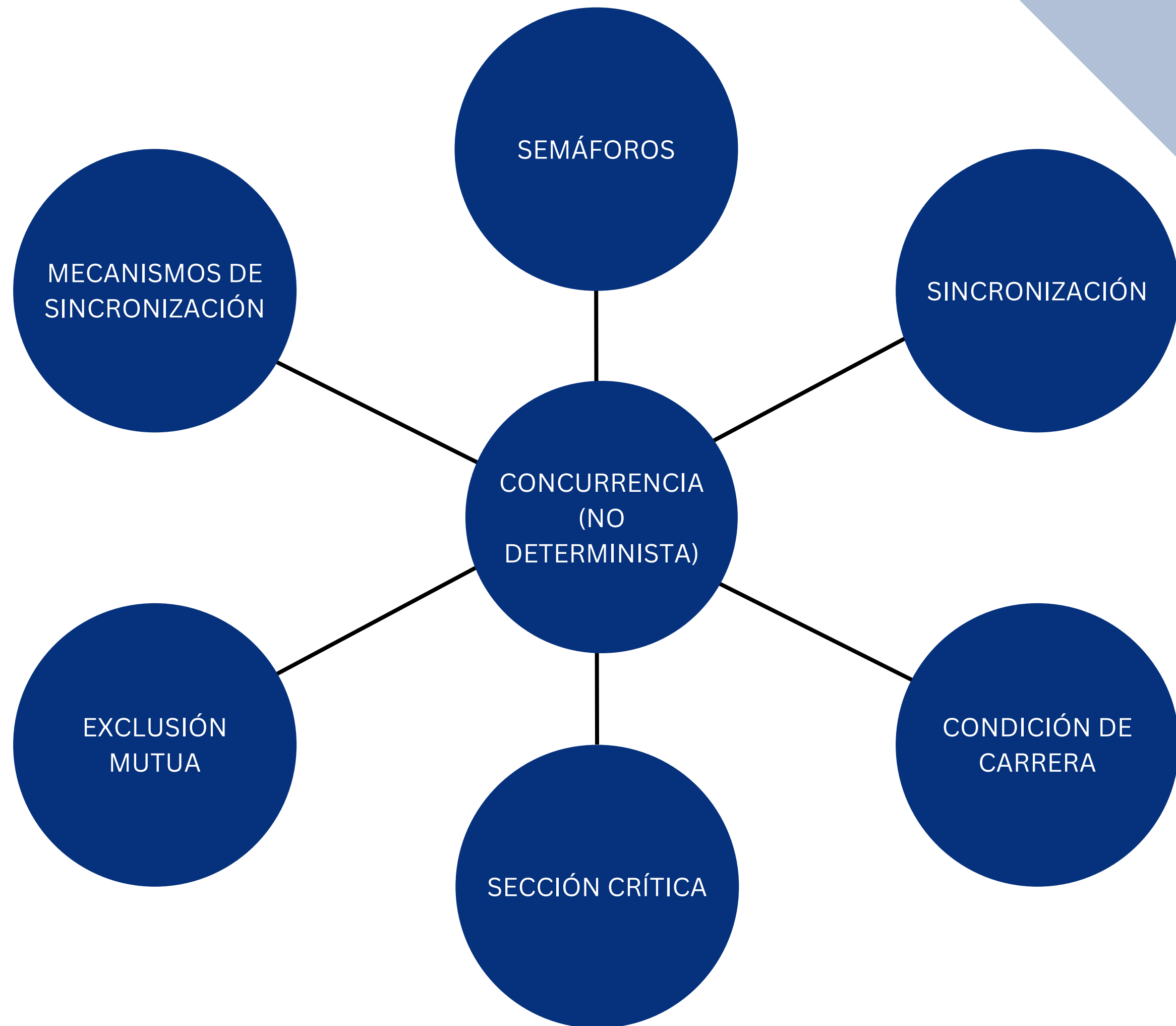


Semáforos

Axel Daniel Malváez Flores
Gabriel Zadquiel Peralta Rionda
Luis Darío Ortiz Izquierdo
Equipo Alpha



¿Qué son los semáforos?

Es un método de sincronización que provee el sistema operativo que no requiere espera ocupada. De tal modo que es una variable que, aparte de la inicialización, sólo puede acceder por medio de 2 operaciones atómicas y mutuamente exclusivas.

Wait(S)

P(S), Down(S)

Significa: Espérate

Signal(S)

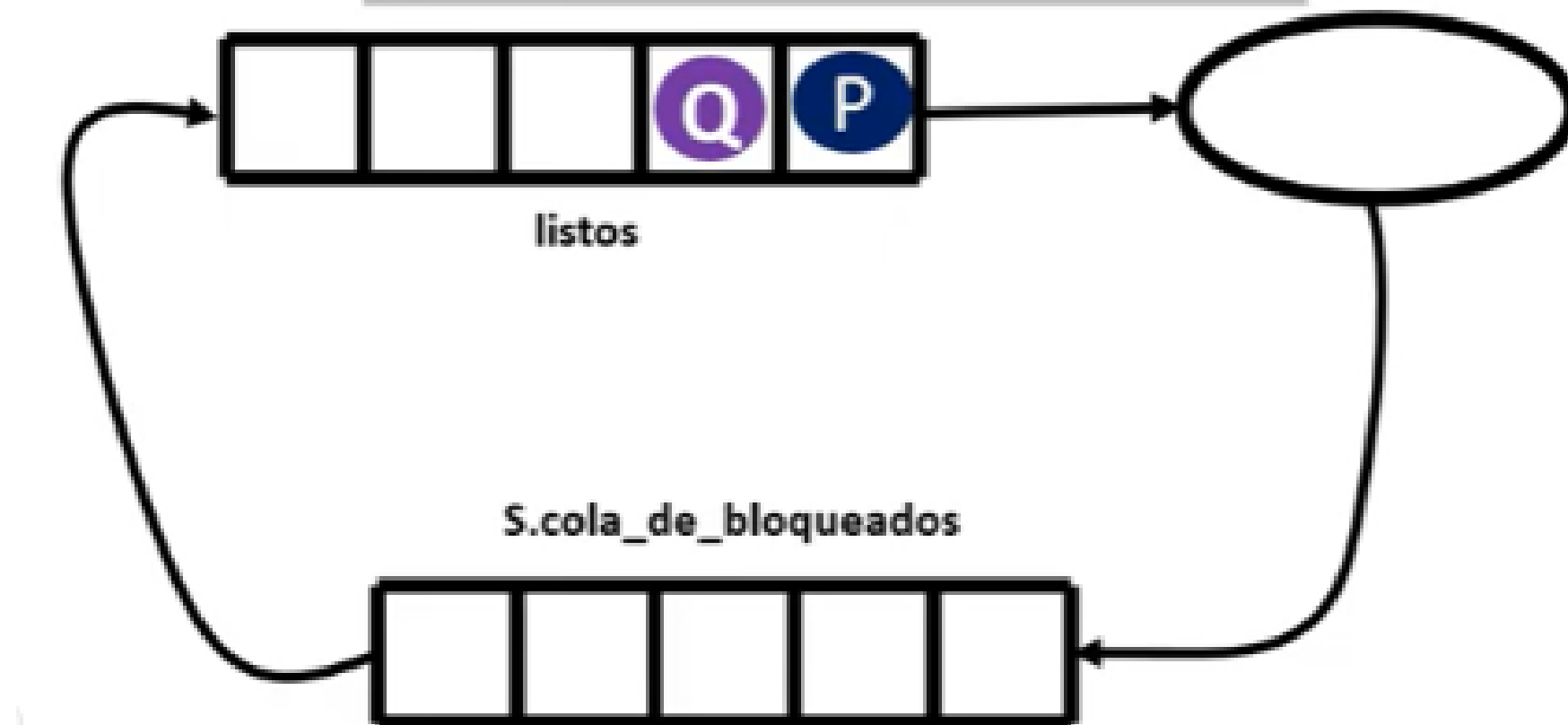
V(S), Up(S), Post(S) o Release(S)

Significa: Vete

Estructura de un semáforo

Semáforos

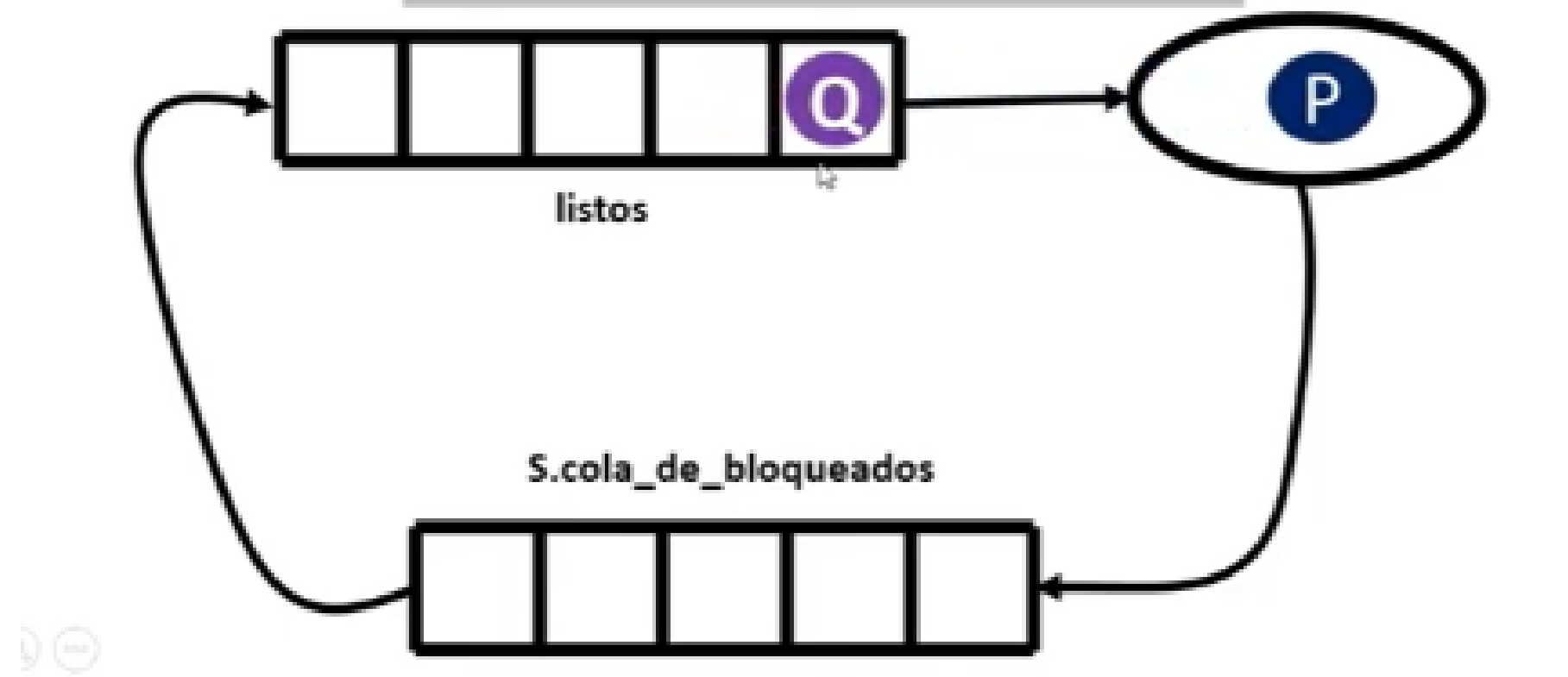
```
struct SEMAPHORE {  
    int count;  
    queue cola_de_bloqueados;  
} S;
```



