

Práctica 3 - Fundamentos de LP

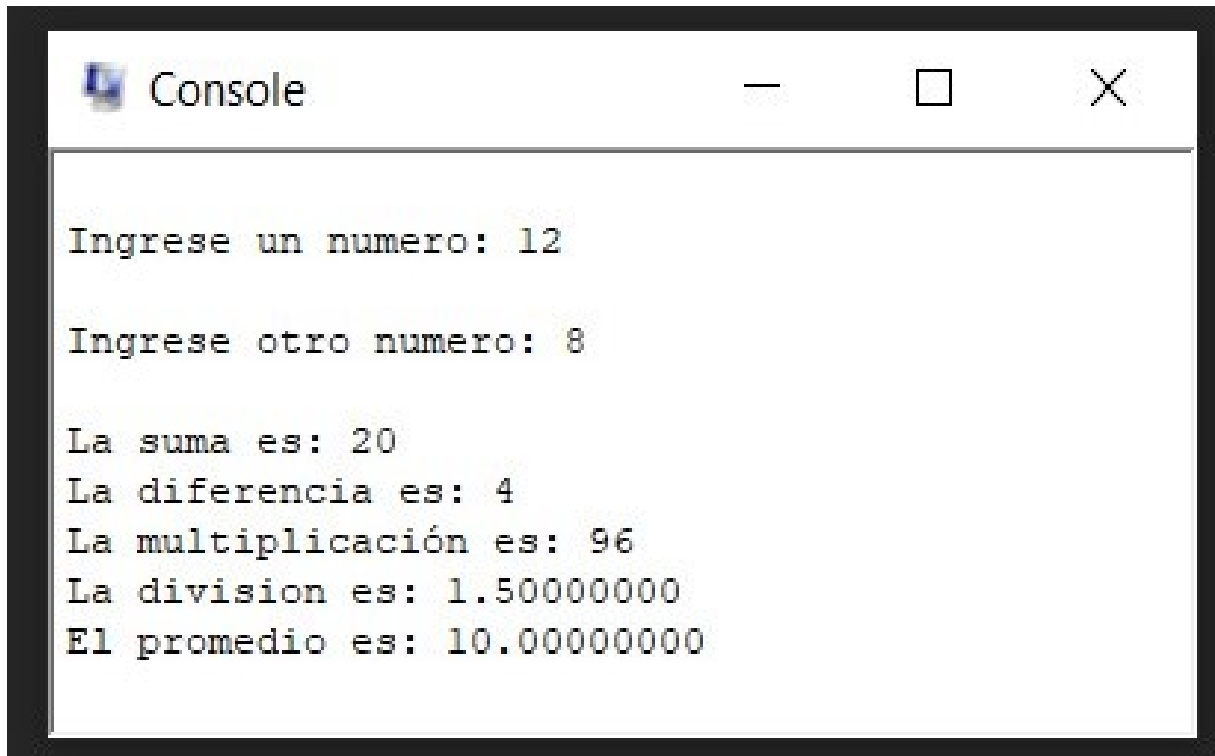
Hecho por: *Elvis André Cruces Gómez*

Ejercicio 1

1. Implementar un programa que muestre la suma, la diferencia, la multiplicación, la división y el promedio de dos números ingresados por teclado.

```
>> Ingrese un numero: 3
>> Ingrese otro numero: 2
>> La suma es: 5
>> La diferencia es: 1
>> La multiplicacion es: 6
>> La division es: 1.5
>> El promedio es: 2.5
```

