



Departamento de Programación  
Facultad de Informática  
Universidad Nacional del Comahue



# Programación Concurrente



*Sincronización – Monitores – Semáforos*  
*Problema del productor/consumidor*  
*Problema de lectores/escritores*

# Sincronización competencia y cooperación

## Mecanismos

Monitores y esperas guardadas

Semáforos binarios y generales

Cerrojos + Variables de condición

Otras posibilidades ...

# Semáforos

Se usan para restringir el número de hilos que pueden acceder a algunos recursos

- **semáforo binario:** gestiona 1 permiso de acceso

void *acquire()* - void *release()*

- **semáforo general:** gestiona N permisos

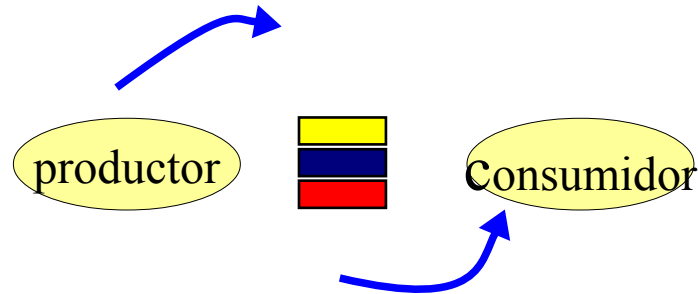
- void *acquire(int n)*

solicita n permisos del semáforo si no hay bastantes, espero y cuando los haya, sigo

- void *release(int n)*

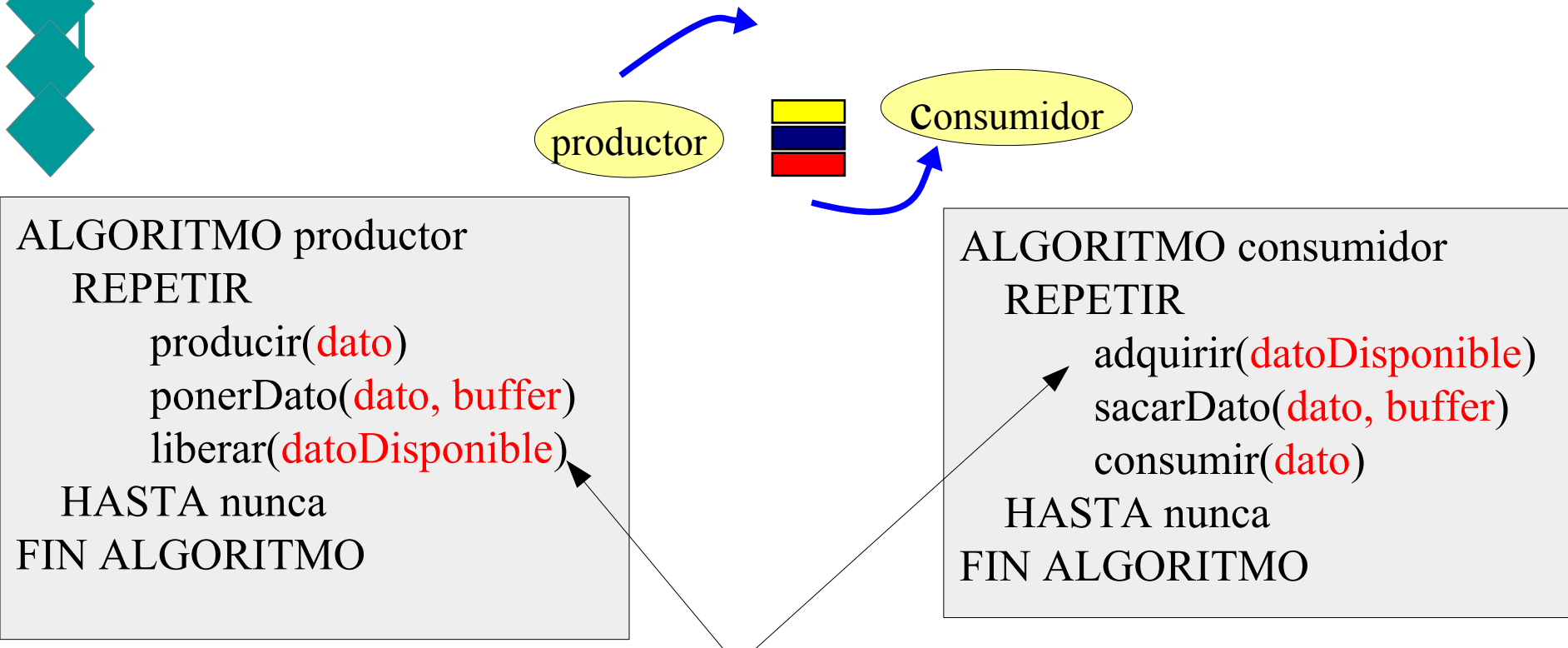
devuelvo n permisos al semáforo si hay alguien esperando, se intenta satisfacerle

# Productor-Consumidor con semáforos



- Problema productor consumidor entre dos procesos
- Para el intercambio de datos se usa un contenedor al cual ambos tienen acceso
- El consumidor debe esperar hasta que el dato haya sido colocado.