Estructura de un semáforo

Semáforos

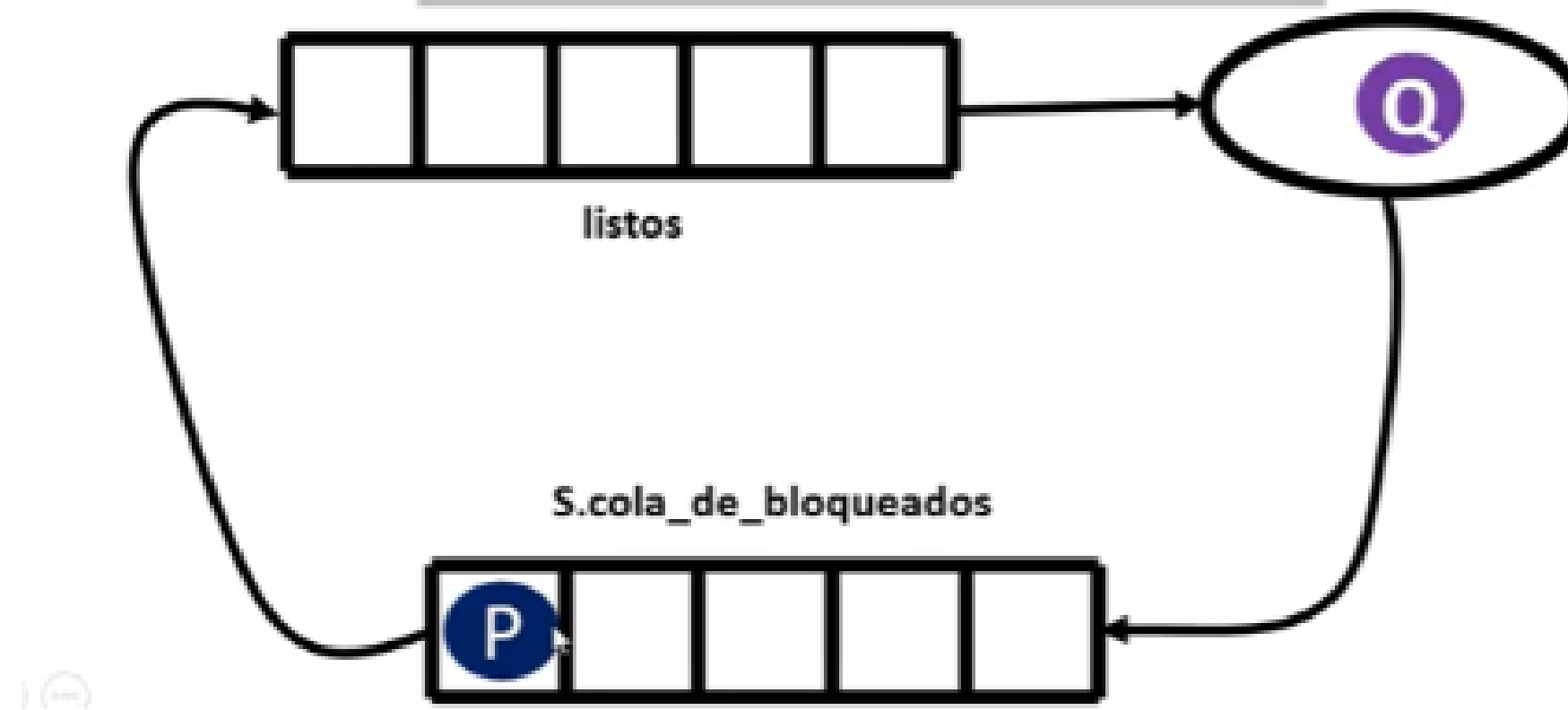
```
struct SEMAPHORE {  
    int count;  
    queue cola_de_bloqueados;  
} S;
```



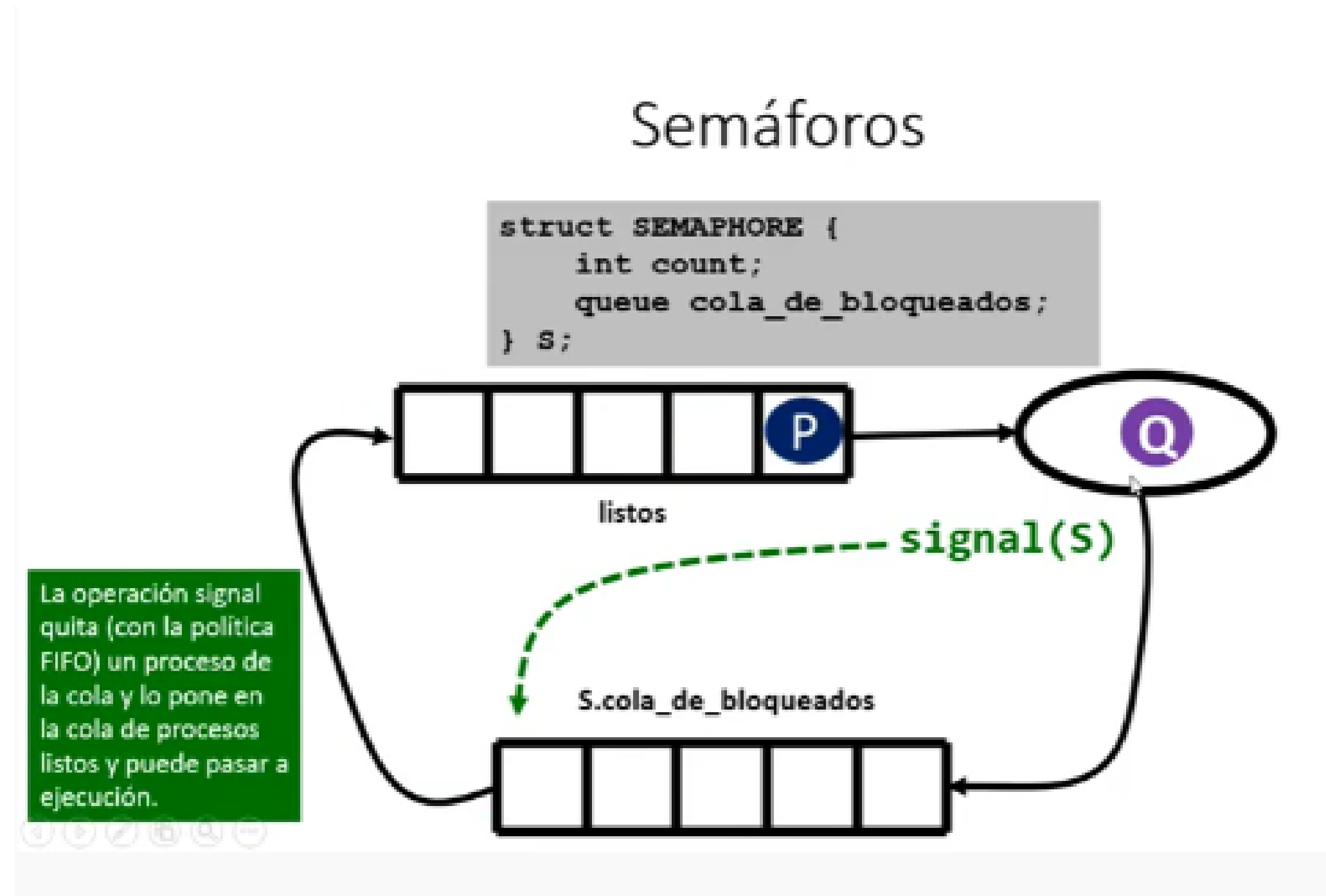
Estructura de un semáforo

Semáforos

```
struct SEMAPHORE {  
    int count;  
    queue cola_de_bloqueados;  
} S;
```



