


Redictado Taller de Programación 2022

CLASE 2

Ordenación de vectores

A silver laptop sits on a light-colored wooden desk. The laptop screen is open and displays a simple Pascal program. The background is a plain, light-colored wall.

```
Program HolaMundo;  
Begin  
  writeln('Hola mundo');  
end.
```

TEMAS DE LA CLASE

1. Concepto de Ordenación
2. Método de Ordenación por Inserción
 - *Pseudocódigo*
 - *Análisis de casos*
 - *Análisis teórico*
3. Ejercitación

Bibliografía:

Cap. 8. Algoritmos, Datos y Programas. *Armando E. De Giusti*.



1. CONCEPTO DE ORDENACIÓN.

Algoritmo de Ordenación

Proceso por el cual, un grupo de elementos puede ser ordenado.

¿Por qué es importante una operación de ordenación en arreglos?

Analicemos situaciones donde es necesaria ésta operación...

1. CONCEPTO DE ORDENACIÓN.

MÉTODOS DE ORDENACIÓN CLÁSICOS

Selección

Intercambio

Insertión

2. MÉTODO DE ORDENACIÓN POR INSERCIÓN

Se parte de una secuencia de dos ítems y se ordena.

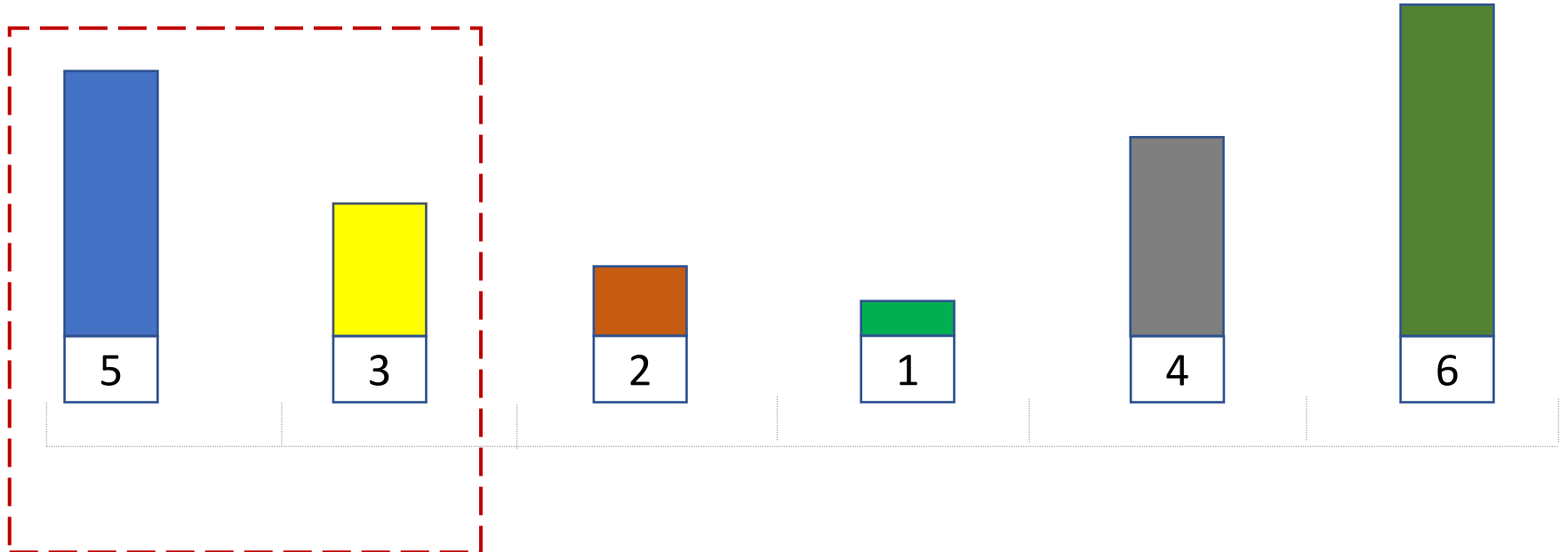
En cada pasada se “*toma*” un nuevo ítem y se inserta en la posición adecuada del arreglo ordenado a su izquierda.

Veamos un ejemplo...

2. MÉTODO DE ORDENACIÓN POR INSERCIÓN

EJEMPLO: Ordenando de menor a mayor...