Código



```
Console

Ingrese un numero: 12

Ingrese otro numero: 8

La suma es: 20
La diferencia es: 4
La multiplicación es: 96
La division es: 1.50000000
El promedio es: 10.00000000
```

```

1  .data
2      num_1: .asciiz "\nIngrese un numero: "
3      num_2: .asciiz "\nIngrese otro numero: "
4      suma: .asciiz "\nLa suma es: "
5      resta: .asciiz "\nLa diferencia es: "
6      multi: .asciiz "\nLa multiplicación es: "
7      divi: .asciiz "\nLa division es: "
8      promedio: .asciiz "\nEl promedio es: "
9  .text
10 main:
11     li $v0, 4
12     la $a0, num_1
13     syscall
14
15     li $v0, 5
16     syscall
17     move $t1, $v0
18
19     li $v0, 4
20     la $a0, num_2
21     syscall
22
23     li $v0, 5
24     syscall
25     move $t2, $v0
26
27 #OPERACIONES
28 #SUMA
29     la $a0, suma
30     li $v0, 4
31     syscall
32
33     add $t0, $t1, $t2
34     li $v0, 1
35     syscall
36     move $a0, $t0
37
38 #RESTA
39     la $a0, resta
40     li $v0, 4
41     syscall
42
43     sub $a0, $t1, $t2
44     li $v0, 1
45     syscall
46
47 #MULTIPLICACIÓN
48     la $a0, multi
49     li $v0, 4
50     syscall
51
52     mul $a0, $t1, $t2
53     li $v0, 1
54     syscall
55
56 #DIVISIÓN
57     la $a0, divi
58     li $v0, 4
59     syscall
60
61     mtc1 $t1, $f1
62     cvt.s.w $f1, $f1
63     mtc1 $t2, $f2
64     cvt.s.w $f2, $f2
65
66     div.s $f12, $f1, $f2
67     li $v0, 2
68     syscall
69
70 #PROMEDIO
71     la $a0, promedio
72     li $v0, 4
73     syscall
74
75     mtc1 $t0, $f3
76     cvt.s.w $f3, $f3
77
78     li.s $f4, 2.0
79
80     div.s $f12, $f3, $f4
81     li $v0, 2
82     syscall
83
84     jr $ra

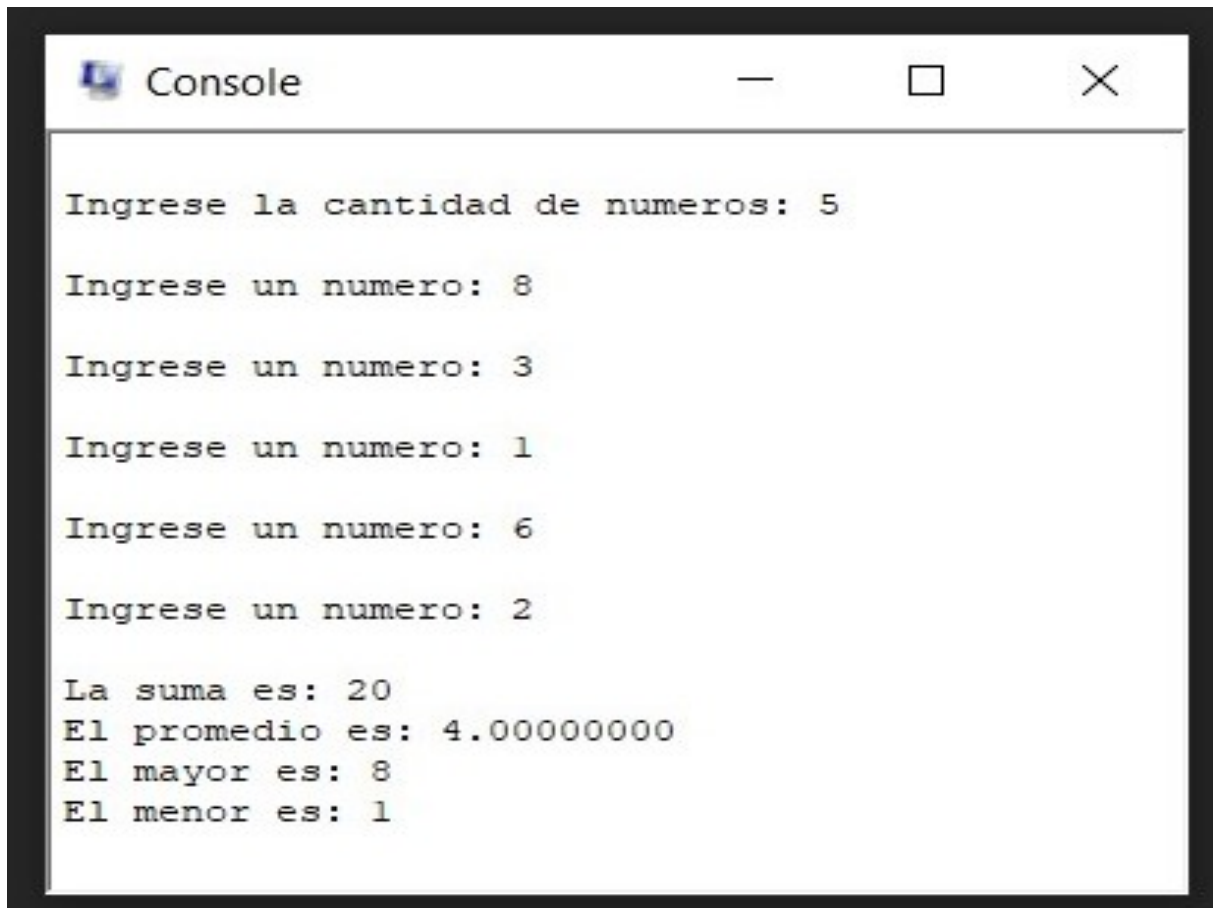
```

