

Programación II

Práctica 00: Acumuladores booleanos

Como vimos en clase un **acumulador** booleano toma en realidad dos valores.

Se denomina acumulador, porque suma un resultado parcial al resultado final de la función.

La variable “ret” es el nombre abreviado de “valor de retorno”.

Cuantificadores

Cuando queremos probar una propiedad P para todo un conjunto de datos:

$$\{\forall x \in lista / P(x)\} \equiv true$$

Utilizaremos la hipótesis de ret = **true** y la acumulación será de la forma:

$$ret = ret \text{ and } P(x)$$

Cuando queremos probar una propiedad P para un solo elemento:

$$\{\exists x \in lista / P(x)\} \equiv true$$

Utilizaremos la hipótesis de ret = **false** y la acumulación será de la forma:

$$ret = ret \text{ or } P(x)$$

Ejercicios obligatorios:

- 1) Realizar una función que dada una matriz
Si tiene más filas que columnas, sume todos los 5
Si tiene igual o más columnas que filas, que sume todos los 6

```
public static int sumaFilasColu(int[][] mtx) ...
```

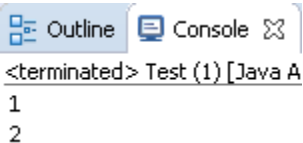
Ayuda:

La cantidad de filas de mtx se consulta con `mtx.length()`

La cantidad de columnas de mtx se consulta con `mtx[0].length()`

```
int[][] mtx = new int[1][2];

System.out.println(mtx.length);
System.out.println(mtx[0].length);
```



- 2) Realizar una función que imprima los primeros 30 valores de 2^n y de $\log_2(m)$ donde $m = 2^n$
- a)

n	2^n	$\log_2(m)$
0	1	0
1	2	1
2	4	2
3
...
29

Ayuda: No hace falta dibujar la “tabla”.

- b) Interpretar cual es la relación entre 2^n y $\log_2(n)$

Además:

<https://prog2-ungs.github.io/practicas/prac0/>