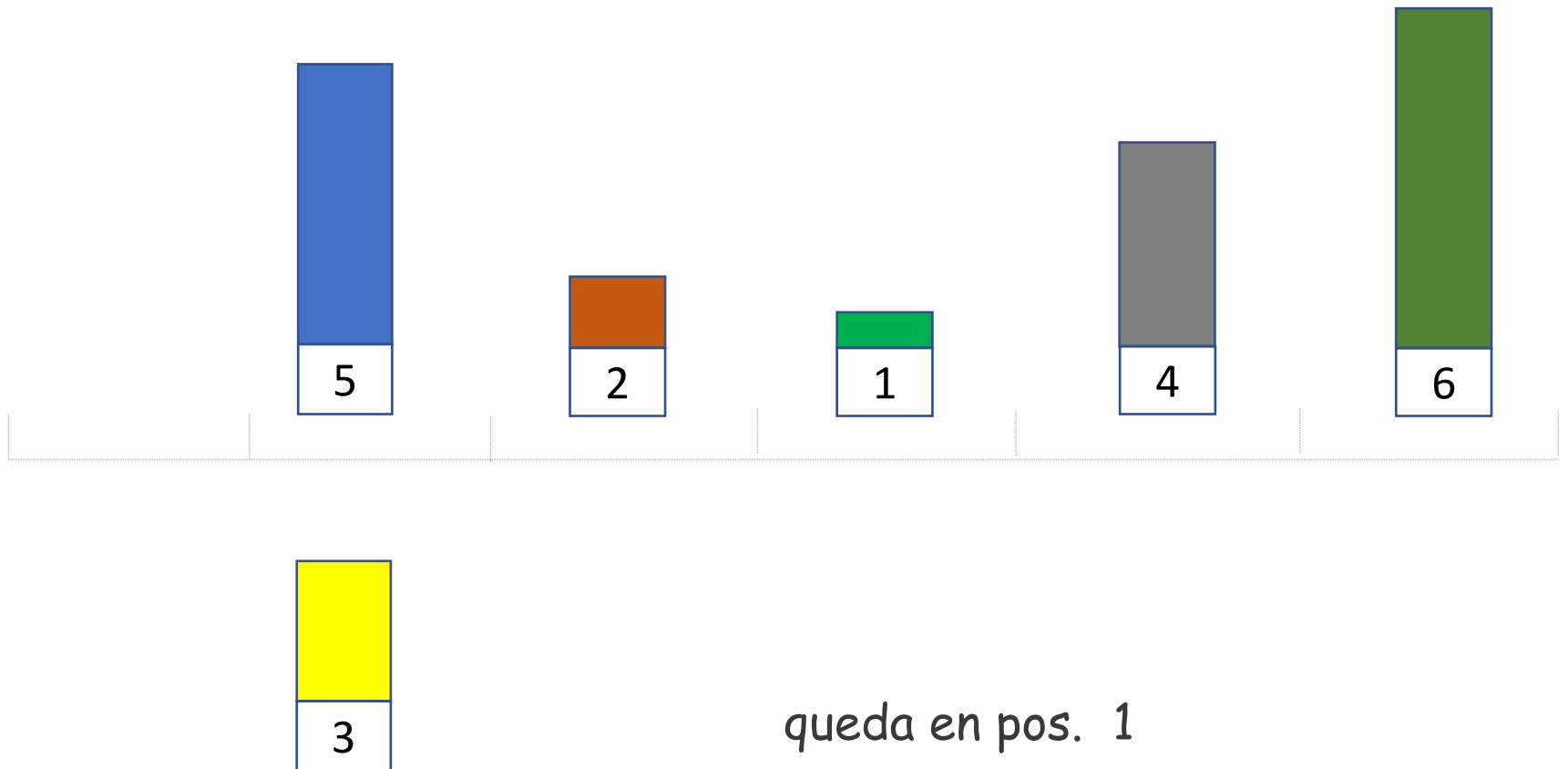
$i=2$ elemento a ordenar = 3



2. MÉTODO DE ORDENACIÓN POR INSERCIÓN

EJEMPLO: Ordenando de menor a mayor...