Ejercicio 2

2. Implementar un programa que solite una cantidad n de números y luego retorne: la suma de estos, el promedio, el mayor y el menor.

```
>> Ingrese la cantidad de numeros: 4
>> Ingrese un numero: 3
>> Ingrese un numero: 2
>> Ingrese un numero: 2
>> Ingrese un numero: 3
>> La suma es: 10
>> El promedio es: 2.5
>> El mayor es: 3
>> El menor es: 2
```

Código



```
Console

Ingrese la cantidad de numeros: 5

Ingrese un numero: 8

Ingrese un numero: 3

Ingrese un numero: 1

Ingrese un numero: 6

Ingrese un numero: 2

La suma es: 20
El promedio es: 4.00000000
El mayor es: 8
El menor es: 1
```

```

1  .data
2      cant: .asciiz "\nIngrese la cantidad de numeros: "
3      numbers: .asciiz "\nIngrese un numero: "
4      suma: .asciiz "\nLa suma es: "
5      promedio: .asciiz "\nEl promedio es: "
6      mayor: .asciiz "\nEl mayor es: "
7      menor: .asciiz "\nEl menor es: "
8  .text
9  main:
10     li $v0, 4
11     la $a0, cant
12     syscall
13
14     li $t1, 0
15
16     li $v0, 5
17     syscall
18     move $t2, $v0
19
20     li $a1, 0
21     li $t3, 0
22     li $t6, 1000000000000000
23
24     Loop:
25         beq $t2, $t1, Exit
26
27         li $v0, 4
28         la $a0, numbers
29         syscall
30
31         li $v0, 5
32         syscall
33         move $t5, $v0
34
35         beqz $a1, LABEL_IF
36         LABEL_IF_ELSE:
37             add $t1, $t1, 0
38             b LABEL_IF
39         LABEL_IF:
40             move $t4, $v0
41         end_LABEL_IF:
42             add $t1, $t1, 0
43
44         bge $t5, $t3, MAYOR_IF
