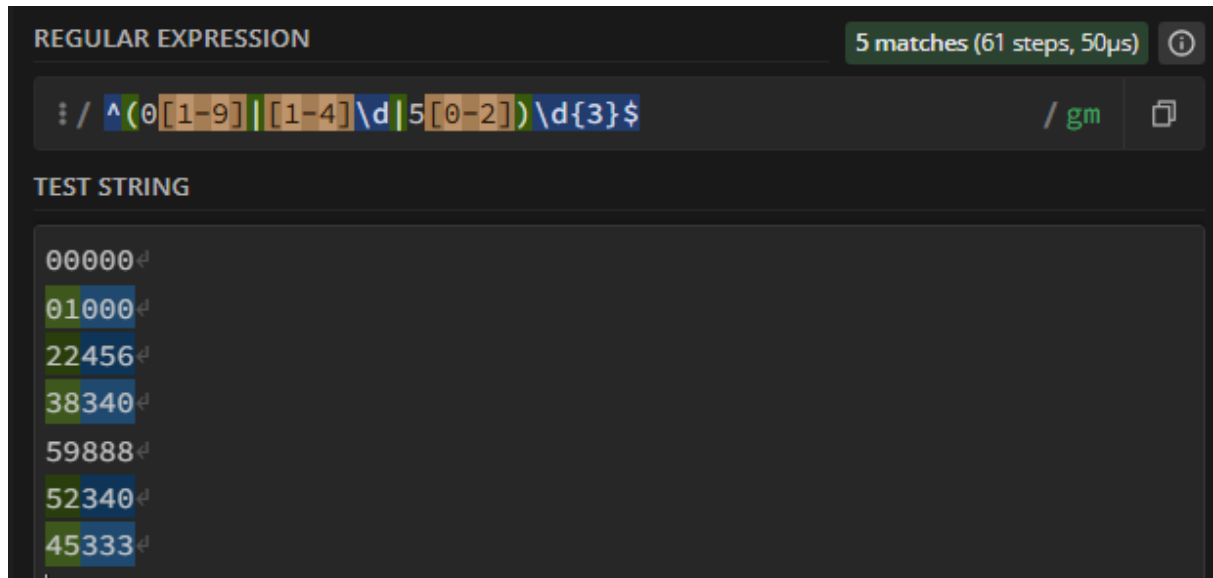


The screenshot shows a regular expression testing interface. At the top, the title is "REGULAR EXPRESSION". On the right, it says "5 matches (20 steps, 40µs)". Below the title, the regular expression pattern is entered as `\([^\)]*\)`. To the right of the pattern, there are flags `/gm` and a copy icon. Below the pattern input, the section is titled "TEST STRING". The test string is a multi-line text: `(hola)  
hola*que*tal*(me*llamo*alejandro)  
(123)  
(esto*es*una  
prueba)  
(####@)  
soto  
musica  
canciones)  
(probando`. The matches found by the regular expression are highlighted in blue: `(hola)`, `(me*llamo*alejandro)`, `(123)`, `(esto*es*una  
prueba)`, and `(####@)`. Each match has a small cursor icon to its right.