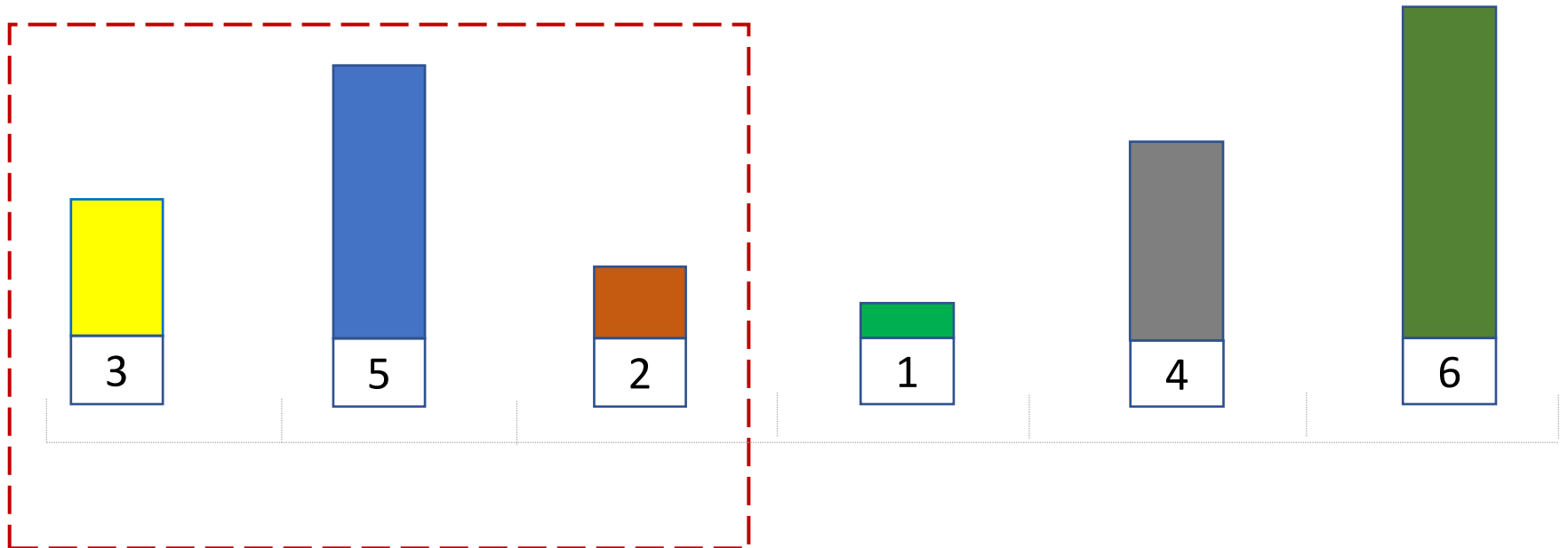
$i=2$ elemento a ordenar = 3



2. MÉTODO DE ORDENACIÓN POR INSERCIÓN

EJEMPLO: Ordenando de menor a mayor...

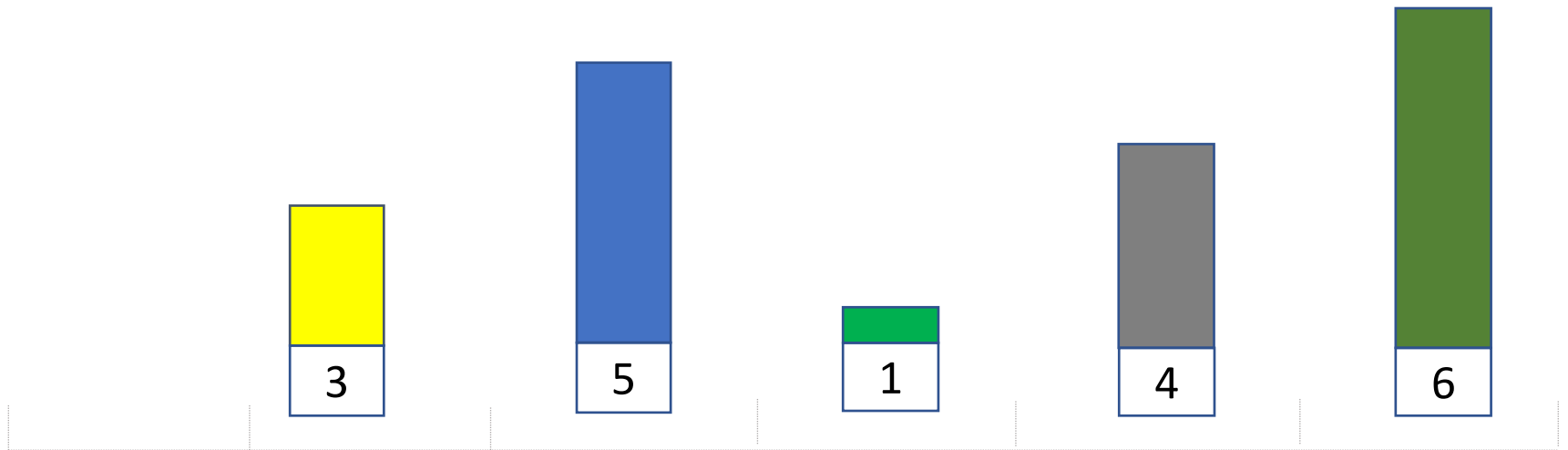
$i=3$ elemento a ordenar = 2



2. MÉTODO DE ORDENACIÓN POR INSERCIÓN

EJEMPLO: Ordenando de menor a mayor...

