8. Calculá el orden del número de asignaciones a la variable t de los siguientes algoritmos.

(a)
```
t := 1
do t < n
    t := t * 2
od
```

(b)
```
t := n
do t > 0
    t := t div 2
od
```

(c)
```
for i := 1 to n do
    t := i
    do t > 0
        t := t div 2
    od
od
```

(d)
```
for i := 1 to n do
    t := i
    do t > 0
        t := t − 2
    od
od
```

a) Pruebo valores de *n* para ver el comportamiento del ciclo:

n	**1**	**2**	**3**	**4**	**5**	**6**	· · ·	**9**	· · ·	**17**	· · ·
t	1	2	4	4	8	8	· · ·	16	· · ·	32	· · ·
ops	1	2	3	3	4	4	· · ·	5	· · ·	6	· · ·

Se puede ver que *t* toma valores de las potencias de 2 (1, 2, 4, 8, 16, 32, 64, 128, ...), donde *n* y *t* crecen de manera exponencial.
La relación entre *t* y *ops* es $t = 2^{ops-1}$, donde al despejar ops

$$2^{ops-1} = t \Rightarrow \log_2(t) = ops - 1 \Rightarrow \mathbf{ops = \log_2(t) + 1}$$

∴ el orden de complejidad del algoritmo es ***O(log(n))*** ó ***logarítmica***.

---

b) Veo el comportamiento del ciclo, dando valores a *n*:

n	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	· · ·	**16**
t	0	0	0	0	0	0	0	0	0	· · ·	0
ops	1	2	2	3	3	3	3	4	4	· · ·	5

Donde las iteraciones del ciclo aumentan en 1 a medida que *n* es una potencia de 2 (1, 2, 4, 8, 16, 32, 64, ...).
Por tanto, puede definirse como $ops = log_2(n) + 1$, donde *ops* es el resultado ENTERO de dividir *n* entre dos.
Sea *C* el algoritmo del enunciado, entonces:

ops(C) = ops(t := n) + ops(while t > 0 do t := t div 2 od) = **1 + $\log_2$(n) + 1**

∴ el orden de complejidad del algoritmo es ***O(log(n))*** ó ***logarítmica***.

---

c)

$$\begin{aligned}
\text{ops(C)} &= \text{ops(for i := 1 to } n \text{ do ... od)} \\
&= \sum_{k=1}^{n} \text{ops(t := 1; while } t > 0 \text{ do t := t div 2 od)} \\
&= \sum_{k=1}^{n} (\text{ops(t := 1)} + \text{ops(while } t > 0 \text{ do t := t div 2 od)}) \\
&= \sum_{k=1}^{n} (log_2(n) + 2) \\
&= \sum_{k=1}^{n} log_2(n) + \sum_{k=1}^{n} 2 \\
&= n * log_2(2) + n * 2 = \mathbf{n(\log_2(n) + 2)}
\end{aligned}$$

∴ el orden de complejidad del algoritmo es ***O(log(n))*** o ***logarítmica***.

d) Veo el comportamiento del ciclo, dando valores a *i*:

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	· · ·	**9**	· · ·	**11**
t	-1	0	-1	0	-1	0	-1	· · ·	-1	· · ·	-1
ops	1	1	2	2	3	3	4	· · ·	5	· · ·	6

Por tanto, puede definirse al ciclo como **ops =** $\frac{1}{2}$ ***i***, donde *ops* es el resultado ENTERO pero REDONDEADO hacia arriba.
Sea *K* el algoritmo del enunciado, entonces:

$$\begin{aligned}
\text{ops(K)} &= \text{ops(for i := 1 to } n \text{ do ... od)} \\
&= \sum_{i=1}^{n} \text{ops(t := i; while } t > 0 \text{ do t := t - 2 od)} \\
&= \sum_{k=1}^{n} (\text{ops(t := 1)} + \text{ops(while } t > 0 \text{ do t := t - 2 od))} \\
&= \sum_{k=1}^{n} (1 + \frac{1}{2} i) \\
&= \sum_{k=1}^{n} 1 + \frac{1}{2} \sum_{k=1}^{n} i \\
&= n + \frac{1}{2} * \frac{n * (n+1)}{2} = \mathbf{\frac{1}{4} n^2 + \frac{5}{4} n}
\end{aligned}$$

∴ el orden de complejidad del algoritmo es ***O(n²)*** o ***cuadrática***.