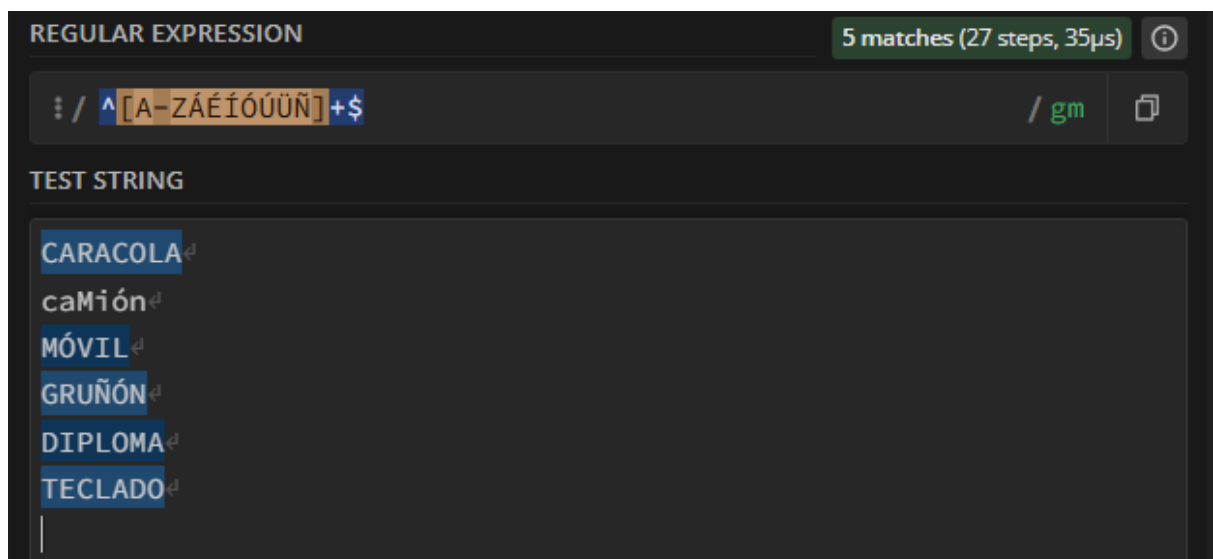
## 2.5. Código postal en España.

- Expresión regular: `^(0[1-9]|[1-4]\d|5[0-2])\d{3}$`



## 2.6. Palabras que contienen sólo letras mayúsculas.

- Expresión regular: `^[A-ZÁÉÍÓÚÛÑ]+$`



2.7. Números de teléfono en formato prefijo  
XXX-XXX-XXX, donde el prefijo del país puede indicarse  
empezando por 00 o bien con un símbolo +; por  
ejemplo, 0034 o +34 para España.

- Expresión regular: `^(00|\+)\d{1,3}\s\d{3}-\d{3}-\d{3}$`

REGULAR EXPRESSION 8 matches (119 steps, 75µs)

`^(00|\\+)\d{1,3}\\s\d{3}-\d{3}-\d{3}$` / gm

TEST STRING

+34 • 600-123-456<sup>d</sup>  
0034 • 987-654-321<sup>d</sup>  
+1 • 123-456-789<sup>d</sup>  
001 • 555-123-456<sup>d</sup>  
+44 • 020-123-456<sup>d</sup>  
0049 • 030-123-456<sup>d</sup>  
+81 • 123-456-789<sup>d</sup>  
00358 • 123-456-789

2.8. Fecha en formato DD/MM/AAAA

- Expresión regular: `^(0|[1-9])|([12][0-9]|3[01])\\/(0|[1-9])|1[0-2])\\/[0-9]{4}$`

REGULAR EXPRESSION 6 matches (97 steps, 60µs)