$i=3$ elemento a ordenar = 2

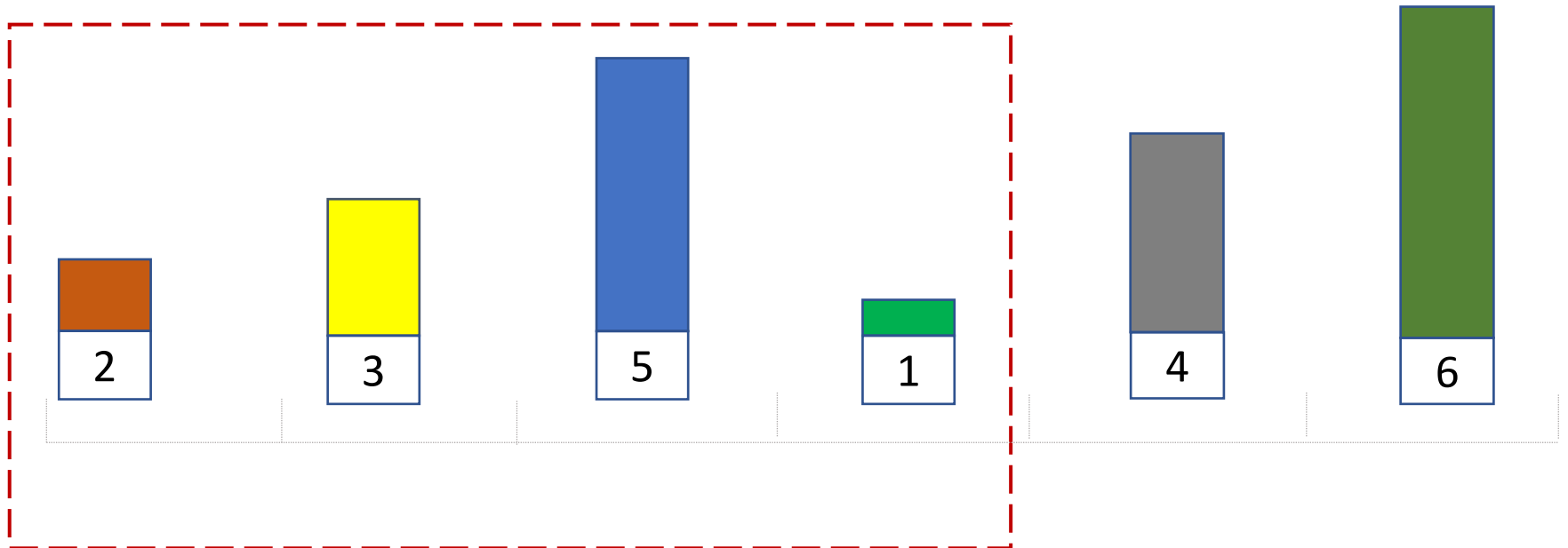


queda en pos. 1

2. MÉTODO DE ORDENACIÓN POR INSERCIÓN

EJEMPLO: Ordenando de menor a mayor...