Estructura de un semáforo



Tipos de semáforos

Semáforos binarios

- Solo puede tener dos valores, 0 y 1
- En windows se llaman mutex

Semáforos generales o enteros

- Pueden tomar muchos valores positivos o incluso valores negativos

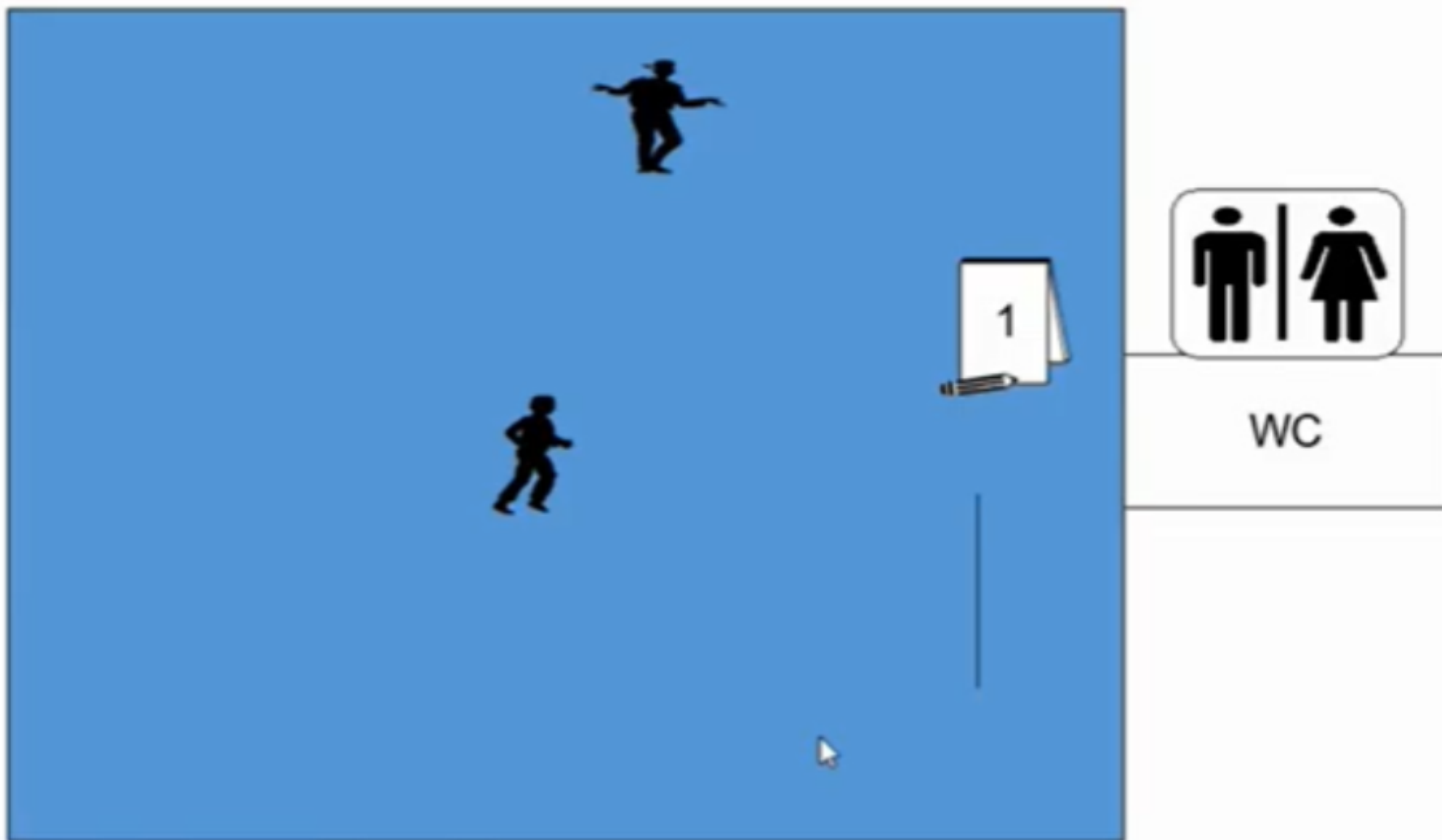
Semáforos binarios

```
struct SEMAPHORE {  
    int valor; (0,1)  
    queue cola_de_bloqueados;  
} s;
```

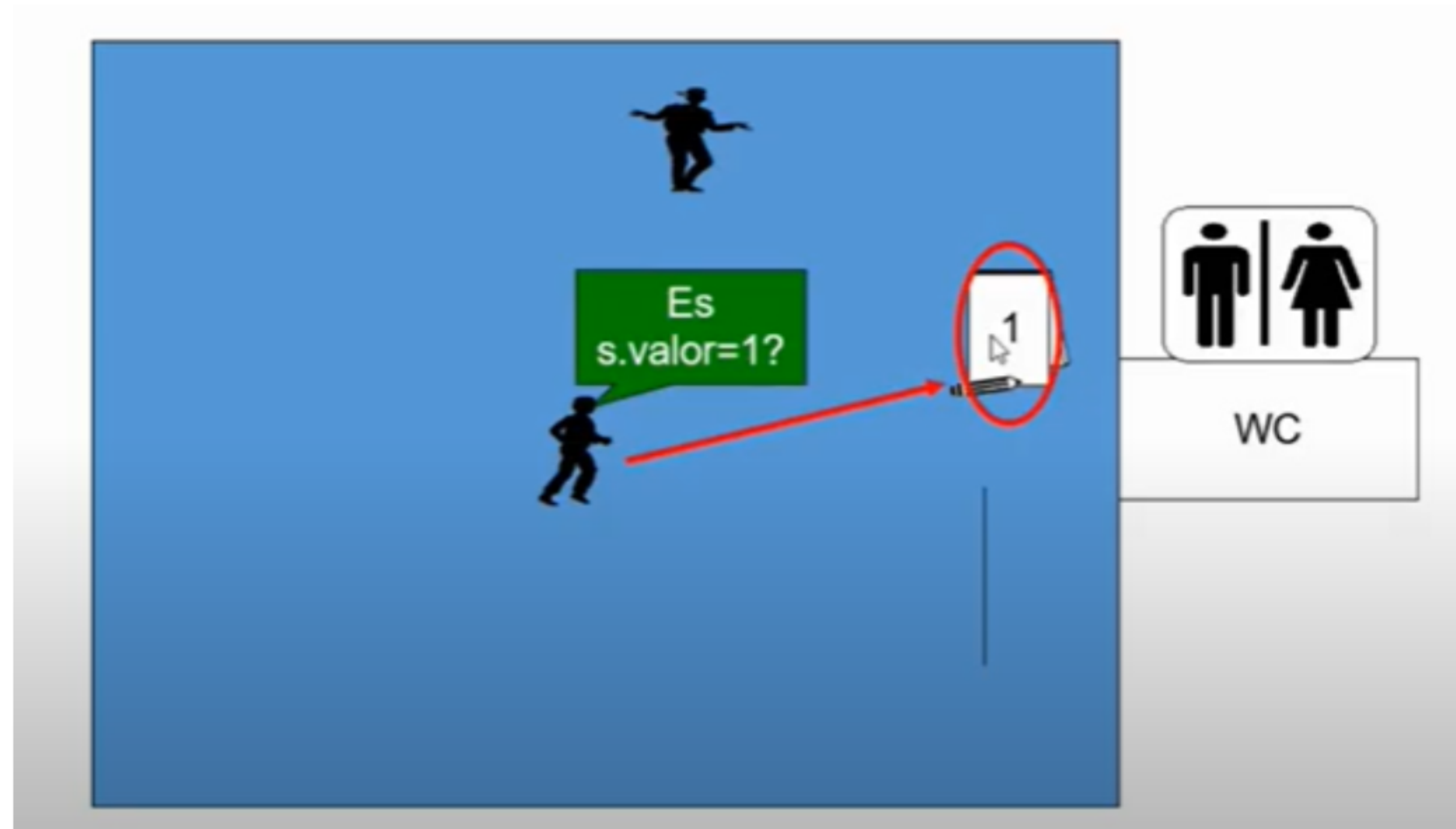
```
WaitB(s):  
    if s.valor=1  
        s.valor=0  
    else {  
        poner este proceso en s.cola_de_bloqueados;  
        bloquear este proceso  
    };
```

```
SignalB(s):  
    If s.cola_de_bloqueados está vacía  
        s.valor=1  
    else {  
        quitar un proceso P de s.cola_de_bloqueados;  
        poner el proceso P en la cola de listos  
    };
```

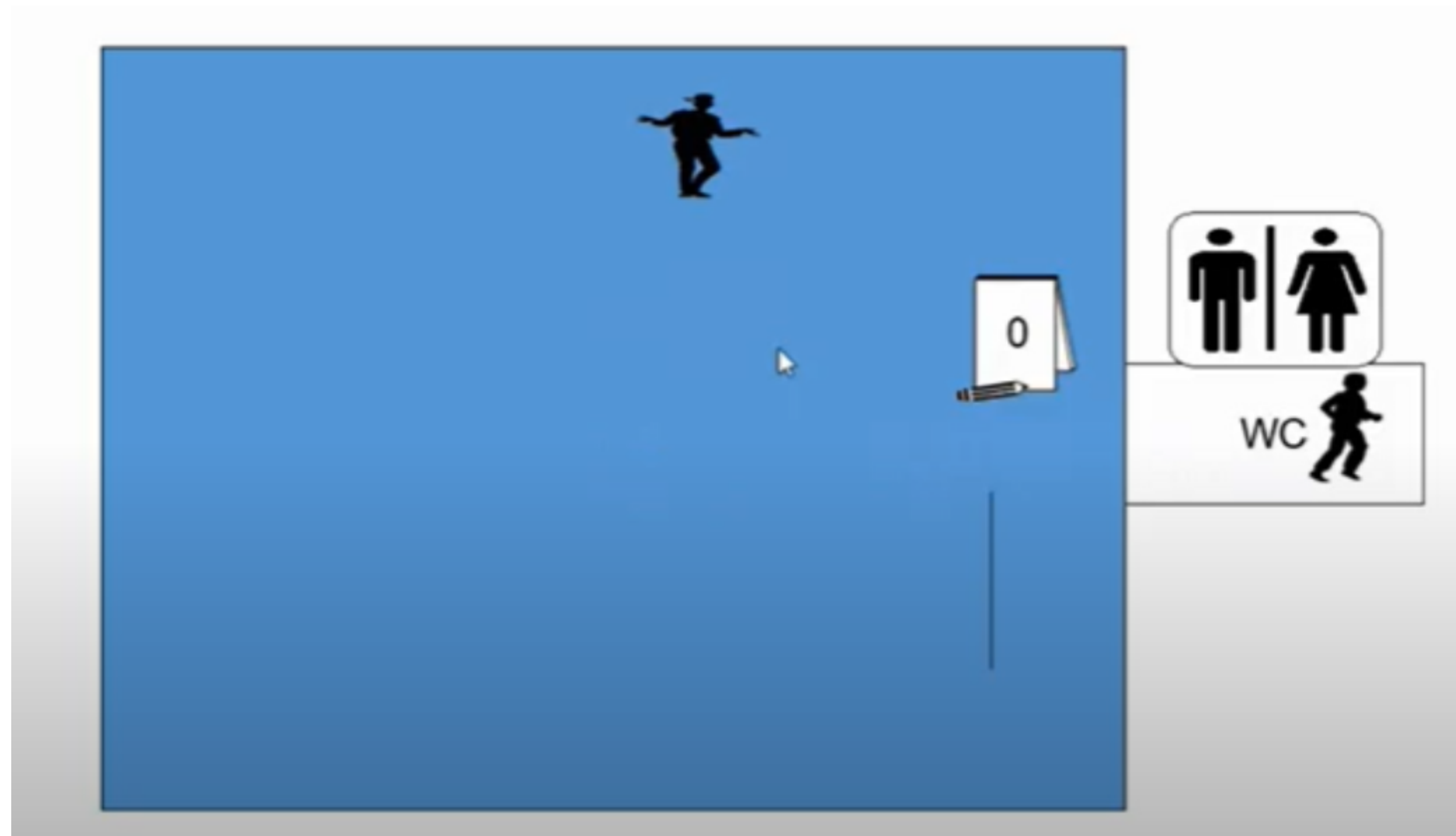
**El problema de
los borrachos
de la cantina
con semáforos
binarios.**