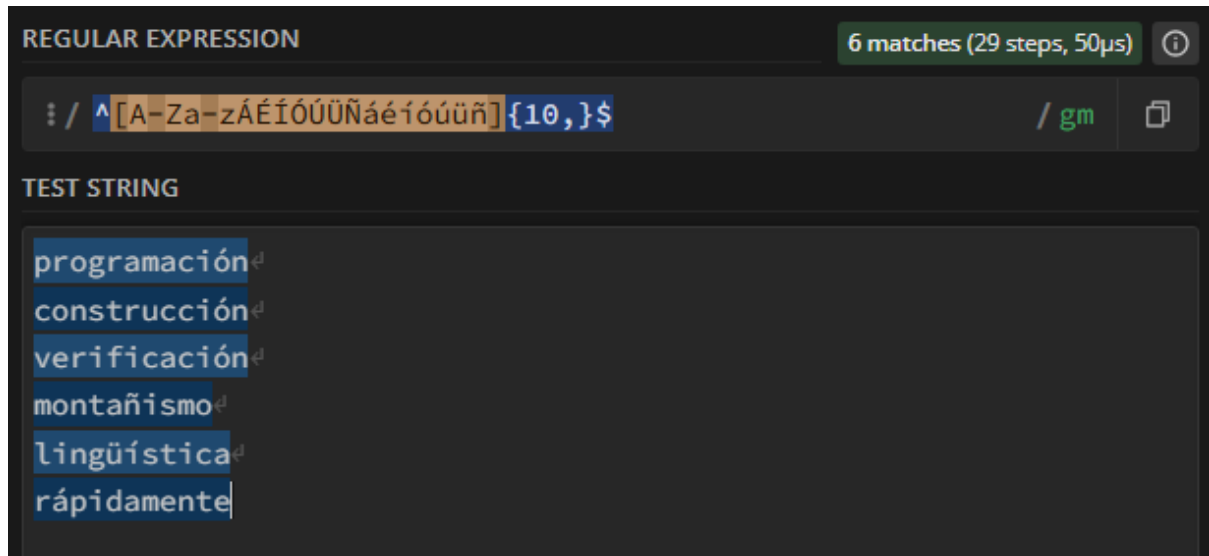
`^(0|[1-9])|([12][0-9]|3[01])\\/(0|[1-9])|1[0-2])\\/[0-9]{4}$` / gm

TEST STRING

01/01/0000<sup>d</sup>  
09/09/1999<sup>d</sup>  
10/10/2020<sup>d</sup>  
31/12/2025<sup>d</sup>  
30/04/2024<sup>d</sup>  
28/02/2023

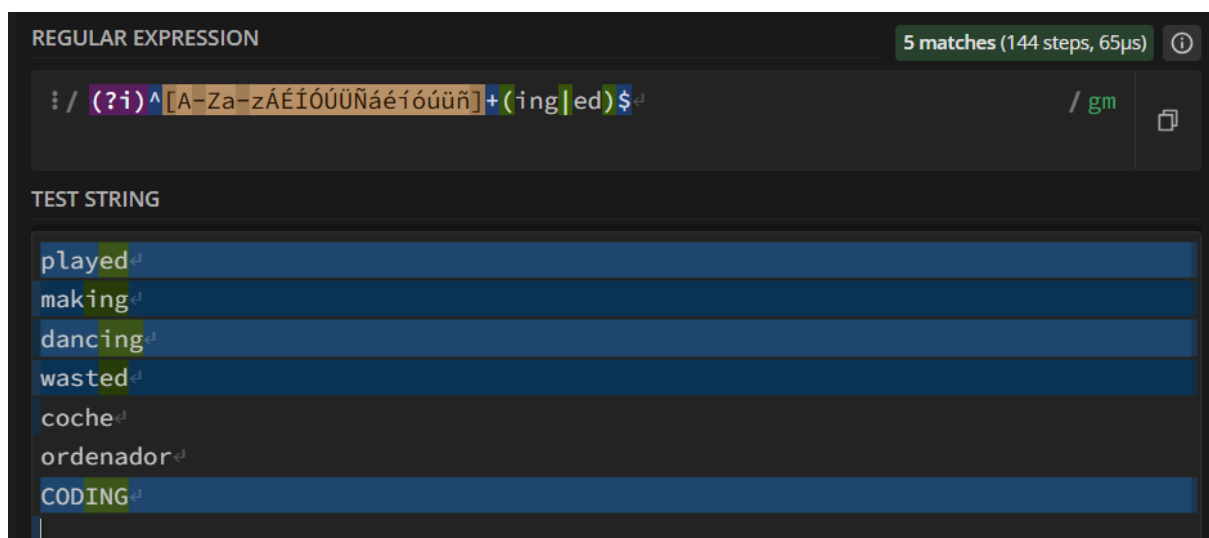
## 2.9. Palabras de al menos 10 letras de longitud.

