Algoritmos y Estructuras de Datos II

Práctico 1 - Parte 1

1. Escribí algoritmos para resolver cada uno de los siguientes problemas sobre un arreglo a de posiciones 1 a n, utilizando **do**. Elegí en cada caso entre estos dos encabezados el que sea más adecuado:

```
proc nombre (in/out a:array[1..n] of nat)        proc nombre (out a:array[1..n] of nat)
    ...                                               ...
end proc                                         end proc
```

   (a) Inicializar cada componente del arreglo con el valor 0.
   (b) Inicializar el arreglo con los primeros n números naturales positivos.
   (c) Inicializar el arreglo con los primeros n números naturales impares.
   (d) Incrementar las posiciones impares del arreglo y dejar intactas las posiciones pares.

2. Transformá cada uno de los algoritmos anteriores en uno equivalente que utilice **for** ... **to** .

3. Escribí un algoritmo que reciba un arreglo a de posiciones 1 a n y determine si el arreglo recibido está ordenado o no. Explicá en palabras **qué** hace el algoritmo. Explicá en palabras **cómo** lo hace.

4. Ordená los siguientes arreglos, utilizando el algoritmo de ordenación por selección visto en clase. Mostrá en cada paso de iteración cuál es el elemento seleccionado y cómo queda el arreglo después de cada intercambio.

   (a) $[7, 1, 10, 3, 4, 9, 5]$ (b) $[5, 4, 3, 2, 1]$ (c) $[1, 2, 3, 4, 5]$

5. Calculá de la manera más exacta y simple posible el número de asignaciones a la variable t de los siguientes algoritmos. Las ecuaciones que se encuentran al final del práctico pueden ayudarte.

(a)
```
t := 0
for i := 1 to n do
    for j := 1 to n^2 do
        for k := 1 to n^3 do
            t := t + 1
        od
    od
od
```

(b)
```
t := 0
for i := 1 to n do
    for j := 1 to i do
        for k := j to j + 3 do
            t := t + 1
        od
    od
od
```

6. Descifrá qué hacen los siguientes algoritmos, explicar cómo lo hacen y reescribirlos asignando nombres adecuados a todos los identificadores

```
proc p (in/out a: array[1..n] of T)
    var x: nat
    for i:= n downto 2 do
        x:= f(a,i)
        swap(a,i,x)
    od
end proc
```

```
fun f (a: array[1..n] of T, i: nat) ret x: nat
    x:= 1
    for j:= 2 to i do
        if a[j] > a[x] then x:= j fi
    od
end fun
```

7. Ordená los arreglos del ejercicio 4 utilizando el algoritmo de ordenación por inserción. Mostrá en cada paso de iteración las comparaciones e intercambios realizados hasta ubicar el elemento en su posición.

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
```

9. Calculá el orden del número de comparaciones del algoritmo del ejercicio 3.

10. Descifrá qué hacen los siguientes algoritmos, explicar cómo lo hacen y reescribirlos asignando nombres adecuados a todos los identificadores

```
proc q (in/out a: array[1..n] of T)
    for i:= n-1 downto 1 do
        r(a,i)
    od
end proc
```

```
proc r (in/out a: array[1..n] of T, in i: nat)
    var j: nat
    j:= i
    do j < n ∧ a[j] > a[j+1] → swap(a,j+1,j)
                               j:= j+1
    od
end proc
```

En las ecuaciones que siguen $n, m \in \mathbb{N}$ y k es una constante arbitraria:

$$\sum_{i=1}^{n} 1 = n$$

$$\sum_{i=m}^{n} 1 = n - m + 1 \qquad \text{si } n \geq m - 1$$

$$\sum_{i=1}^{n} i = \frac{n * (n+1)}{2}$$

$$\sum_{i=1}^{n} i^2 = \frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6}$$

$$\sum_{i=1}^{n} i^3 = \frac{n^4}{4} + \frac{n^3}{2} + \frac{n^2}{4}$$

$$\sum_{i=m}^{n} (k * a_i) = k * (\sum_{i=m}^{n} a_i)$$

$$\sum_{i=m}^{n} (a_i + b_i) = (\sum_{i=m}^{n} a_i) + (\sum_{i=m}^{n} b_i)$$

$$\sum_{i=m}^{n} (a_i - b_i) = (\sum_{i=m}^{n} a_i) - (\sum_{i=m}^{n} b_i)$$

$$\sum_{i=0}^{n} a_{n-i} = \sum_{i=0}^{n} a_i$$

La última ecuación de la derecha dice simplemente que:

$$a_n + a_{n-1} + \ldots + a_1 + a_0 = a_0 + a_1 + \ldots + a_{n-1} + a_n$$