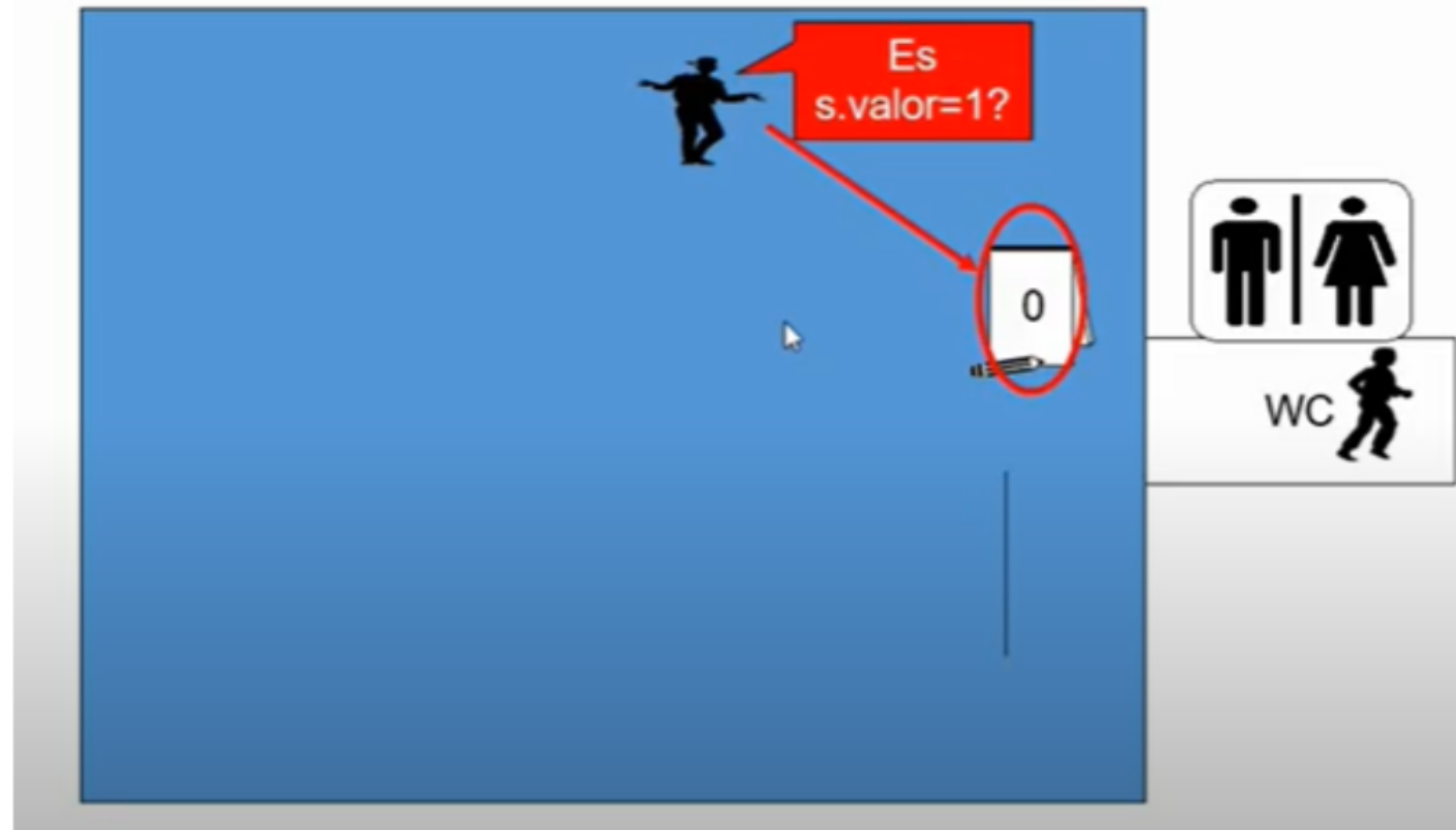
**El problema de
los borrachos
de la cantina
con semáforos
binarios.**



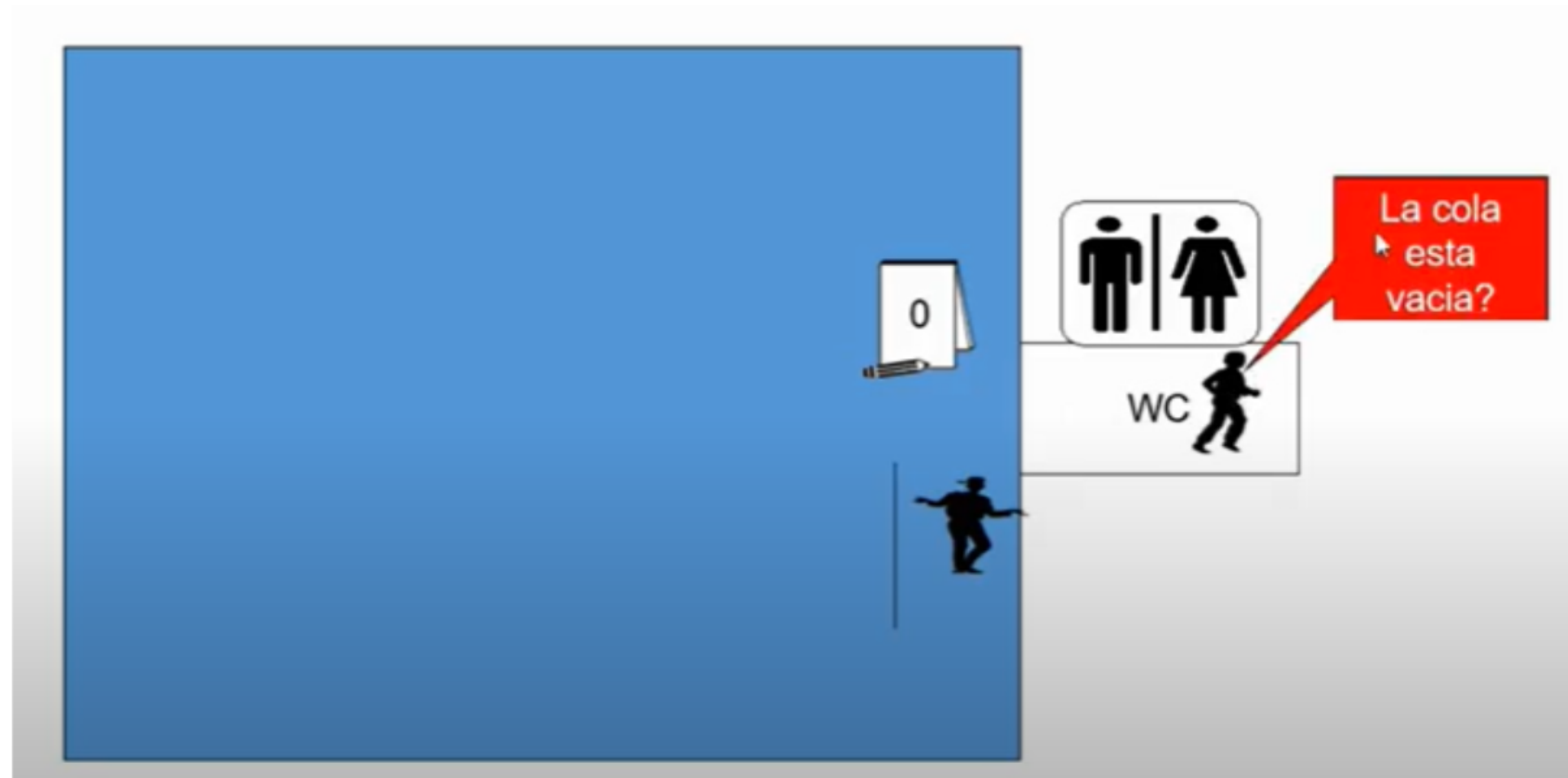
**El problema de
los borrachos
de la cantina
con semáforos
binarios.**



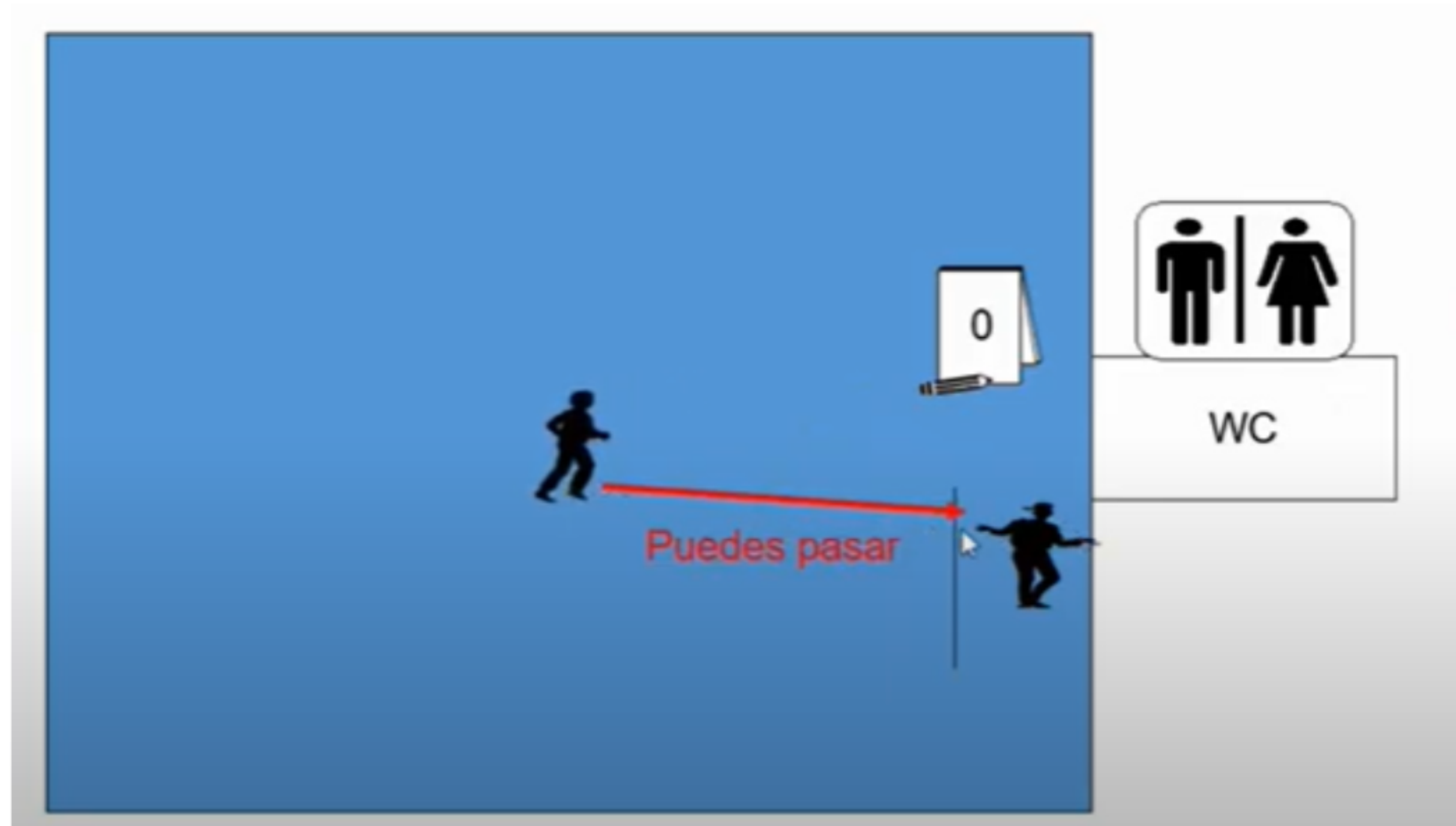
**El problema de
los borrachos
de la cantina
con semáforos
binarios.**



**El problema de
los borrachos
de la cantina
con semáforos
binarios.**



**El problema de
los borrachos
de la cantina
con semáforos
binarios.**



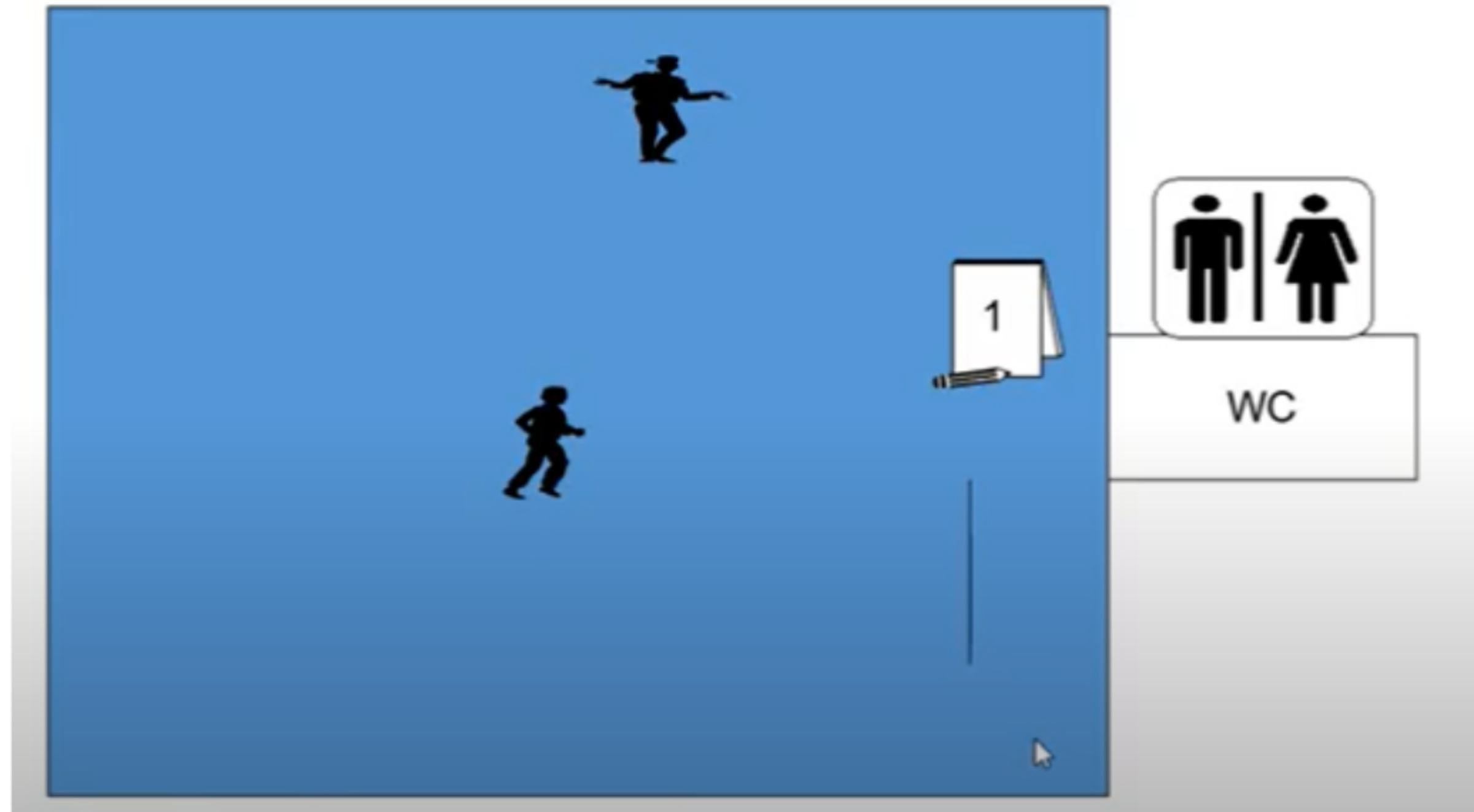
Semáforos generales (con negativos)

```
struct SEMAPHORE {  
    int contador;  
    queue cola_de_bloqueados;  
} s;
```

