



CICLO 01

[FORMACIÓN POR CICLOS]

Fundamentos de Programación

Colección de datos
Inserción en vector
ordenado



Operación de inserción en vector ordenado

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
V	13	3	8	12	16	23	28	39	45	50	53	68	74	81											

Si se quiere insertar el 25 se detecta, por inspección, que debe quedar en la posición 6 y los datos conservan la propiedad de que están ordenados ascendentemente. Lo anterior implica que hay que mover los datos desde la posición 6 hasta la 13 una posición a la derecha.

En general, llamemos **k** la posición donde debe quedar el dato a insertar.

Hay que mover los datos desde la posición **k** hasta la posición **V[0]** una posición a la derecha. Asignar el dato a insertar a la posición **k** y actualizar la posición **V[0]**.

Para determinar **k** basta con recorrer el vector a partir de la posición **1** hasta encontrar un dato que sea mayor que el dato a insertar. Esa es la posición **k**.

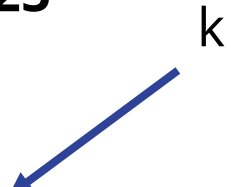
Tendremos dos algoritmos: uno que llamaremos **buscaDondeInsertar**, con el cual se determina **k**. Al otro lo llamaremos **insertar**, el cual moverá los datos desde la posición **k** hasta **V[0]** y asignará el dato a la posición **k**.

d = 25

V

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
13	3	8	12	16	23	28	39	45	50	53	68	74	81											

k



Nuestros algoritmos son:

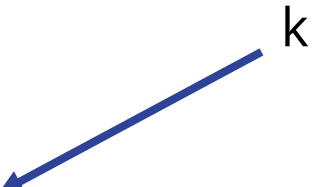
```
def buscarDondeInsertar(V, d):  
    k = 1  
    while k <= V[0] and V[k] < d:  
        k = k + 1  
    return k
```

d = 25

V

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
13	3	8	12	16	23	28	39	45	50	53	68	74	81											

k



Nuestros algoritmos son:

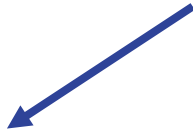
```
def buscarDondeInsertar(V, d):  
    k = 1  
    while k <= V[0] and V[k] < d:  
        k = k + 1  
    return k
```

d = 25

V

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
13	3	8	12	16	23	28	39	45	50	53	68	74	81											

k



Nuestros algoritmos son:

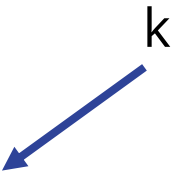
```
def buscarDondeInsertar(V, d):  
    k = 1  
    while k <= V[0] and V[k] < d:  
        k = k + 1  
    return k
```

d = 25

V

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
13	3	8	12	16	23	28	39	45	50	53	68	74	81											

k



Nuestros algoritmos son:

```
def buscarDondeInsertar(V, d):  
    k = 1  
    while k <= V[0] and V[k] < d:  
        k = k + 1  
    return k
```

d = 25

k
↙

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
V	13	3	8	12	16	23	28	39	45	50	53	68	74	81											

Nuestros algoritmos son:

```
def buscarDondeInsertar(V, d):  
    k = 1  
    while k <= V[0] and V[k] < d:  
        k = k + 1  
    return k
```


d = 25

k
↓

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
V	13	3	8	12	16	23	28	39	45	50	53	68	74	81											

Nuestros algoritmos son:

```
def buscarDondeInsertar(V, d):  
    k = 1  
    while k <= V[0] and V[k] < d:  
        k = k + 1  
    return k
```

d = 25

V

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
13	3	8	12	16	23	28	39	45	50	53	68	74	81											

k ↓

i ↘

```
def insertar(V, d, k):  
    for i in range(V[0], k - 1, -1):  
        V[i+1] = V[i]  
    V[k] = d  
    V[0] = V[0] + 1
```

d = 25

k
↓
i
↓

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
V	13	3	8	12	16	23	28	39	45	50	53	68	74	81	81									

```
def insertar(V, d, k):  
    for i in range(V[0], k - 1, -1):  
        V[i+1] = V[i]  
    V[k] = d  
    V[0] = V[0] + 1
```

d = 25

V

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
13	3	8	12	16	23	28	39	45	50	53	68	74	74	81										