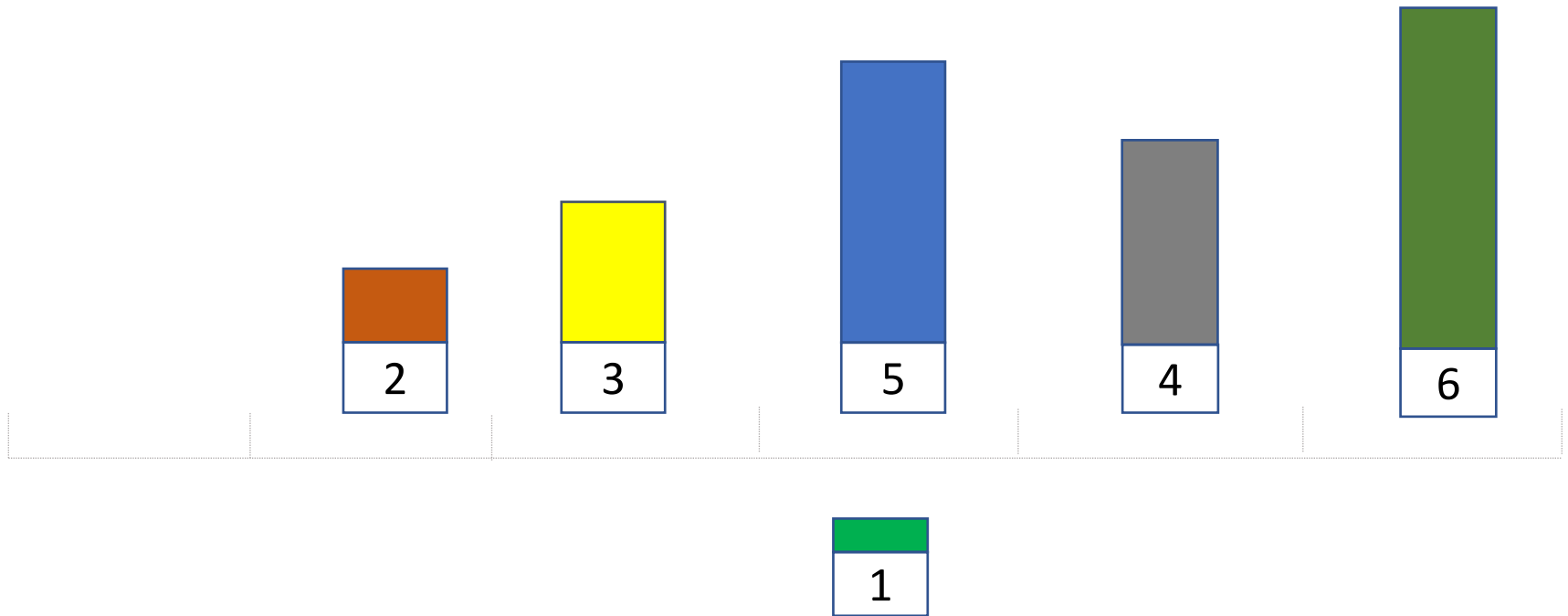
$i=4$ elemento a ordenar = 1



2. MÉTODO DE ORDENACIÓN POR INSERCIÓN

EJEMPLO: Ordenando de menor a mayor...

$i=4$ elemento a ordenar = 1

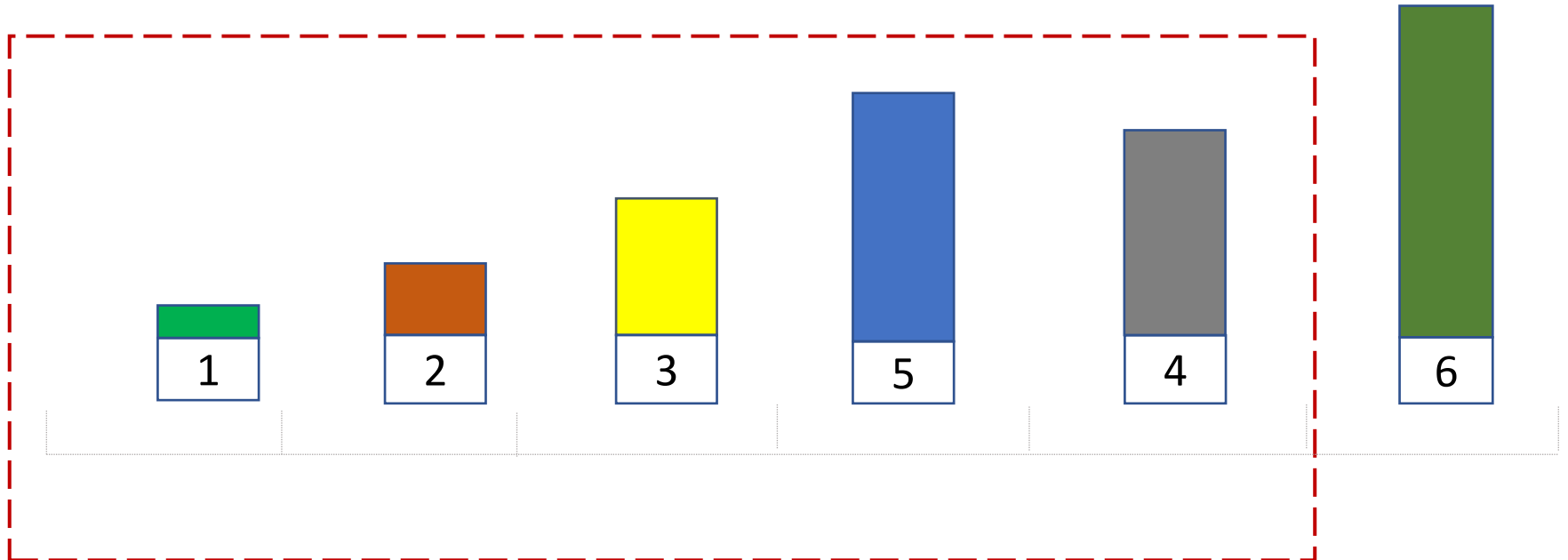


queda en pos. 1

2. MÉTODO DE ORDENACIÓN POR INSERCIÓN

EJEMPLO: Ordenando de menor a mayor...

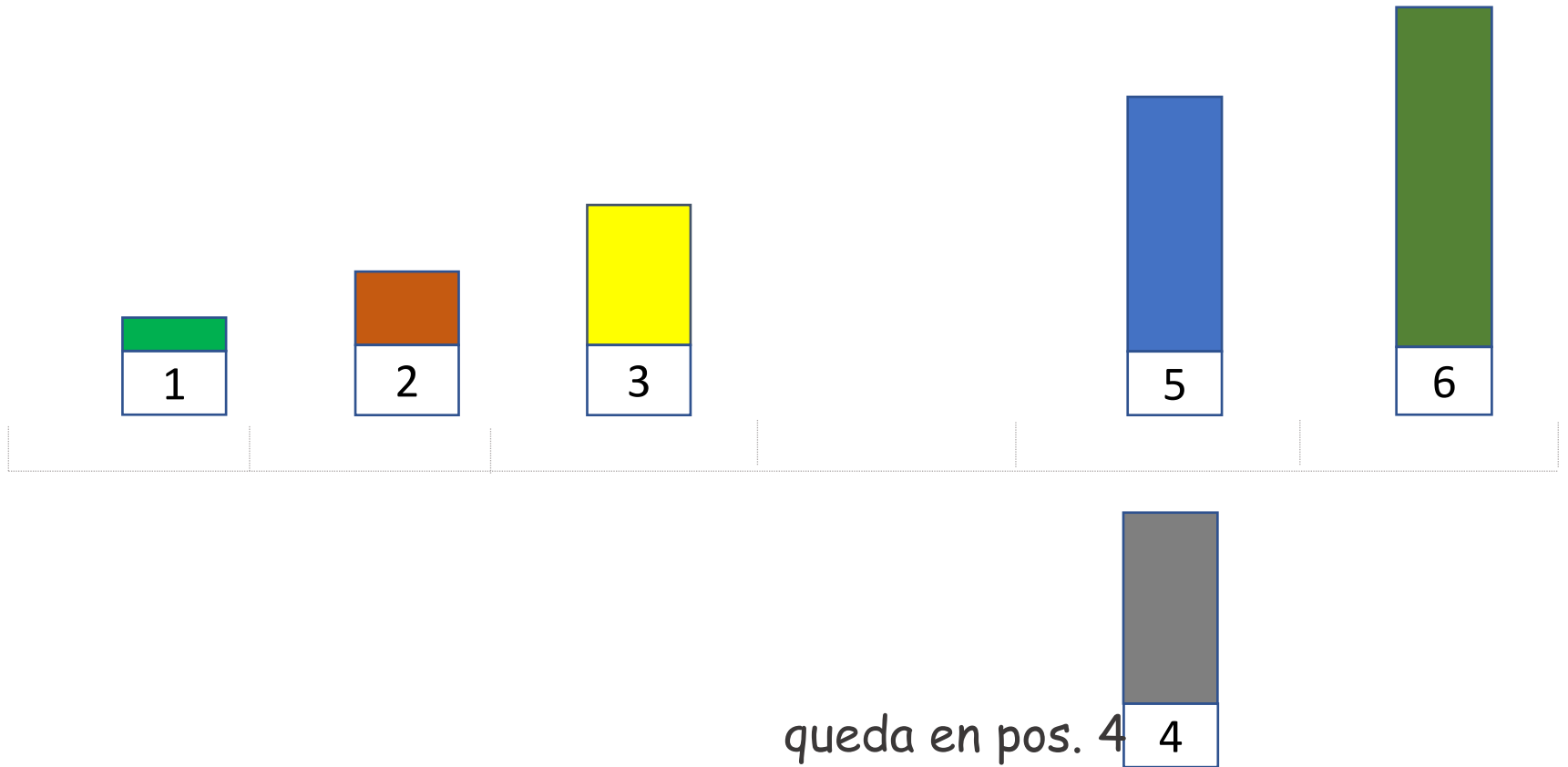
$i=5$ elemento a ordenar = 4



2. MÉTODO DE ORDENACIÓN POR INSERCIÓN

EJEMPLO: Ordenando de menor a mayor...

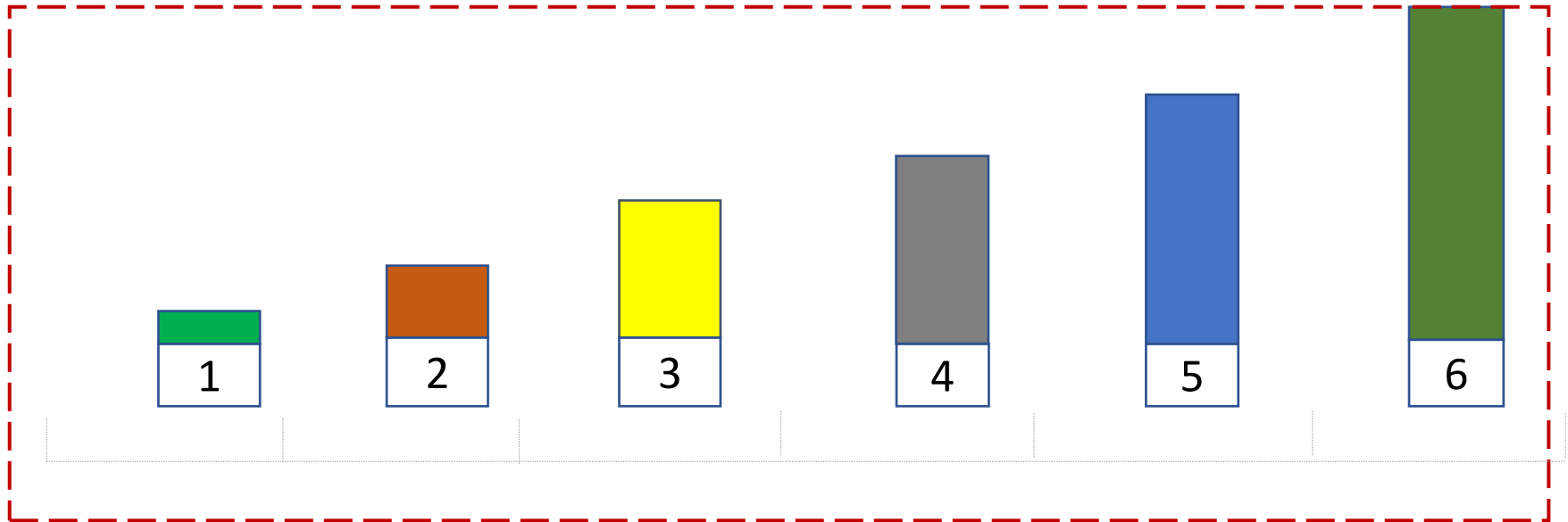
$i=5$ elemento a ordenar = 4



2. MÉTODO DE ORDENACIÓN POR INSERCIÓN

EJEMPLO: Ordenando de menor a mayor...

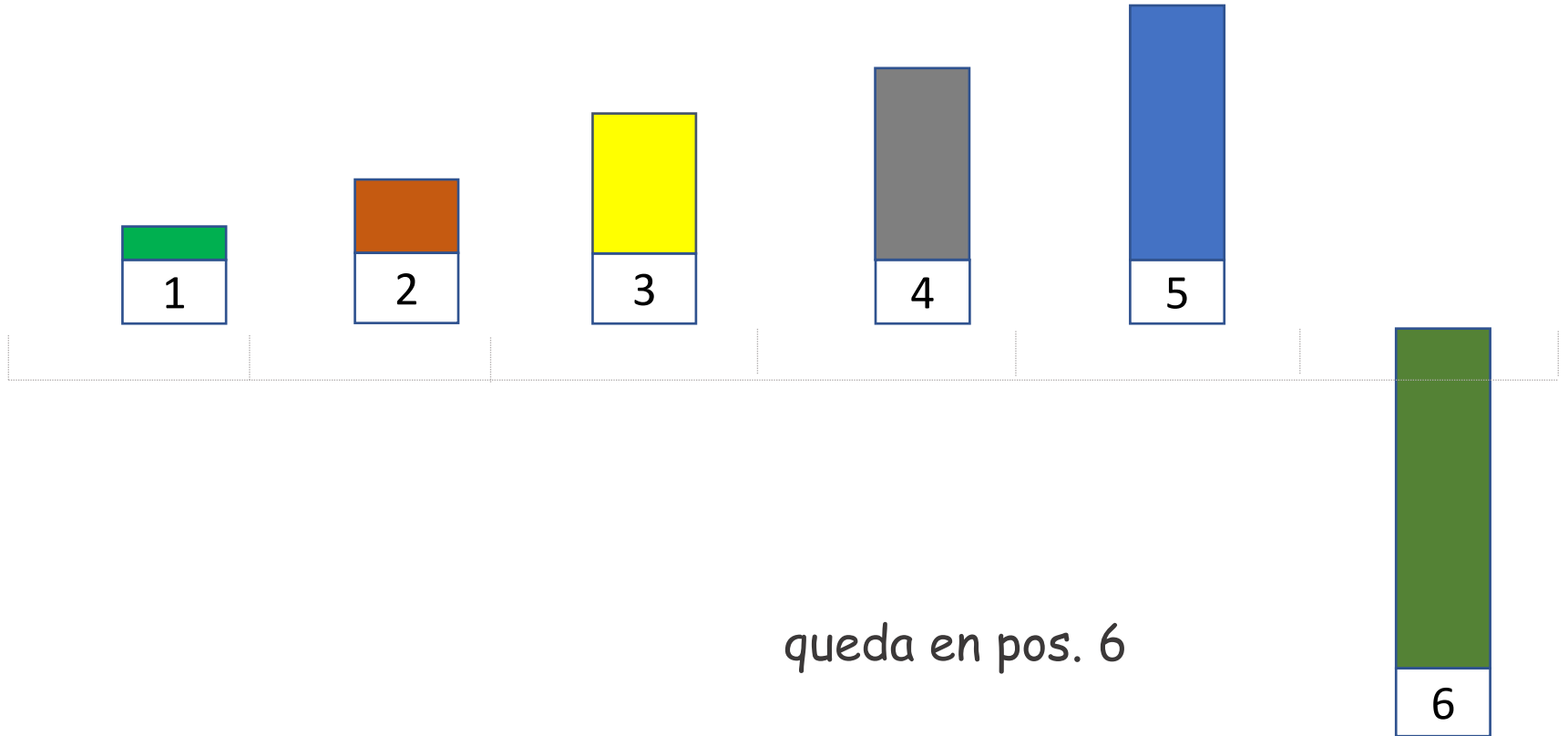
$i=6$ elemento a ordenar = 6



2. MÉTODO DE ORDENACIÓN POR INSERCIÓN

EJEMPLO: Ordenando de menor a mayor...