45         MAYOR_IF_ELSE:
46             add $t1, $t1, 0
47             b END_MAYOR_IF
48         MAYOR_IF:
49             move $t3, $t5
50         END_MAYOR_IF:
51             add $t1, $t1, 0
52
53         ble $t5, $t6, MENOR_IF
54         MENOR_IF_ELSE:
55             add $t1, $t1, 0
56             b END_MENOR_IF
57         MENOR_IF:
58             move $t6, $t5
59         END_MENOR_IF:
60             add $t1, $t1, 0
61
62         move $a2, $v0
63         add $a1, $a1, $a2
64         add $t1, $t1, 1
65
66     j Loop
67     Exit:
68
69     #OPERACIONES
70     #SUMA
71     la $a0, suma
72     li $v0, 4
73     syscall
74
75     move $a0, $a1
76     li $v0, 1
77     syscall
78
79     #PROMEDIO
80     li $v0, 4
81     la $a0, promedio
82     syscall
83
84     mtcl $a1, $f1
85     cvt.s.w $f1, $f1
86     mtcl $t2, $f2
87     cvt.s.w $f2, $f2
88     div.s $f12, $f1, $f2
89     li $v0, 2
90     syscall
91
92     #MAYOR
93     la $a0, mayor
94     li $v0, 4
95     syscall
96
97     move $a0, $t3
98     li $v0, 1
99     syscall
100
101     #MENOR
102     la $a0, menor
103     li $v0, 4
104     syscall
105
106     move $a0, $t6
107     li $v0, 1
108     syscall
109
110     jr $ra

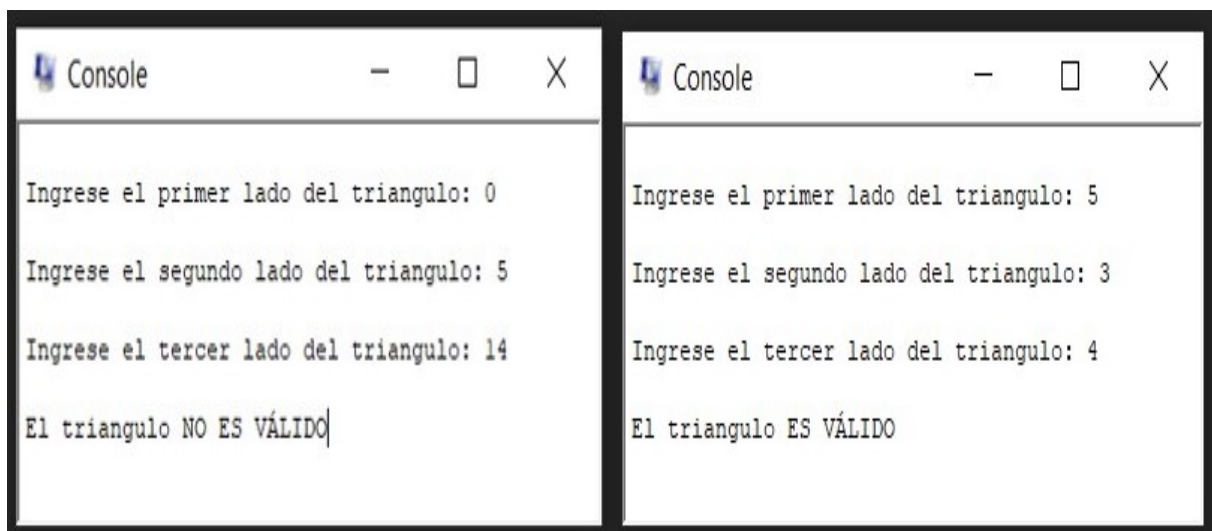
```

Ejercicio 3

3. Implemente un programa que solicite por teclado: la longitud de los tres lados de un triángulo. Luego el programa debe indicar si es un triángulo válido.

```
>> Ingrese el primer lado del triángulo: 2
>> Ingrese el segundo lado del triángulo: 3
>> Ingrese el tercer lado del triángulo: 2
>> El triángulo es válido
```