k (points to index 6)
i (points to index 11)

```
def insertar(V, d, k):  
    for i in range(V[0], k - 1, -1):  
        V[i+1] = V[i]  
    V[k] = d  
    V[0] = V[0] + 1
```

d = 25

V

0	1	2	3	4	5	k	7	8	9	10	i	12	13	14	15	16	17	18	19	20	21	22	23	24
13	3	8	12	16	23	28	39	45	50	53	68	68	74	81										

```
def insertar(V, d, k):  
    for i in range(V[0], k - 1, -1):  
        V[i+1] = V[i]  
    V[k] = d  
    V[0] = V[0] + 1
```

d = 25

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
V	13	3	8	12	16	23	28	39	45	50	53	53	68	74	81										

A diagram illustrating an array V of size 25. The array is represented as a horizontal row of 25 cells. The indices 0 through 24 are written above the cells. The values 13, 3, 8, 12, 16, 23, 28, 39, 45, 50, 53, 53, 68, 74, 81 are written inside the first 15 cells. The remaining 10 cells are empty. A blue arrow labeled 'k' points to the cell at index 6, which contains the value 28. An orange arrow labeled 'i' points to the cell at index 9, which contains the value 50.

```
def insertar(V, d, k):  
    for i in range(V[0], k - 1, -1 ):  
        V[i + 1] = V[i]  
    V[k] = d  
    V[0] = V[0] + 1
```

d = 25

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
V	13	3	8	12	16	23	28	39	45	50	50	53	68	74	81										

A diagram illustrating an array V of size 25. The array is shown as a horizontal row of 25 cells. The first 15 cells contain the values 13, 3, 8, 12, 16, 23, 28, 39, 45, 50, 50, 53, 68, 74, and 81. The remaining 10 cells are empty. Above the array, the indices 0 through 24 are labeled. A blue arrow labeled 'k' points to the cell at index 6, which contains the value 28. An orange arrow labeled 'i' points to the cell at index 8, which contains the value 45.

```
def insertar(V, d, k):  
    for i in range(V[0], k - 1, -1):  
        V[i + 1] = V[i]  
    V[k] = d  
    V[0] = V[0] + 1
```

d = 25

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
V	13	3	8	12	16	23	28	39	45	45	50	53	68	74	81										

A diagram illustrating an array V of size 25. The array is represented as a horizontal row of 25 cells. The first 15 cells contain the values 13, 3, 8, 12, 16, 23, 28, 39, 45, 45, 50, 53, 68, 74, and 81. The remaining 10 cells are empty. Above the array, the indices 0 through 24 are labeled. A blue arrow labeled 'k' points to the cell at index 6, which contains the value 28. An orange arrow labeled 'i' points to the cell at index 7, which contains the value 39.

```
def insertar(V, d, k):  
    for i in range(V[0], k - 1, -1 ):  
        V[i + 1] = V[i]  
    V[k] = d  
    V[0] = V[0] + 1
```


d = 25

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
V	13	3	8	12	16	23	28	39	39	45	50	53	68	74	81										

Diagram illustrating an array V of size 25. The array contains values: 13, 3, 8, 12, 16, 23, 28, 39, 39, 45, 50, 53, 68, 74, 81, followed by 10 empty slots. A blue arrow labeled 'k' points to index 6 (value 28). An orange arrow labeled 'i' points to index 7 (value 39).

```
def insertar(V, d, k):  
    for i in range(V[0], k - 1, -1):  
        V[i + 1] = V[i]  
    V[k] = d  
    V[0] = V[0] + 1
```

d = 25

i k



V

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
13	3	8	12	16	23	28	28	39	45	50	53	68	74	81										

```
def insertar(V, d, k):  
    for i in range(V[0], k - 1, -1):  
        V[i + 1] = V[i]  
    V[k] = d  
    V[0] = V[0] + 1
```

d = 25

k



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
V	13	3	8	12	16	23	25	28	39	45	50	53	68	74	81										

```
def insertar(V, d, k):  
    for i in range(V[0], k - 1, -1):  
        V[i + 1] = V[i]  
    V[k] = d  
    V[0] = V[0] + 1
```

Nuestro vector queda así:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
V	14	3	8	12	16	23	25	28	39	45	50	53	68	74	81										