

Guía de ejercicios prácticos

A continuación, se plantean una serie de fragmentos de códigos. Habrá que realizar un análisis de algoritmo para determinar su coste computacional y obtener su orden.

1. Fragmento de código condicional:

```
if num > 10:
    print('es mayor que 10')
    print('su cuadrado es 100')
    aux = num % 2
else:
    print('es menor que 10')
    print('su cubo es 1000')
    aux = num // 2
    print(aux)
    producto = num * 2
```

2. Fragmento de código condicional:

```
if num > 10:
    print('es mayor que 10')
    print('su cuadrado es 100')
    aux = num % 2
    if(num % 2 == 0):
        print('es divisible por 2')
    else:
        print('es divisible por 2')
        num = num + 1
        aux = num * num * num
else:
    print('es menor que 10')
    print('su cubo es 1000')
    aux = num // 2
    print(aux)
    producto = num * 2
    if(producto % 3 == 0):
        print('es divisible por 3')
```

3. Fragmento de código de ciclo:

```
for i in range(0, n):
    print('inicio ciclo j')
    for j in range(0, m):
        print('inicio ciclo k')
        for k in range(0, 3):
            print(lista[k])
```

4. Fragmento de código de ciclo:

```
for i in range(0, n):
    if(aux is not None):
        print('inicio ciclo j')
        for j in range(0, m):
            print(num + (j * valor))
    else:
        print('inicio ciclo k')
        for k in range(0, 1000):
            print(aux[k] + num)
```

5. Fragmento de código condicional:

```
if num > 10:
    print('es mayor que 10')
    print('su cuadrado es 100')
    aux = num % 2
    print('tabla de multiplicar')
    for i in range(1, 16):
        print(i, num * i)
else:
    print('es menor que 10')
    print('su cubo es 1000')
    aux = num // 2
    print(aux)
    producto = num * 2
    if(producto % 3 == 0):
        print('es divisible por 3')
```

6. Fragmento de código:

```
numero = int(input('ingrese un número'))
while (numero != 0) and (len(lista) < 10000):
    lista.append(numero)
    numero = int(input('ingrese número'))

for i in range(0, len(lista)):
    print(lista[i])
```

7. Búsqueda binaria iterativa:

```
while (p ≤ u) and (pos = -1):
    med = (p + u) // 2
    if (lista[med] = x):
        pos = med
    else:
        if(x > lista[med]):
            p = med + 1
        else:
            p = med - 1
```

8. Fragmento de código (mezcla de listas):

```
i = 0
j = 0

while (i < len(L1)) and (j < len(L2)):
    if(L1[i] < L2[j]):
        L3.append(L1[i])
        i += 1
    else:
        L3.append(L2[j])
        j += 1

if(i = len(L1)):
    for k in range(j, len(L2)):
        L3.append(L2[k])
else:
    for k in range(j, len(L1)):
        L3.append(L1[k])
```

9. Fragmento de código:

```
for i in range(0, n):
    ac = 0
    for j in range(0, m):
        ac = ac + lista[i, j]
    aux[i] = ac

i = 0
while (i < len(aux)):
    print(aux[i])
    i += 1
```

10. Fragmento de código:

11. Código de multiplicación de dos matrices $M1[n \times m]$ y $M2[m \times o]$:
12. Código de suma de dos matrices de $[n \times n]$.
13. Código para calcular la traza de una matriz cuadrada.
14. Código para calcular el determinante de una matriz cuadrada de $[3 \times 3]$, regla de Sarrus.
15. Función para determinar si un número es primo.
16. Función factorial iterativa.
17. Función Fibonacci iterativa.