Solución



```

1      .data
2      lado1: .asciiz "\nIngrese el primer lado del triangulo: "
3      lado2: .asciiz "\nIngrese el segundo lado del triangulo: "
4      lado3: .asciiz "\nIngrese el tercer lado del triangulo: "
5      valido: .asciiz "\nEl triangulo ES VÁLIDO"
6      invalido: .asciiz "\nEl triangulo NO ES VÁLIDO"
7
8      .text
9      main:
10         li $v0, 4
11         la $a0, lado1
12         syscall
13
14         li $v0, 5
15         syscall
16         move $t1, $v0
17
18         li $v0, 4
19         la $a0, lado2
20         syscall
21
22         li $v0, 5
23         syscall
24         move $t2, $v0
25
26         li $v0, 4
27         la $a0, lado3
28         syscall
29
30         li $v0, 5
31         syscall
32         move $t3, $v0
33
34         add $t4, $t1, $t2
35         add $t5, $t2, $t3
36         add $t6, $t1, $t3
37
38         bge $t3, $t4, TrianguloINV
39         bge $t2, $t6, TrianguloINV
40         bge $t1, $t5, TrianguloINV
41
42         b TrianguloVal
43         #TRIANGULO VALIDO
44         TrianguloVal:
45             li $v0, 4
46             la $a0, valido
47             syscall
48             b FIN
49         #TRIANGULO INVALIDO
50         TrianguloINV:
51             li $v0, 4
52             la $a0, invalido
53             syscall
54             b FIN
55
56         FIN:
57         jx $ra

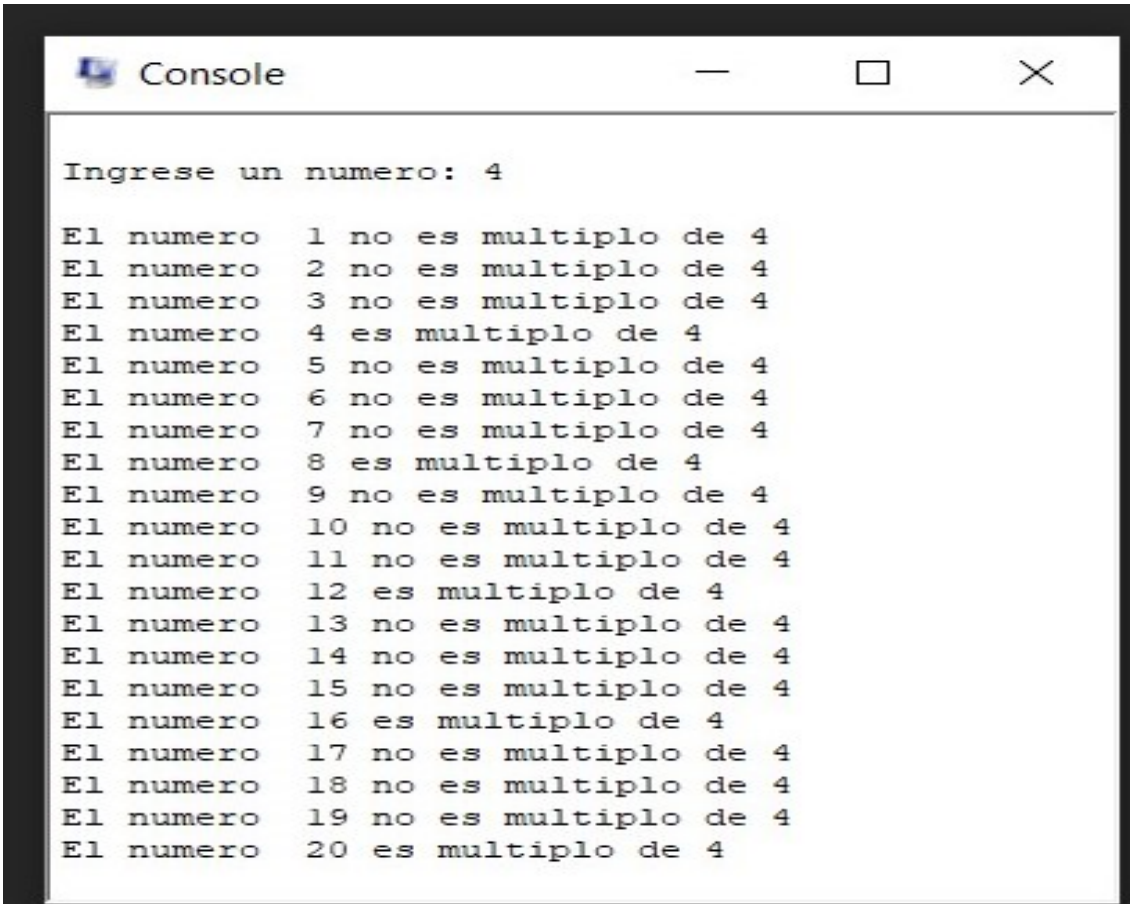
```