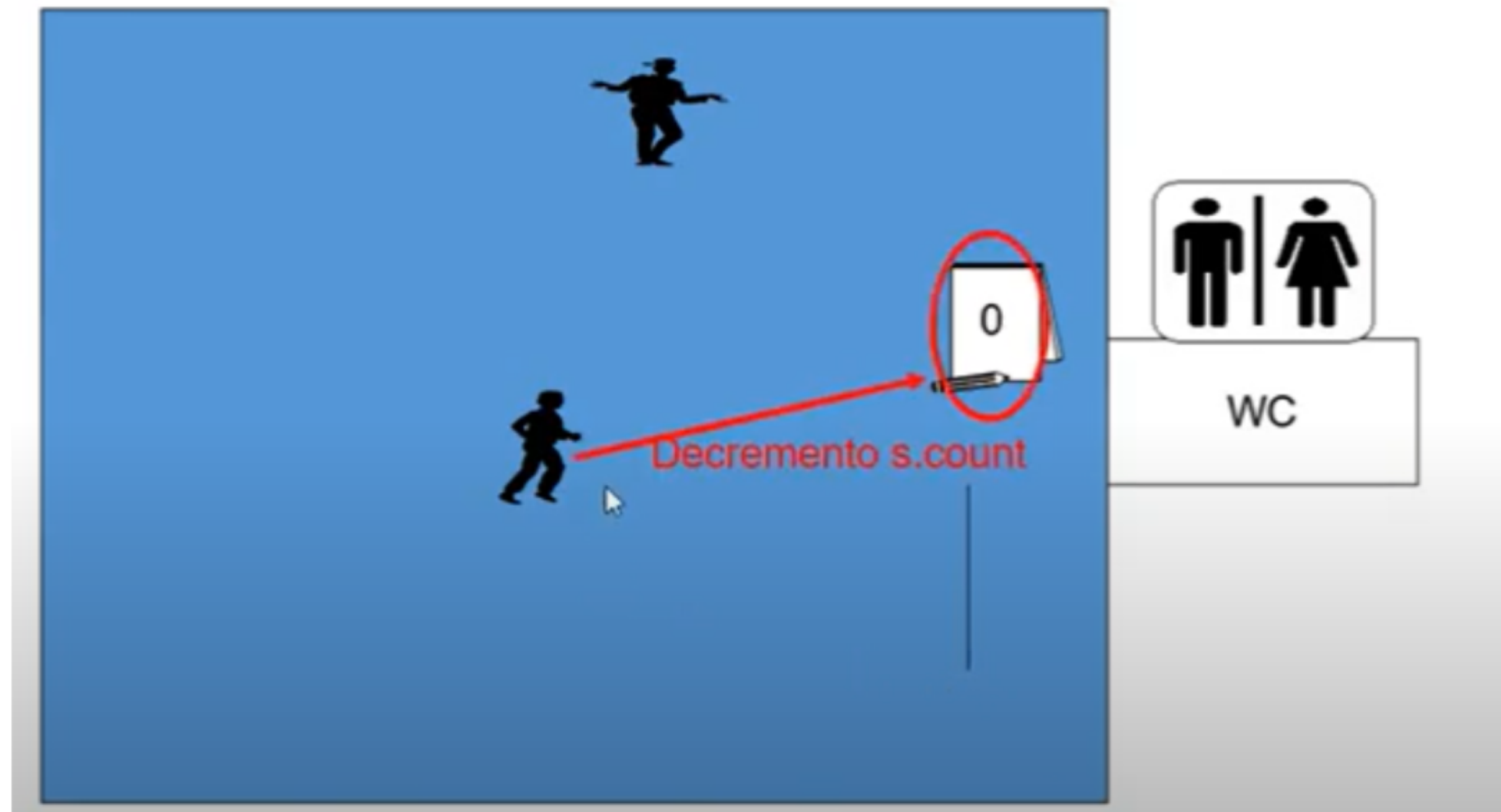
```
Wait(s):  
    s.contador--;  
    if s.contador < 0 then  
    {  
        poner este proceso en s.cola_de_bloqueados;  
        bloquear este proceso  
    }
```

```
Signal(s):  
    s.contador++;  
    if s.contador <= 0  
    {  
        quitar un proceso P de s.cola_de_bloqueados;  
        poner el proceso P en la cola de listos  
    }
```

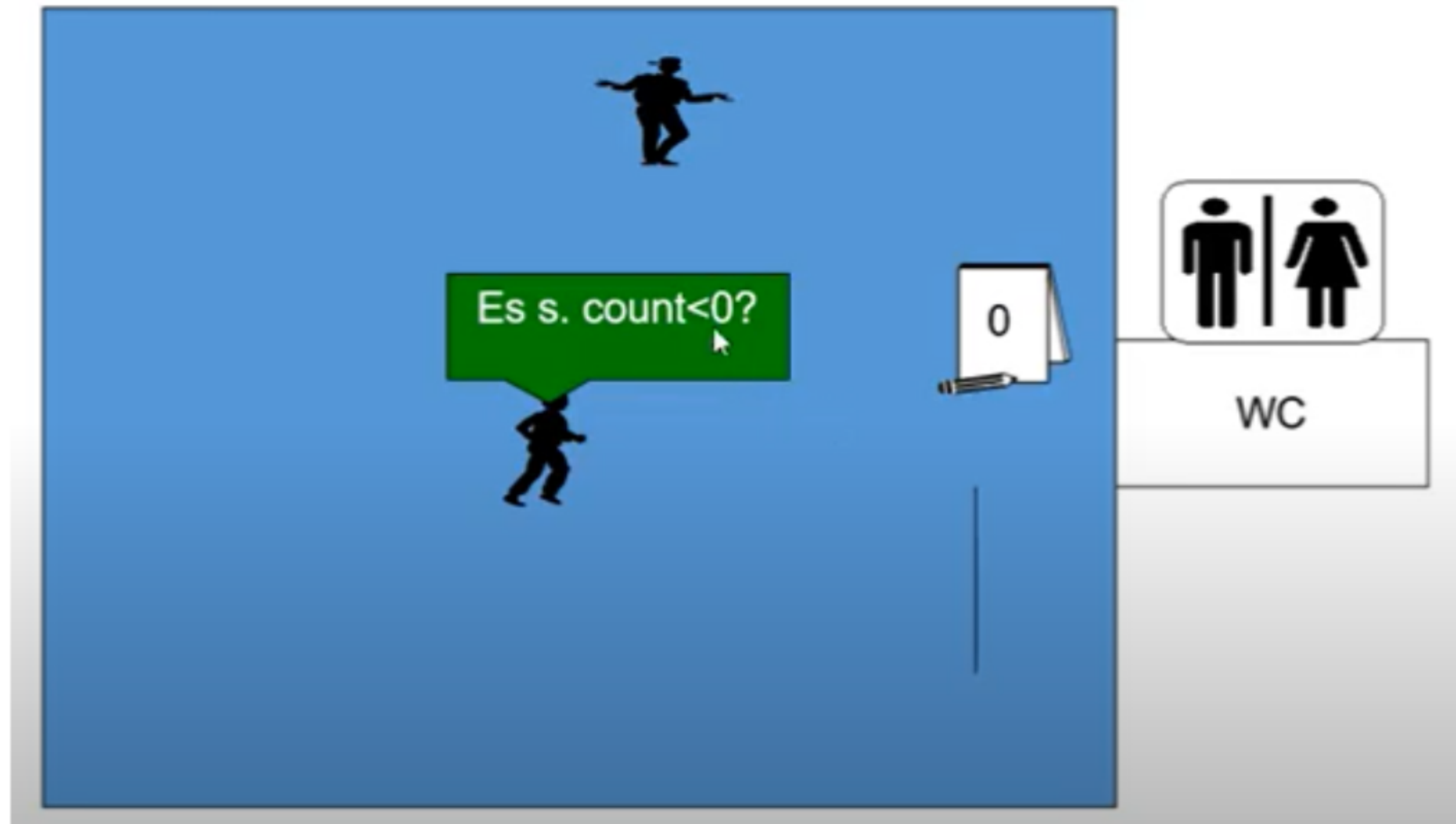

**El problema de
los borrachos
de la cantina
con semáforos
enteros
(Utilizando
valores -)**



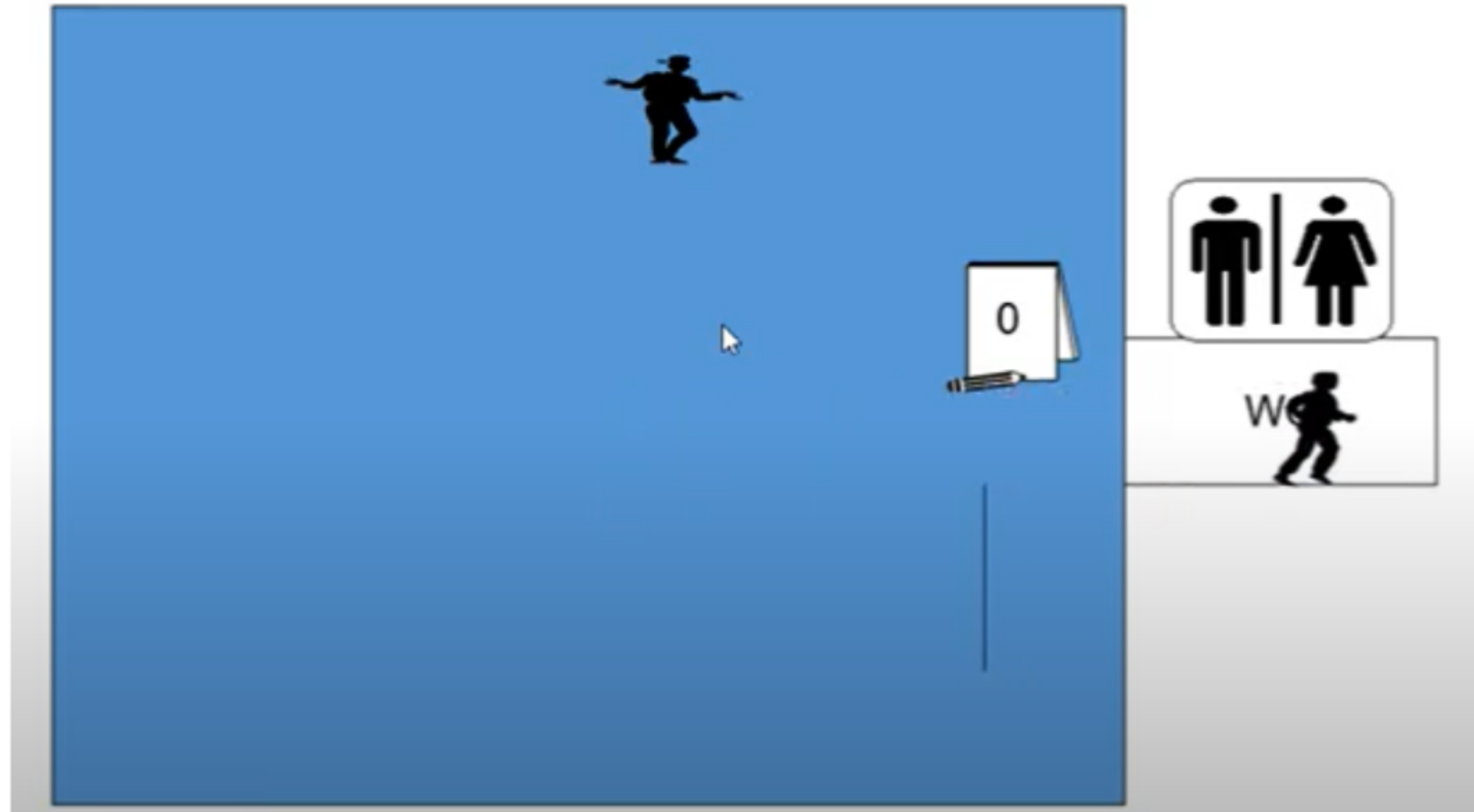
**El problema de
los borrachos
de la cantina
con semáforos
enteros
(Utilizando
valores -)**



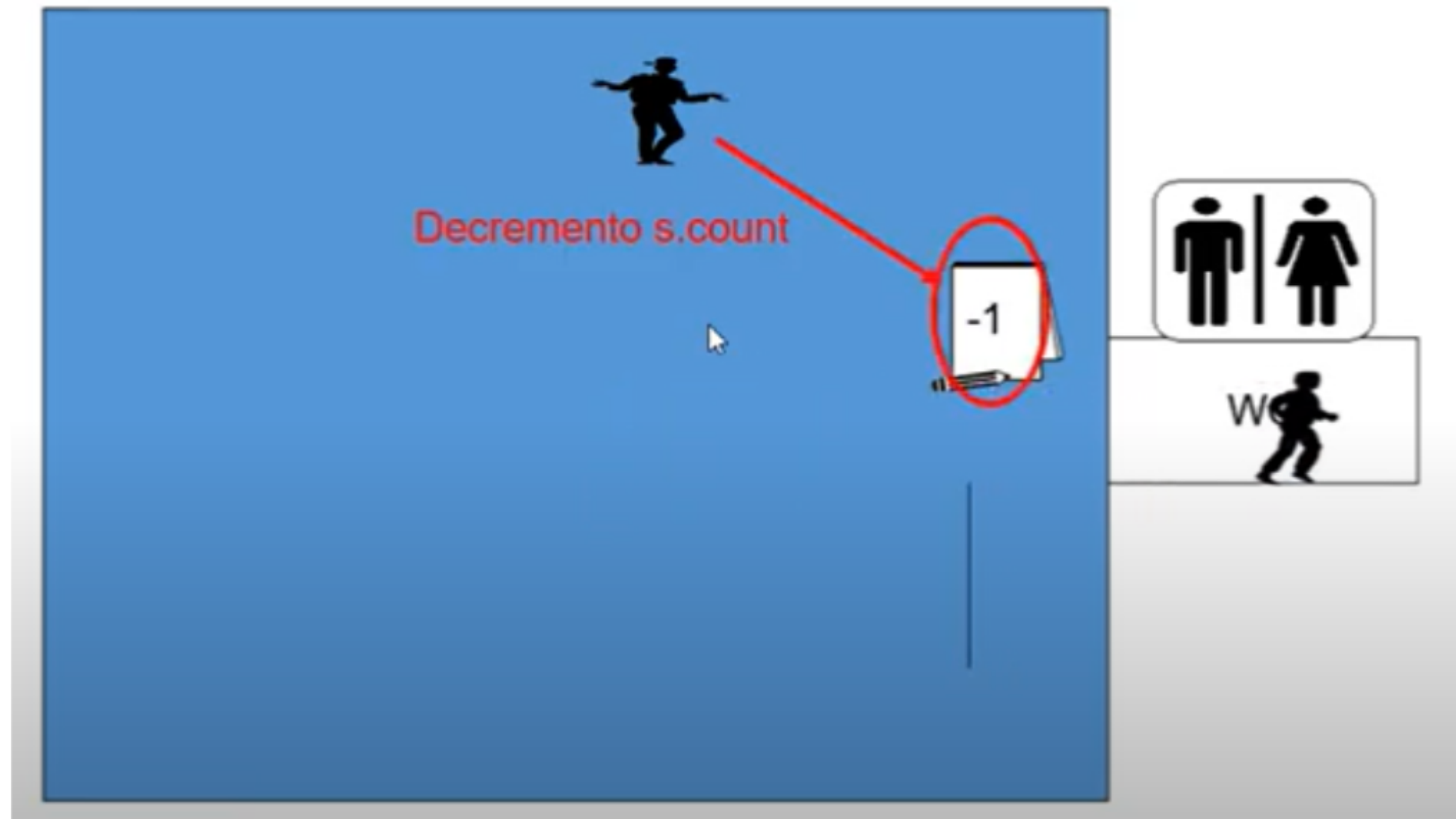
**El problema de
los borrachos
de la cantina
con semáforos
enteros
(Utilizando
valores -)**



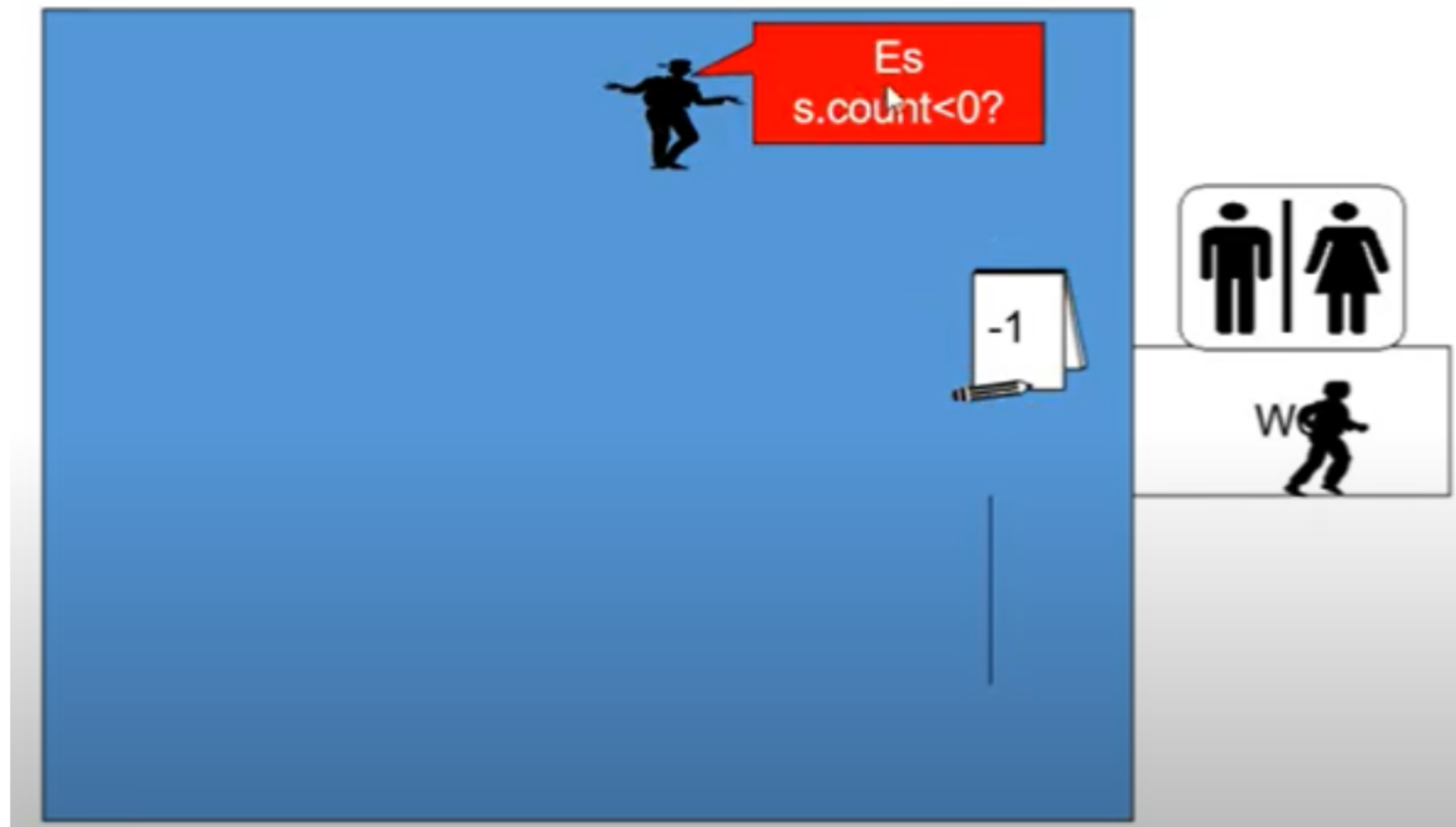
**El problema de
los borrachos
de la cantina
con semáforos
enteros
(Utilizando
valores -)**



