# Documentación del Programa: Cola y Pila con Nodos

## 1. Objetivo del Programa

El objetivo principal de este programa es gestionar una cola de elementos de tal manera que los valores en posiciones **pares** (0, 2, 4, ...) permanezcan en la cola original, mientras que aquellos ubicados en **posiciones impares** (1, 3, 5, ...) sean transferidos a una pila implementada con nodos enlazados. La operación debe realizarse en **una sola pasada** por la cola, optimizando el tiempo de procesamiento y preservando el orden lógico de las estructuras.

---

## 2. Estructura del Código

El proyecto está compuesto por tres módulos principales:

- **`Pila.py`**: Contiene la clase `Pila` y `Nodo`, ambos implementados utilizando nodos enlazados.

- **`Cola.py`**: Implementa la clase `Cola` mediante nodos enlazados, junto con funcionalidades de recorrido.

- **`main.py`**: Actúa como el módulo principal, donde se implementa un menú interactivo y se coordina la lógica del programa.

---

## 3. Implementación de Clases

### 3.1 Clase Nodo (usado en `Pila.py` y `Cola.py`)

**Propósito**: Representar cada elemento de la pila o cola como un nodo enlazado.
**Atributos**:

- `dato`: Valor del nodo.
- `siguiente`: Referencia al siguiente nodo en la estructura.

---

### 3.2 Clase Pila (en `Pila.py`)

**Propósito**: Representa una **estructura de tipo LIFO** (Last In, First Out) utilizando nodos.
**Atributos**:

- `peek`: Nodo superior de la pila.

  **Métodos Principales**:

- `apilar(dato)`: Agrega un nodo a la cima.
- `imprimir_pila()`: Muestra los elementos de la pila desde el tope hasta la base.

---

### 3.3 Clase Cola (en `Cola.py`)

**Propósito**: Representa una **estructura de tipo FIFO** (First In, First Out).
**Atributos**:

- `frente`: Primer nodo.
- `final`: Último nodo.

- `actual`: Nodo temporal para iteraciones.

  **Métodos Principales**:

- `empujar(dato)`: Añade un nuevo nodo al final.

- `desencolar()`: Elimina y devuelve el primer nodo.

- `recorrer_cola()`: Recorre la cola y transfiere elementos en posiciones impares a una pila.

- `imprimir_cola()`: Muestra los elementos desde el frente hasta el final.

---

## 4. Algoritmo de Separación (`recorrer_cola`)

**Entrada**: Cola con elementos agregados por el usuario.
**Salida**: Una pila que contiene los elementos de **índices impares**.
**Modificación**: La cola queda con los elementos en **posiciones pares**.
**Lógica**:

1. Se inicializa una pila vacía.

2. Se recorre la cola desde el frente.

3. Por cada nodo, se verifica su posición:

    - Si es impar, se apila.

    - Si es par, se conserva.

4. Se continúa hasta vaciar la cola original.

5. Se reencolan los elementos pares para restaurar el orden FIFO.

---

## 5. Cómo Ejecutar el Programa

1. Asegúrate de tener los archivos `main.py`, `Cola.py` y `Pila.py` en el mismo directorio.

2. Abre una terminal o consola.

3. Navega a la ubicación del proyecto.

4. Ejecuta el script principal con el comando:

bash
CopiarEditar
```
python main.py
```

5. Utiliza el menú para:

    - Ingresar datos en la cola.

    - Recorrer y separar la cola.

    - Imprimir los contenidos de la cola o la pila.

---

## 6. Salida Esperada de los Ejemplos

plaintext

CopiarEditar

```
--- Menú ---
1. Agregar dato a la cola
2. Recorrer la cola y separar impares en pila
3. Imprimir cola
4. Imprimir pila
5. Salir
```

**Supuesto:**

Entrada:

```
Cola: [10, 20, 30, 40, 50, 60]
```

Procesamiento:

- Pares (índices 0, 2, 4): 10, 30, 50 → permanecen en la cola.
- Impares (índices 1, 3, 5): 20, 40, 60 → se apilan.

**Resultado:**

plaintext

CopiarEditar

```
Cola resultante (pares): [10, 30, 50]
Pila resultante (impares): [60, 40, 20]
```