Ejercicio 4

4. Implemente un programa que solicite un número n , luego este debe mostrar que números desde 1 a 20, son multiplos de n .

```
>> Ingrese un numero: 3
>> El numero 1 no es multiplo de 3
>> El numero 2 no es multiplo de 3
>> El numero 3 si es multiplo de 3
>> El numero 4 no es multiplo de 3
>> El numero 5 no es multiplo de 3
>> El numero 6 si es multiplo de 3
...
```

Solución



The screenshot shows a console window titled "Console" with standard window controls (minimize, maximize, close). The text inside the console displays the program's execution for the input number 4. It prompts the user to enter a number, then iterates through numbers 1 to 20, checking if each is a multiple of 4. The output shows that numbers 1, 2, 3, 5, 6, 7, 9, 10, 11, 13, 14, 15, 17, 18, 19 are not multiples, while 4, 8, 12, 16, and 20 are.

```
Ingresa un numero: 4

El numero 1 no es multiplo de 4
El numero 2 no es multiplo de 4
El numero 3 no es multiplo de 4
El numero 4 es multiplo de 4
El numero 5 no es multiplo de 4
El numero 6 no es multiplo de 4
El numero 7 no es multiplo de 4
El numero 8 es multiplo de 4
El numero 9 no es multiplo de 4
El numero 10 no es multiplo de 4
El numero 11 no es multiplo de 4
El numero 12 es multiplo de 4
El numero 13 no es multiplo de 4
El numero 14 no es multiplo de 4
El numero 15 no es multiplo de 4
El numero 16 es multiplo de 4
El numero 17 no es multiplo de 4
El numero 18 no es multiplo de 4
El numero 19 no es multiplo de 4
El numero 20 es multiplo de 4
```



```

1  .data
2  number: .asciiz "\nIngrese un numero: "
3  numero: .asciiz "\nEl numero  "
4  esMultiplo: .asciiz " es multiplo de "
5  noMultiplo: .asciiz " no es multiplo de "
6
7  .text
8  main:
9      li $t1, 1
10     li $t2, 21
11
12     li $v0, 4
13     la $a0, number
14     syscall
15
16     li $v0, 5
17     syscall
18     move $a1, $v0
19
20     LoopMultiplo:
21         beq $t2, $t1, Exit
22
23         div $t1, $a1
24         mfhi $t3
25
26         beq $t3, 0, Multiplo
27
28         NoMultiplo:
29             li $v0, 4
30             la $a0, numero
31             syscall
32             move $a0 $t1
33
34             li $v0, 1
35             syscall
36             move $t1 $a0
37
38             li $v0, 4
39             la $a0, noMultiplo
40             syscall
41             move $a0 $a1
42
43             li $v0, 1
44             syscall
45             move $a1 $a0
46
47             add $t1, $t1, 1
48             j LoopMultiplo
49
50     Multiplo:
51         li $v0, 4
52         la $a0, numero
53         syscall
54         move $a0 $t1
55
56         li $v0, 1
57         syscall
58         move $t1 $a0
59
60         li $v0, 4
61         la $a0, esMultiplo
62         syscall
63         move $a0 $a1
64
65         li $v0, 1
66         syscall
67         move $a1 $a0
68
69         add $t1, $t1, 1
70         j LoopMultiplo
71     Exit:
72
73     jr $ra

```

Repositorio: