Emilio Azael Valencia López

Estructura de Datos 12 de septiembre de 2025

**Inventario de Productos en un Almacén**

# Introducción

Un almacén necesita controlar el inventario de 3 productos. La información se organiza en una matriz donde cada fila representa un producto y las columnas son su ID, cantidad en stock y precio unitario. Requerimos crear un programa que imprima los datos del inventario, luego, debe calcular y mostrar el valor total de un producto específico (cantidad × precio). Finalmente, debe actualizar la cantidad en stock de ese producto después de vender 10 unidades.

# Desarrollo

## Algoritmo

1. **Crear la matriz de inventario**

* Se define una matriz donde cada fila representa un producto.
* Cada producto tiene 3 datos:
  + ID del producto
  + Stock (cantidad en existencia)
  + Precio unitario

1. **Imprimir el inventario completo**

* Se recorre la matriz con dos ciclos for anidados:
  + El primero para las filas (productos).
  + El segundo para las columnas (atributos de cada producto).
* Se muestra en pantalla una tabla con el ID, stock y precio de cada producto.

1. **Buscar un producto específico**

* Se define un product\_id (ej. 31).
* Se recorre la matriz y se compara la primera columna de cada fila (ID).
* Si coincide, se calcula el valor total en stock:

***valor total = stock × precio unitario***

* Se imprime el valor calculado.

1. **Actualizar inventario tras una venta**

* Se simula la venta de 10 unidades del producto encontrado.
* Se resta al stock.
* Se recalcula el nuevo valor total.
* Se imprime el valor actualizado del producto.

## Código en Python

# Crear una matriz para almacenar los datos de 3 productos

matriz = [

    [19, 400, 50],

    [31, 150, 200],

    [33, 950, 22]

]

# Imprimir los datos del inventario

print("Inventario total:\n ID Stock Precio")

for f in range(len(matriz)):

    for c in range(len(matriz[f])):

        print(f" {matriz[f][c]} ", end="")

    print()

# Calcular valor de un producto especifico

product\_id = 31

for f in range(len(matriz)):

    if matriz[f][0] == product\_id:

        valor\_total = matriz[f][1] \* matriz[f][2]

        print(f"Valor total del producto {product\_id}: ${valor\_total}")

# Vender 10 unidades del producto y actualizarlo

for f in range(len(matriz)):

    if matriz[f][0] == product\_id:

        matriz[f][1] -= 10

        valor\_total = matriz[f][1] \* matriz[f][2]

        print(f"Valor total del producto {product\_id} (Actualizado): ${valor\_total}")

# Conclusión

Este ejercicio muestra cómo las **matrices** son útiles para almacenar y manipular datos estructurados de varios elementos al mismo tiempo, como un inventario con múltiples productos y atributos. Gracias a la organización en filas y columnas, se pueden realizar operaciones como búsquedas, cálculos de valor total y actualizaciones de manera ordenada y eficiente. Con este ejemplo se comprende la importancia de las estructuras de datos bidimensionales en la programación, ya que permiten modelar problemas reales (como la gestión de inventarios) y facilitan la construcción de algoritmos claros y prácticos para su control y administración.