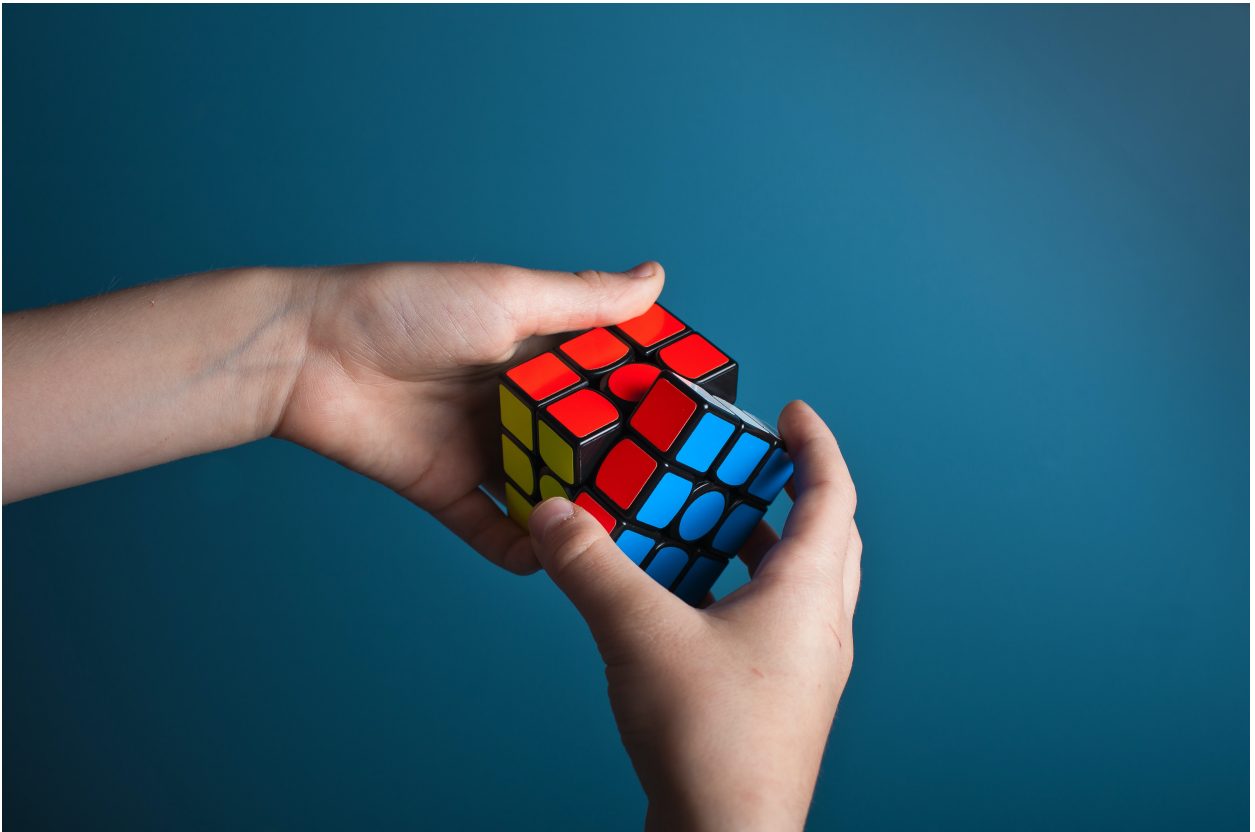




Solución al problema

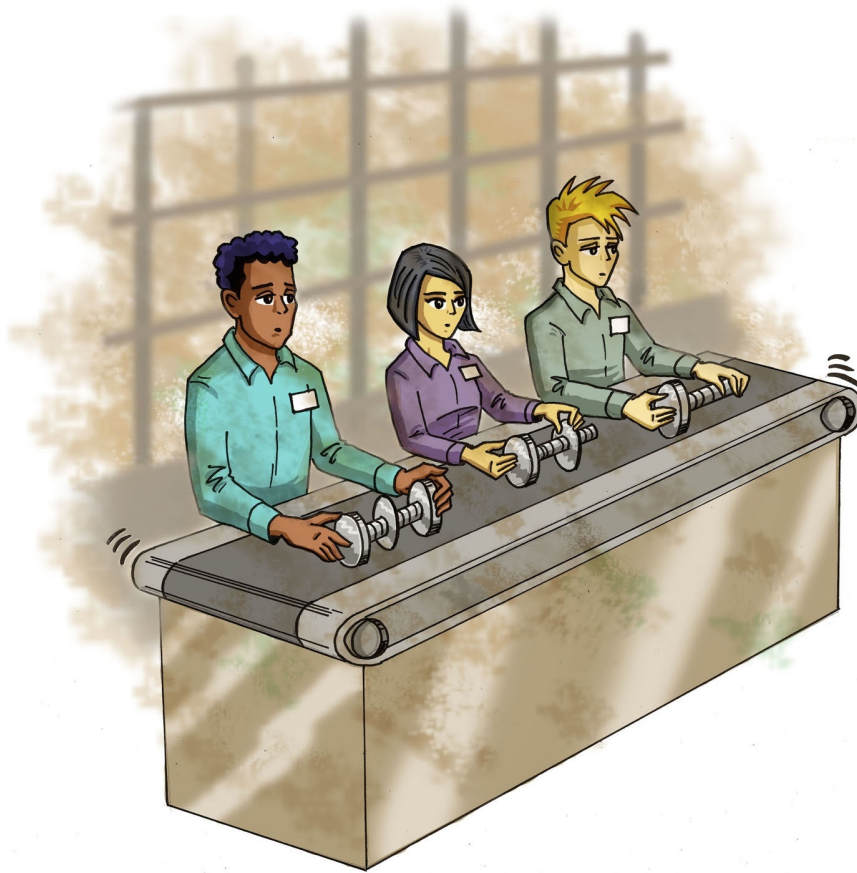


Descripción

La idea de usar un semáforo es evitar que el **productor** intente seguir generando datos en el canal de comunicación (buffer) cuando éste ya se ha llenado. Por otro lado, queremos evitar que el **consumidor** intente obtener datos del canal de comunicación cuando éste este vacío.

Productor

1. Producimos



2. Almacenamos

<https://gph.is/1qEbq8i>

3. Notifica

<https://gph.is/2AV10Uz>

Consumidor

1. Espera

<http://gph.is/2cCs4a6>

2. Adquirir elemento producido

<http://gph.is/2aKPYwr>

3. Notifica y consume

<http://gph.is/1VcV7QX>

Pseudocódigo

```
semaforo.disponible <- tamaño_max
semaforo.ocupado <- 0

wait(semaforo): // operación P
  while semaforo == 0 // proceso se espera en este ciclo hasta que el
                      semáforo sea > 0
    semaforo = semaforo - 1

signal(semaforo): // operación V
  semaforo = semaforo + 1

Productor
while true:
```

```

    item <- producir()
    wait(semaforo.disponible)
    buffer.ingresar(item) // sección crítica
    signal(semaforo.ocupado)

Consumidor
while true:
    wait(semaforo.ocupado)
    item <- buffer.get() // sección crítica
    signal(semaforo.disponible)
    consumir(item)

```

Pseudocódigo 2

```

semaforo.disponible <- tamaño_max
semaforo.ocupado <- 0
semaforo.exclusion <- 1

Productor
while true:
    item <- producir()
    wait(semaforo.disponible)
    wait(semaforo.exclusion)
    buffer.ingresar(item) // sección crítica
    signal(semaforo.exclusion)
    signal(semaforo.ocupado)

Consumidor
while true:
    wait(semaforo.ocupado)
    wait(semaforo.exclusion)
    item <- buffer.get() // sección crítica
    signal(semaforo.exclusion)
    signal(semaforo.disponible)
    consumir(item)

```