- Expresión regular: `^[A-Za-zÁÉÍÓÚÛÑáéíóúüñ]{10,}$`



## 2.10. Palabras que terminen con “ing” o “ed”.

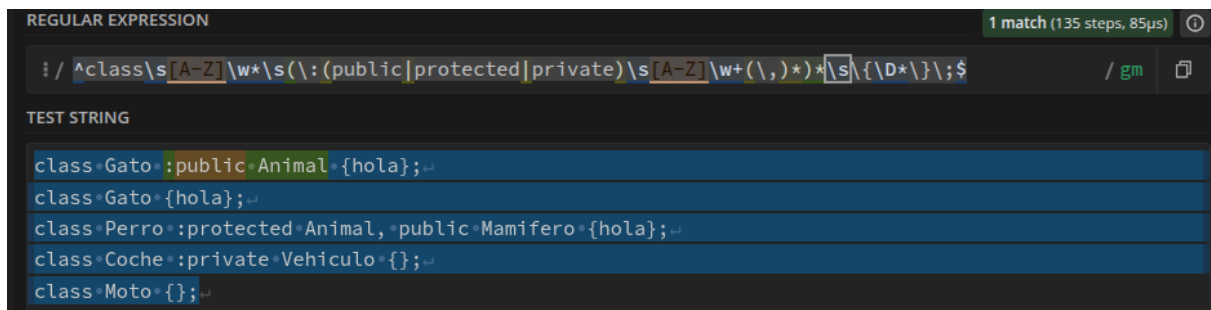
- Expresión regular: `(?i)^[A-Za-zÁÉÍÓÚÛÑáéíóúüñ]+(ing|ed)$`



## 3. Modificación

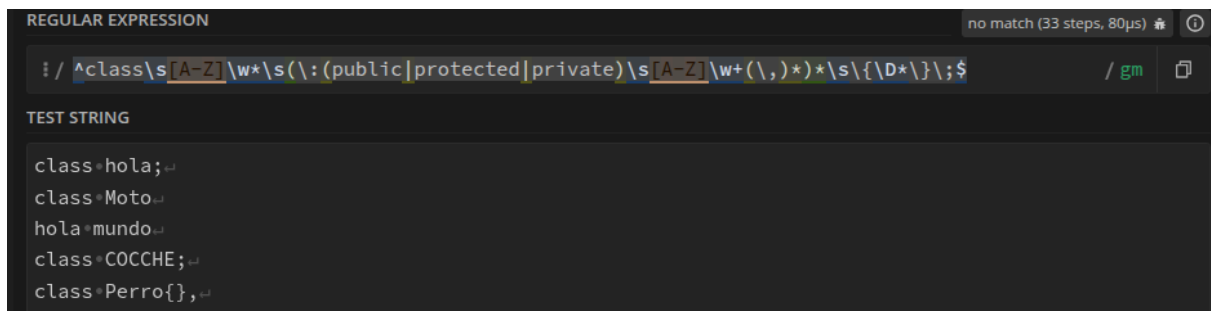
### 3.1. Sentencias class en C++

- Expresión regular:  
`^class\s[A-Z]\w*\s(\:(public|protected|private)\s[A-Z]\w+(\,)*\s{\D*}\);$`
- Cadenas que si acepta:



The screenshot shows a regular expression tester interface. The top bar indicates "1 match (135 steps, 85µs)". The regular expression entered is `^class\s[A-Z]\w*\s(\:(public|protected|private)\s[A-Z]\w+(\,)*\s{\D*}\);$`. The test string contains five lines of C++ code: `class Gato: public Animal {hola};`, `class Gato {hola};`, `class Perro: protected Animal, public Mamifero {hola};`, `class Coche: private Vehiculo {};`, and `class Moto {};`. The first line is highlighted in blue, indicating a successful match.