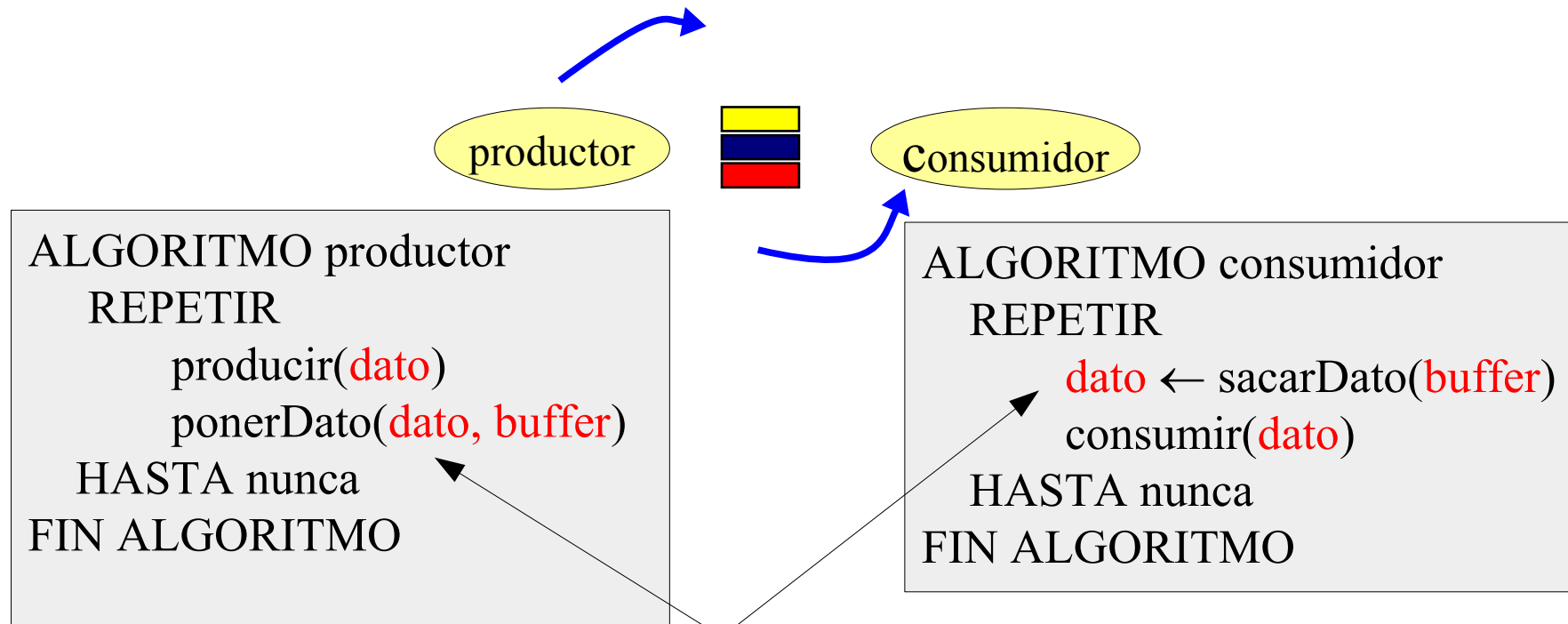
# Productor/Consumidor con semáforos



De la sincronización debería ocuparse el buffer/contenedor

buffer ilimitado → 1 semáforo para coordinar los procesos

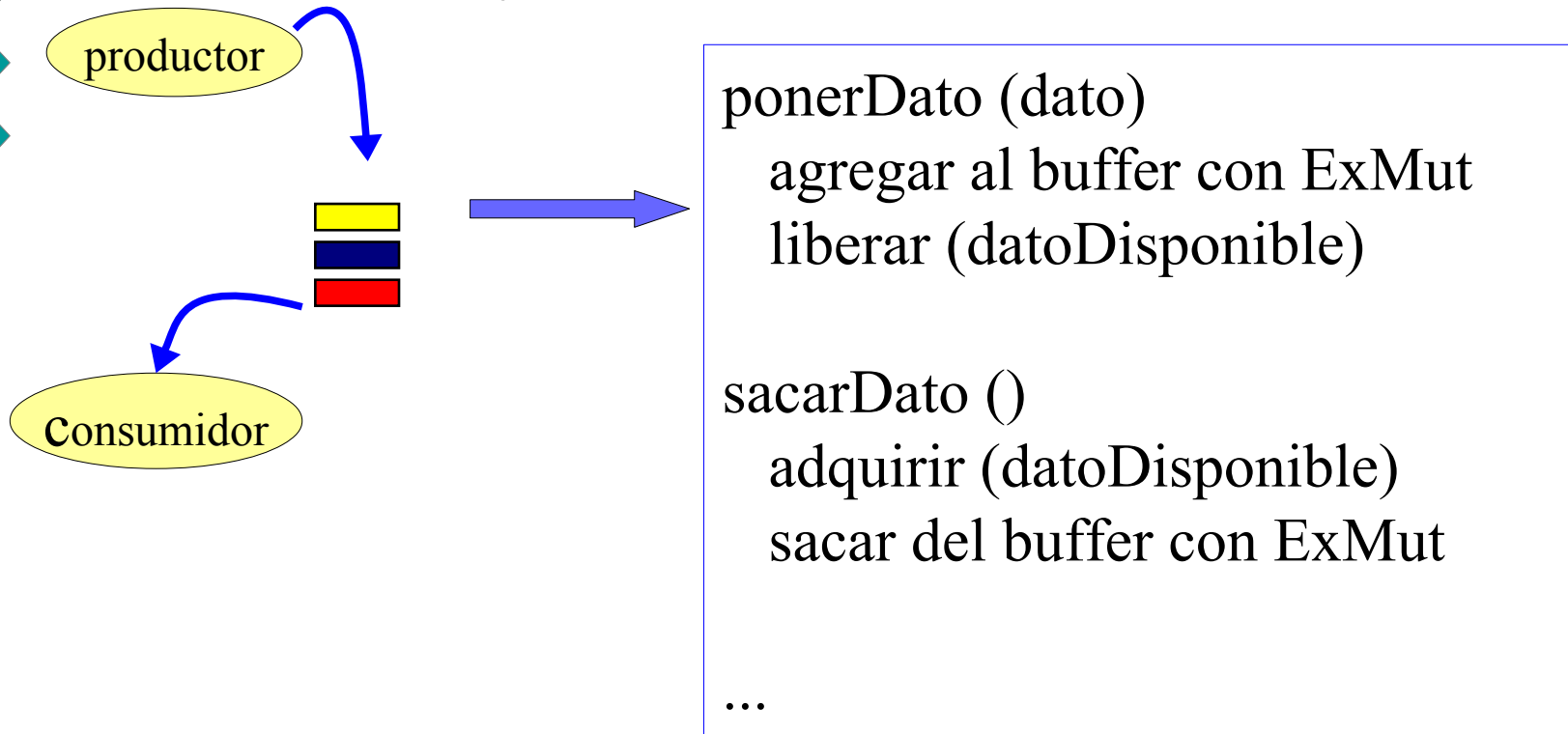
# Productor/Consumidor con semáforos




Operaciones del objeto compartido “buffer”, en las que se debe controlar la exclusion mutua al guardar y tomar un elemento del buffer

*“ponerDato” y “sacarDato” deben trabajar con el semáforo para lograr la cooperación entre productor y consumidor*

# Productor/Consumidor con semáforos y buffer ilimitado



*“ponerDato” y “sacarDato” deben trabajar con el semáforo para lograr la cooperación entre productor y consumidor*



# Productor/consumidor con buffer de tamaño $n$ sincronizados por semáforos

- **mutex:** semáforo binario, proporciona exclusión mutua para el acceso al buffer de productos. - se inicializa a 1.
- **vacio:** semáforo contador para controlar los huecos vacíos del buffer. - se inicializa a  $n$ , tamaño del buffer
- **lleno:** semáforo contador para controlar el número de huecos llenos del buffer. - se *inicializa a 0*.

Hay que tener mucho cuidado en el orden de adquirir y liberar los semáforos, para no provocar deadlock



# Productor/consumidor con buffer de tamaño $n$ sincronizados por semáforos

```
ponerDato (dato) {  
    adquirir(vacio)  
    adquirir(mutex);  
    /* guarda en el buffer */  
    liberar(mutex)  
    liberar(lleno);  
}
```

```
sacarDato () {  
    adquirir(lleno)  
    adquirir(mutex);  
    /* recupera el dato */  
    liberar(mutex)  
    liberar(vacio)  
}
```

- **mutex**, ¿qué sucedería si se inicializa en 0?
- **vacio**, se inicializa en  $n$ , tamaño del buffer
- **lleno**, se *inicializa* en 0.

# Productor/consumidor con buffer de tamaño n sincronizados por semáforos

```
ponerDato (dato) {  
    adquirir(mutex)  
    adquirir(vacio);  
    /* guarda en el buffer */  
    liberar(mutex)  
    liberar(lleno);  
}
```

```
sacarDato () {  
    adquirir(mutex)  
    adquirir(lleno);  
    /* recupera el dato */  
    liberar(mutex)  
    liberar(vacio)  
}
```

- ¿Que sucedería si se tomaran los semáforos en este orden?
- ¿por qué?

# Lectores/escritores

- Un grupo de lectores/escritores quieren tener acceso a un libro.
- Existen varios lectores, varios escritores y un libro con cantidad máxima de páginas
- Cuando un escritor quiere acceder a un libro éste debe estar desocupado.
  - Lector:
    - Puede haber uno o varios lectores leyendo.
    - Si hay un escritor, entonces el lector deberá esperar a que el escritor acabe para poder leer
  - Escritor:
    - Si hay un escritor, entonces el escritor que quiere escribir debe esperar a que no haya nadie leyendo, ni escribiendo.

*Utilizar:*  
*-Semáforo,*  
*-Lector,*  
*-Escritor*

# Hacer los algoritmos....

Clases: **Escritor**, **Lector**, **Libro**

Considerar las operaciones siguientes:

*empezarLeer()*, *terminarLeer()*, *empezarEscribir()*, *terminarEscribir()*,  
*finalizado()*, *hayEscrito()*,

- Datos de interés
  - `int cantiLec = ...` // Cantidad de lectores
  - `int cantiPag = ...` // Escritas
  - `int totalPag = ...` // Cantidad máximas de páginas del libro

# Ejemplo de algoritmos

ALGORITMO lector(libro)

MIENTRAS no terminaLectura HACER

SI libro.hayEscrito() ENTONCES

empezarLeer (libro, lector)

leer()

terminarLeer (libro, lector)

SINO

esperar

FINSI

FIN MIENTRAS

FIN ALGORITMO

Clase Libro

finalizado()

empezarLeer (lector)

empezarEscribir (escritor)

hayEscrito()

terminarEscribir(escritor)

terminarLeer(lector)

ALGORITMO escritor (libro)

MIENTRAS no libro.finalizado() HACER

empezarEscribir(libro, escritor)

escribir()

terminarEscribir(libro, escritor)

FIN MIENTRAS

FIN ALGORITMO



# lectores/escritores

- Objeto compartido: libro
- Con monitores
- Con semáforos
  - Semáforos binarios
  - Semáforos generales



Semáforo: mutex1, mutex2, lectores, escritores;  
Entero: nLectores=0; nEscritores=0

## Semaforos binarios

Metodo empezarLeer()  
adquirir(lectores)  
adquirir(mutex1)  
nLectores++  
SI (nLectores==1) ENTONCES  
adquirir(escritores)  
FIN SI  
liberar(mutex1)  
liberar(lectores)

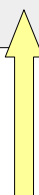
Metodo terminarLeer()  
adquirir(mutex1)  
nLectores--  
SI (nLectores==0) ENTONCES  
liberar(escritores)  
FIN SI  
liberar(mutex1)

analizar



```
Semáforo: mutex1, mutex2, lectores, escritores;  
Entero: nLectores=0; nEscritores=0
```

```
Metodo empezarEscribir  
  adquirir(mutex2)  
  nEscritores++  
  SI (nEscritores==1) ENTONCES  
    adquirir(lectores)  
  FIN SI  
  liberar(mutex2)  
  liberar(escriitores)
```




```
Metodo terminarEscribir  
  liberar(escriitores)  
  adquirir(mutex2)  
  nEscritores--  
  SI (nEscritores==0) ENTONCES  
    liberar(lectores)  
  FIN SI  
  liberar(mutex2)
```



analizar





Semáforo: mutex1, mutex2, lectores, escritores;  
Entero: nLectores=0; nEscritores=0

Metodo empezarEscribir  
adquirir(mutex2)  
nEscritores++  
SI (nEscritores==1) ENTONCES  
    adquirir(lectores)  
FIN SI  
liberar(mutex2)  
adquirir(escriitores)

**CUIDADO**  
adquirir(escriitores)



Metodo terminarEscribir  
liberar(escriitores)  
adquirir(mutex2)  
nEscritores--  
SI (nEscritores==0) ENTONCES  
    liberar(lectores)  
FIN SI  
liberar(mutex2)



analizar



# lectores/escritores: semáforos

- Cómo se inicializan los semáforos para comenzar?
- Permite el acceso simultáneo de lectores impidiendo el uso a escritores ?
- Pruebe el algoritmo con distintas opciones en lectores y escritores
- Es posible la inanición de algún tipo de proceso?