$i=6$ elemento a ordenar = 6



2. MÉTODO DE ORDENACIÓN POR INSERCIÓN

2.1 PSEUCODIGO: Ordenar vector de n elementos de menor a mayor
(n: dimensión lógica)

```
Repetir desde  $i:=2$  hasta  $n$   
     $actual := v[i]$       // Guardo elemento a ordenar  
     $j := i - 1$   
    Mientras ( $j > 0$ ) y ( $v[j] > actual$ )  
         $v[j+1] := v[j]$   
         $j := j - 1$   
     $v[j+1] := actual$   // Guardar elemento en  $v[j+1]$ 
```


2. MÉTODO DE ORDENACIÓN POR INSERCIÓN

2.2 Análisis de casos

Link al simulador de ordenación por inserción

http://lwh.free.fr/pages/algo/tri/tri_insertion_es.html

```
Repetir desde i:=2 hasta n
    actual:= v[i]      // Guardo elemento a ordenar
    j:= i-1
    Mientras (j > 0) y (v[j] > actual)
        v[j+1]:= v[j]
        j:= j - 1
    v[j+1]:= actual    // Guardar elemento en v[j+1]
```

¿Qué pasa si los datos están ordenados de menor a mayor ?

1 | 2 | 3 | 4 | 5 | 6

¿Qué pasa si los datos están ordenados de mayor a menor ?

6 | 5 | 4 | 3 | 2 | 1

¿Qué modificaciones haría en este algoritmo si quisiera ordenar un vector de registros?

2. MÉTODO DE ORDENACIÓN POR INSERCIÓN

2.3 Análisis teórico

```
Repetir desde i:=2 hasta n
  actual:= v[i]      // Guardo elemento a ordenar
  j:= i-1
  Mientras (j > 0) y (v[j] > actual)
    v[j+1]:= v[j]
    j:= j - 1
  v[j+1]:= actual    // Guardar elemento en v[j+1]
```

C: Nro comparaciones (de elementos) **M**: Nro de asignaciones de elementos

Mejor caso

$O(n)$

$$n-1 \leq C \leq n(n-1)/2$$

$$2(n-1) \leq M \leq 2(n-1) + n(n-1)/2$$

Peor caso

$O(n^2)$



Ejercitación

ACTIVIDAD 1: En el **ProgramaVector.pas** (creado en la clase previa)

1. Implementar el módulo **OrdenaciónPorInserción** (use el pseudocódigo)
2. Invocar al módulo **OrdenacionPorInsercion**
3. Mostrar el vector ordenado
4. Implementar el módulo **BuscarElementoVectorOrdenado** que busque de manera eficiente un valor X sobre el vector ordenado y devuelva su posición o 0 en caso de no existir. Para ello utilice Búsqueda Binaria (o dicotómica).
5. Lea un número e informe su posición en el vector o 0 si no existe.



Ejercitación

ACTIVIDAD 2

Implementar un programa que procese la información de los jugadores de un torneo de básquet. De cada jugador se lee: dni, nombre y altura. El ingreso de los jugadores finaliza cuando se lee el dni 0 (que no debe procesarse). Se pide:

- a. Cargar la información en una lista. Los jugadores deben quedar en el mismo orden que fueron leídos.
- b. Mostrar la información almacenada en la lista.
- c. Generar un vector con aquellos jugadores que superan el promedio de altura. Se sabe de antemano que como máximo son 20.
- d. Mostrar la información almacenada en el vector.
- e. Ordenar el vector de jugadores por dni (ascendente)
- f. Mostrar el vector ordenado.
- g. Buscar en el vector al jugador cuyo dni coincide con uno ingresado e imprimir su altura.

NOTA: Reutilice los módulos implementados con anterioridad.



Ejercitación

ACTIVIDAD 3: Implementar **programaVECTORDELISTAS.pas** para resolver el siguiente problema:

Se cuenta con información de los empleados de una empresa. De cada empleado se conoce su número de empleado, apellido_y_nombre, antigüedad en años y categoría (1..4).

El programa debe:

- a. Leer los datos de empleados hasta que se ingresa el nro. de empleado 0 y guardarlos ordenados alfabéticamente por apellido_y_nombre y agrupados por categoría.
- b. Una vez guardados (implemente un módulo para cada inciso)
 - i. Para cada categoría, mostrar la información de todos sus empleados.
 - ii. Generar un vector que contenga, para cada categoría, el empleado más antiguo.
 - iii. Ordenar el vector por antigüedad (descendente) e informar su contenido.

NOTA: Reutilice los módulos implementados con anterioridad.