



CICLO 1

[FORMACIÓN POR CICLOS]

Fundamentos de **PROGRAMACIÓN**



Ingeni@
Soluciones TIC



UNIVERSIDAD
DE ANTIOQUIA

Facultad de Ingeniería

Lectura

DECISIONES

anidadas



Elaborar algoritmos más complejos utilizando la instrucción **if** con su componente opcional **else**.

Elabore un algoritmo que lea tres datos numéricos enteros y que los imprima ordenados ascendenteamente.

Análisis:

1. Datos de entrada: tres datos numéricos enteros: **a**, **b**, **c**.
2. Cálculos: determinar el menor de los tres datos para imprimirla primero, luego determinar el menor de los otros dos para imprimirla de segundo y luego imprimir el tercer dato.
3. Datos de salida: los mismos tres datos ordenados ascendenteamente.

Una primera forma de escribir este algoritmo es siendo exhaustivos en la comparación de los datos.

Realmente, las diferentes situaciones que se pueden presentar para escribir los tres datos son:

- La primera, cuando **a** es menor que **b** y **b** es menor que **c**.
La segunda, cuando **a** es menor que **c** y **c** es menor que **b**.
La tercera, cuando **b** es menor que **a** y **a** es menor que **c**.

Y así sucesivamente.

A cada situación le corresponde una relación de orden diferente.

1. a, b, c
2. a, c, b
3. b, a, c
4. b, c, a
5. c, a, b
6. c, b, a

Por consiguiente, nuestro programa deberá escribir solo una de ellas.

Un algoritmo en Python para efectuar esa tarea es:

```
a = int(input("Entre un número entero: "))
b = int(input("Entre otro número entero: "))
c = int(input("Entre un tercer número entero: "))
if a < b and b < c:
    print("a", a, "b", b, "c", c)
if a < c and c < b:
    print("a", a, "c", c, "b", b)
if b < a and a < c:
    print("b", b, "a", a, "c", c)
if b < c and c < a:
    print("b", b, "c", c, "a", a)
if c < a and a < b:
    print("c", c, "a", a, "b", b)
if c < b and b < a:
    print("c", c, "b", b, "a", a)
```

Uno de los inconvenientes de este algoritmo es que cuando una situación sea verdadera, continúa preguntando por las demás situaciones, lo cual genera ineficiencia. Sin embargo, el inconveniente más grande que tiene ese algoritmo es que si hay al menos dos datos iguales no imprime nada.

¿Cómo solucionar esto? Podríamos pensar que poniendo en las comparaciones menor o igual (\leq) en vez de únicamente menor ($<$). Nuestro algoritmo será:

```
a = int(input("Entre un número entero: "))
b = int(input("Entre otro número entero: "))
c = int(input("Entre un tercer número entero: "))
if a <= b and b <= c:
    print("a", a, "b", b, "c", c)
if a <= c and c <= b:
    print("a", a, "c", c, "b", b)
if b <= a and a <= c:
    print("b", b, "a", a, "c", c)
if b <= c and c <= a:
    print("b", b, "c", c, "a", a)
if c <= a and a <= b:
    print("c", c, "a", a, "b", b)
if c <= b and b <= a:
    print("c", c, "b", b, "a", a)
```



Esta solución tampoco es buena, puesto que cuando haya dos datos iguales imprimiría dos veces, y, peor aún, cuando los tres datos sean iguales imprimiría los datos seis veces.

Para solucionar los problemas antes relacionados utilizamos el componente else. Veamos cómo queda nuestro programa:

```
a= int(input("Entre un número entero: "))
b = int(input("Entre otro número entero: "))
c = int(input("Entre un tercer número entero: "))
if a <= b and b <= c:
    print("a", a, "b", b, "c", c)
else:
    if a <= c and c <= b:
        print("a", a, "c", c, "b", b)
    else:
        if b <= a and a <= c:
            print("b", b, "a", a, "c", c)
        else:
            if b <= c and c <= a:
                print("b", b, "c", c, "a", a)
            else:
                if c <= a and a <= b:
                    print("c", c, "a", a, "b", b)
                else:
                    print("c", c, "b", b, "a", a)
```

De esta manera, cuando encuentre que una condición (situación) es verdadera, procede a imprimir los datos en forma ordenada y no sigue preguntando por las demás condiciones.

Una tercera forma en que podemos elaborar el algoritmo es la siguiente:

Comparamos **a** con **b**. Pueden suceder dos situaciones: una, que **a** sea menor que **b**, y dos, que **b** sea menor que **a**.

1. Si **a** es menor que **b**, implica que habrá que escribir el dato **a** antes que el dato **b**; por tanto, las posibles formas de escribir los tres datos son:

- 1.** a, b, c
- 2.** a, c, b
- 3.** c, a, b

Si **b** es menor que **c**, imprimimos la primera posibilidad: a, b, c; de lo contrario, debemos comparar **a** con **c**.

Si **a** es menor que **c**, imprimimos la segunda posibilidad: a, c, b; de lo contrario, imprimimos la tercera posibilidad: c, a, b.

2. Si **b** es menor que **a**, implica que habrá que escribir el dato **b** antes que el dato **a**; por tanto, las posibles formas de escribir los tres datos son:

- 1.** b, a, c
- 2.** b, c, a
- 3.** c, b, a

Si **a** es menor que **c**, imprimimos la primera posibilidad: b, a, c; de lo contrario, debemos comparar **b** con **c**.

Si **b** es menor que **c**, imprimimos la segunda posibilidad: b, c, a; de lo contrario, imprimimos la tercera posibilidad: c, b, a.

Con base en el anterior análisis, nuestro programa queda:

```
a = int(input("Entre un número entero: "))
b = int(input("Entre otro número entero: "))
c = int(input("Entre un tercer número entero: "))
if a < b:
    if b < c:
        print("a", a, "b", b, "c", c)
    else:
        if a < c:
            print("a", a, "c", c, "b", b)
        else:
            print("c", c, "a", a, "b", b)
    else:
        if a < c:
            print("b", b, "a", a, "c", c)
        else:
            print("c", c, "b", b, "a", a)
```

Observe que utilizando la instrucción **else** de esta forma, basta con evaluar la pregunta con el operador menor que (<).