- Cadenas que no acepta:

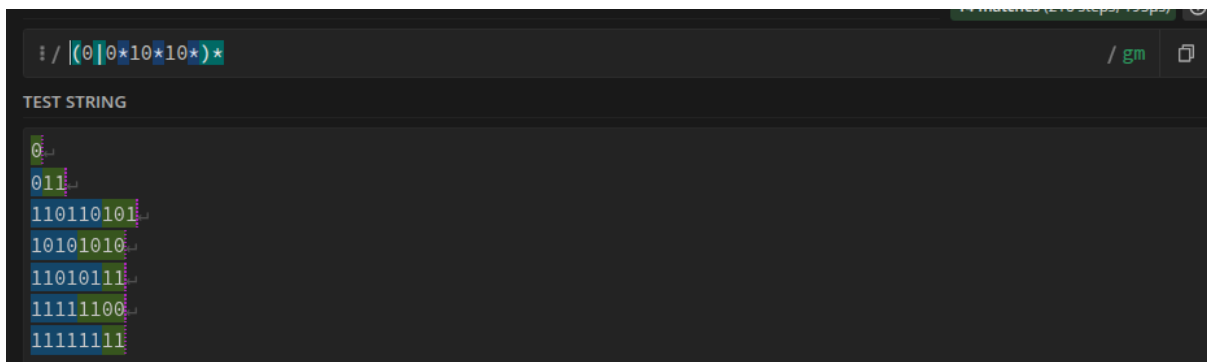


The screenshot shows the same regular expression tester interface, but the test string contains five lines of invalid C++ code: `class hola;`, `class Moto`, `hola mundo`, `class COCCHE;`, and `class Perro{}`. The top bar indicates "no match (33 steps, 80µs)".

### 3.2. Lenguaje binario con número par de 1's

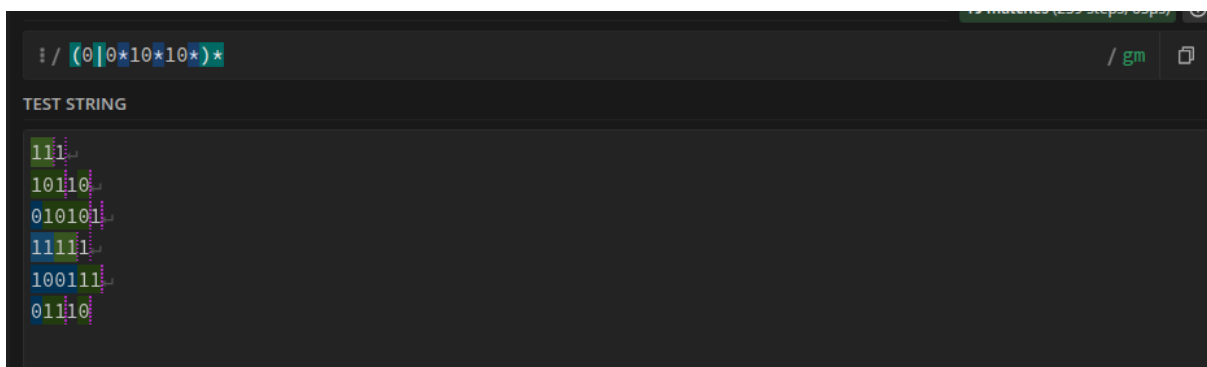
- Expresión regular: `(0|0*10*10*)*`

- Cadenas que si acepta:



A screenshot of a regex testing interface. The top bar shows the regex pattern `/(0|0*10*10*)*` in a teal font. Below the pattern bar, the text "TEST STRING" is displayed. A list of binary strings is shown, each with a green highlight on the left and a red dashed line on the right, indicating they are accepted by the regex. The strings are: `0`, `011`, `110110101`, `10101010`, `11010111`, `11111100`, and `11111111`.

- Cadenas que no acepta:



A screenshot of a regex testing interface, similar to the one above. The top bar shows the same regex pattern `/(0|0*10*10*)*`. Below the pattern bar, the text "TEST STRING" is displayed. A list of binary strings is shown, each with a green highlight on the left and a red dashed line on the right, indicating they are not accepted by the regex. The strings are: `111`, `10110`, `01010`, `1111`, `100111`, and `01110`.