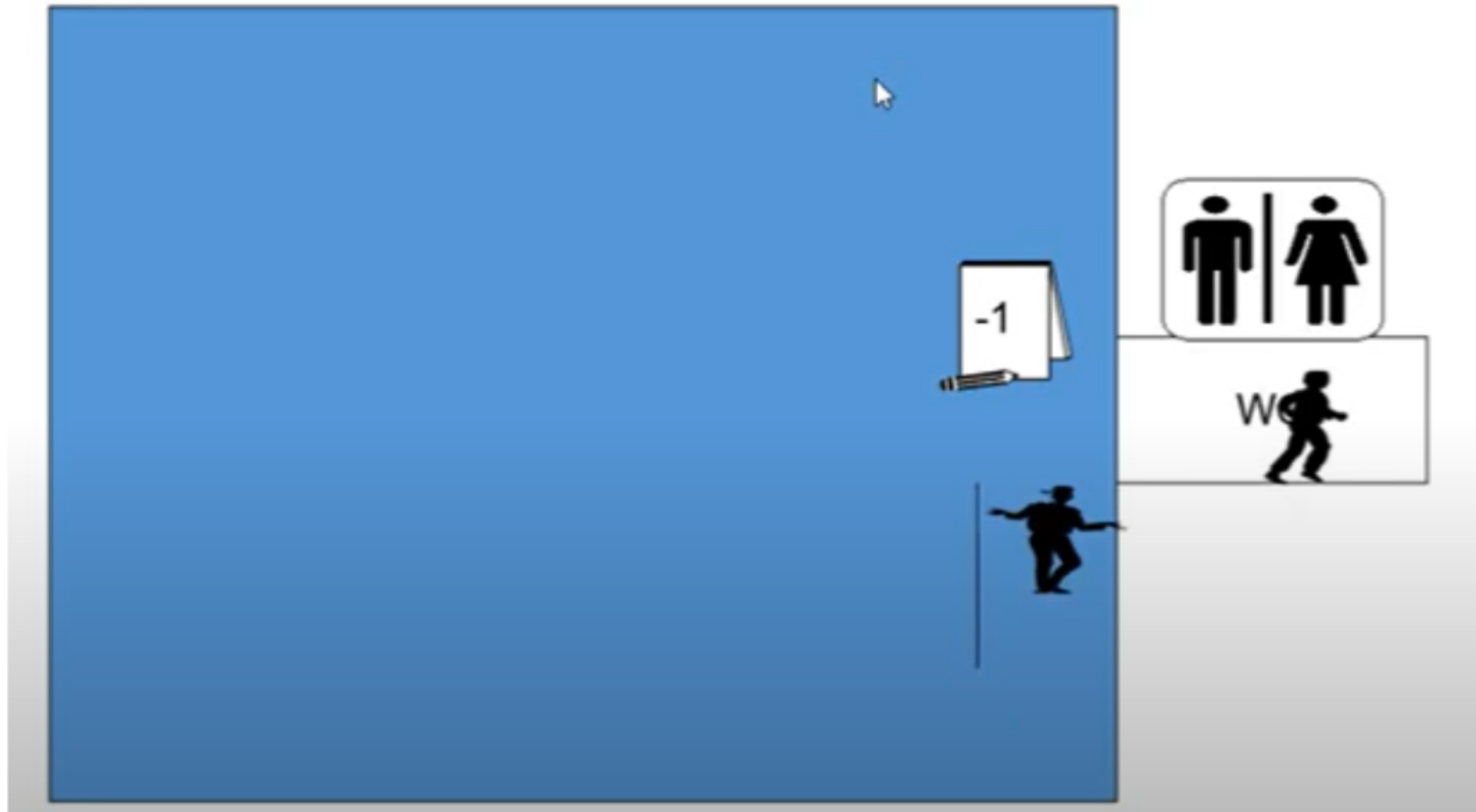
**El problema de
los borrachos
de la cantina
con semáforos
enteros
(Utilizando
valores -)**



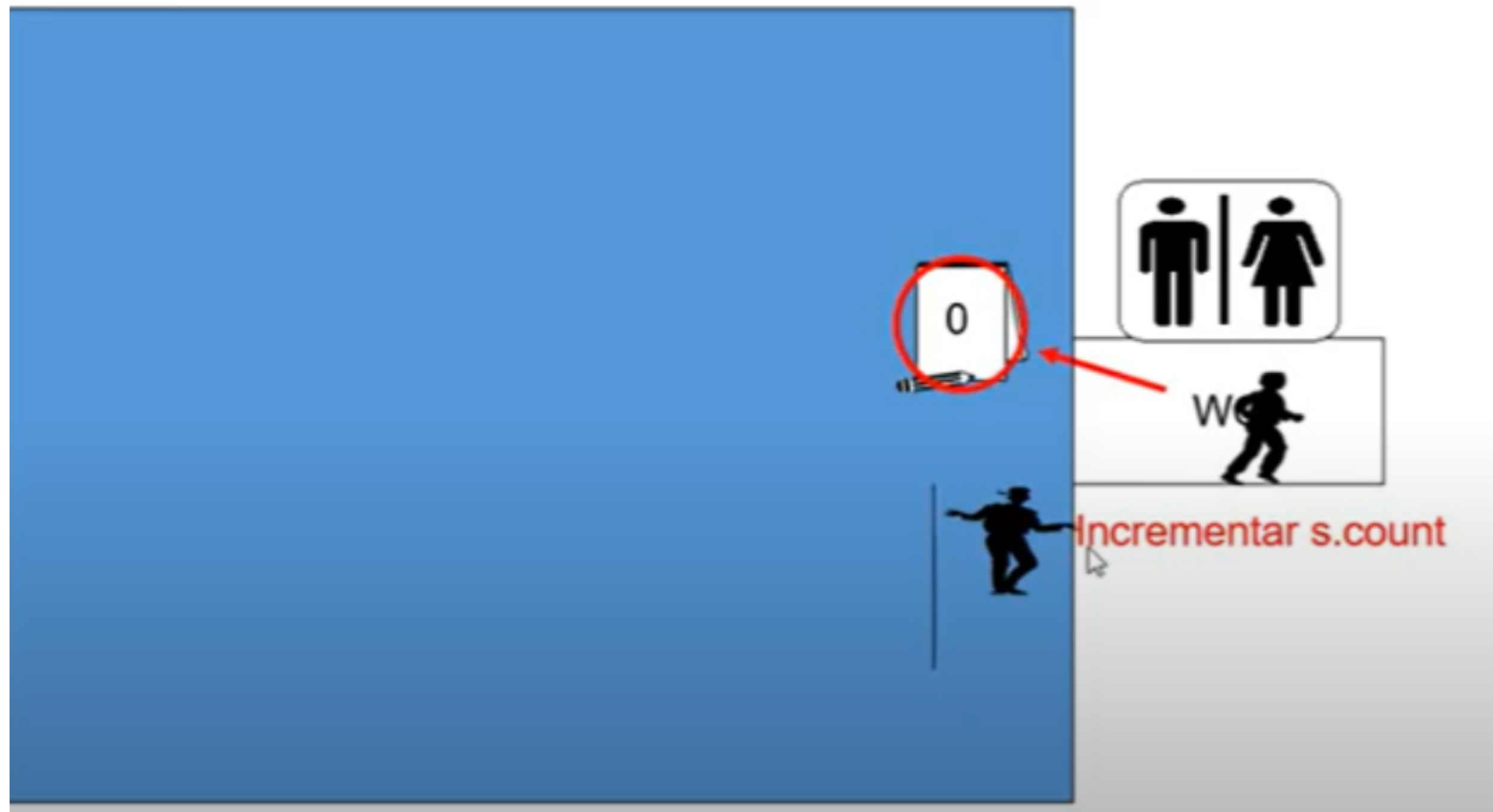
**El problema de
los borrachos
de la cantina
con semáforos
enteros
(Utilizando
valores -)**



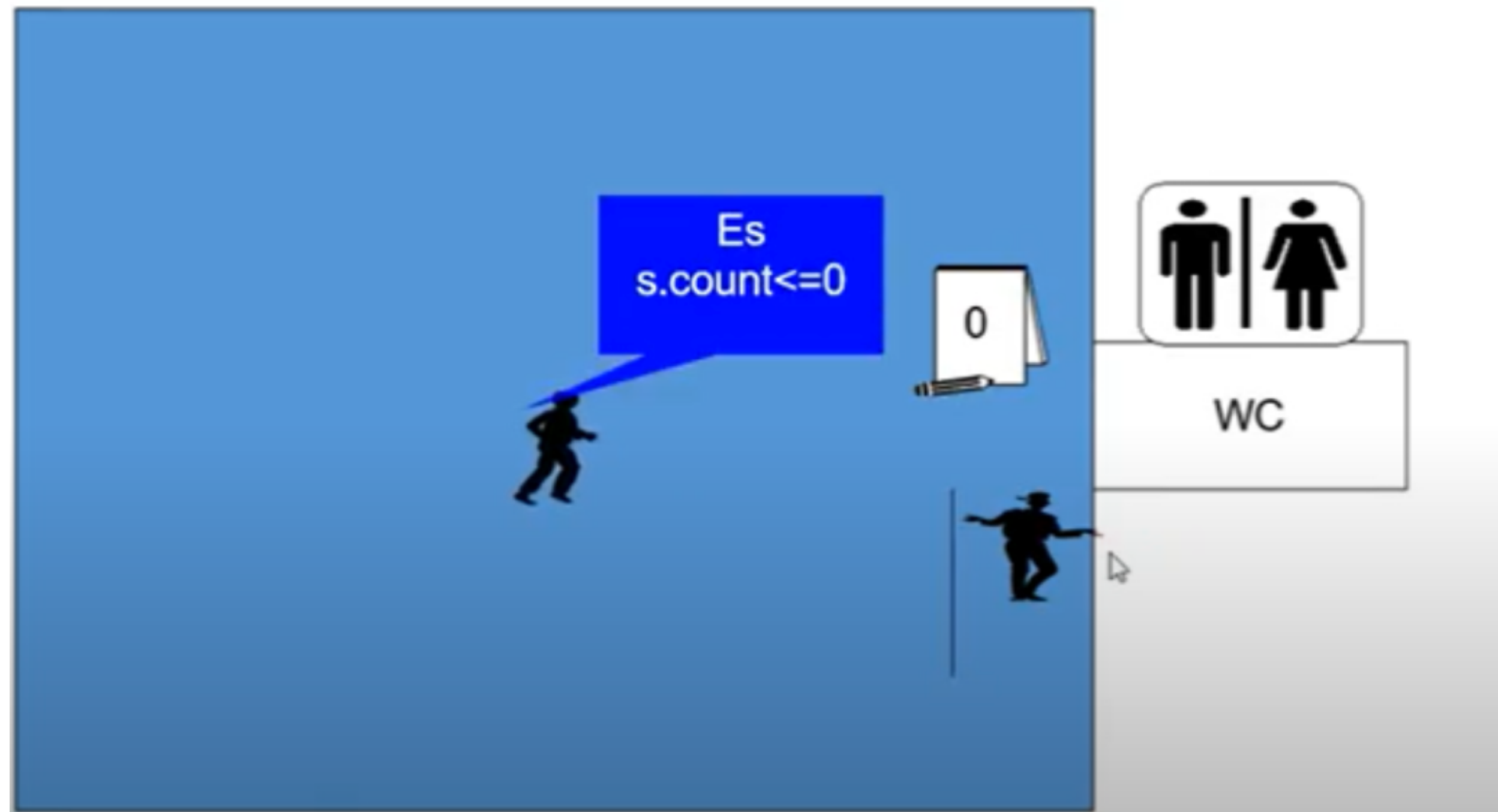
**El problema de
los borrachos
de la cantina
con semáforos
enteros
(Utilizando
valores -)**



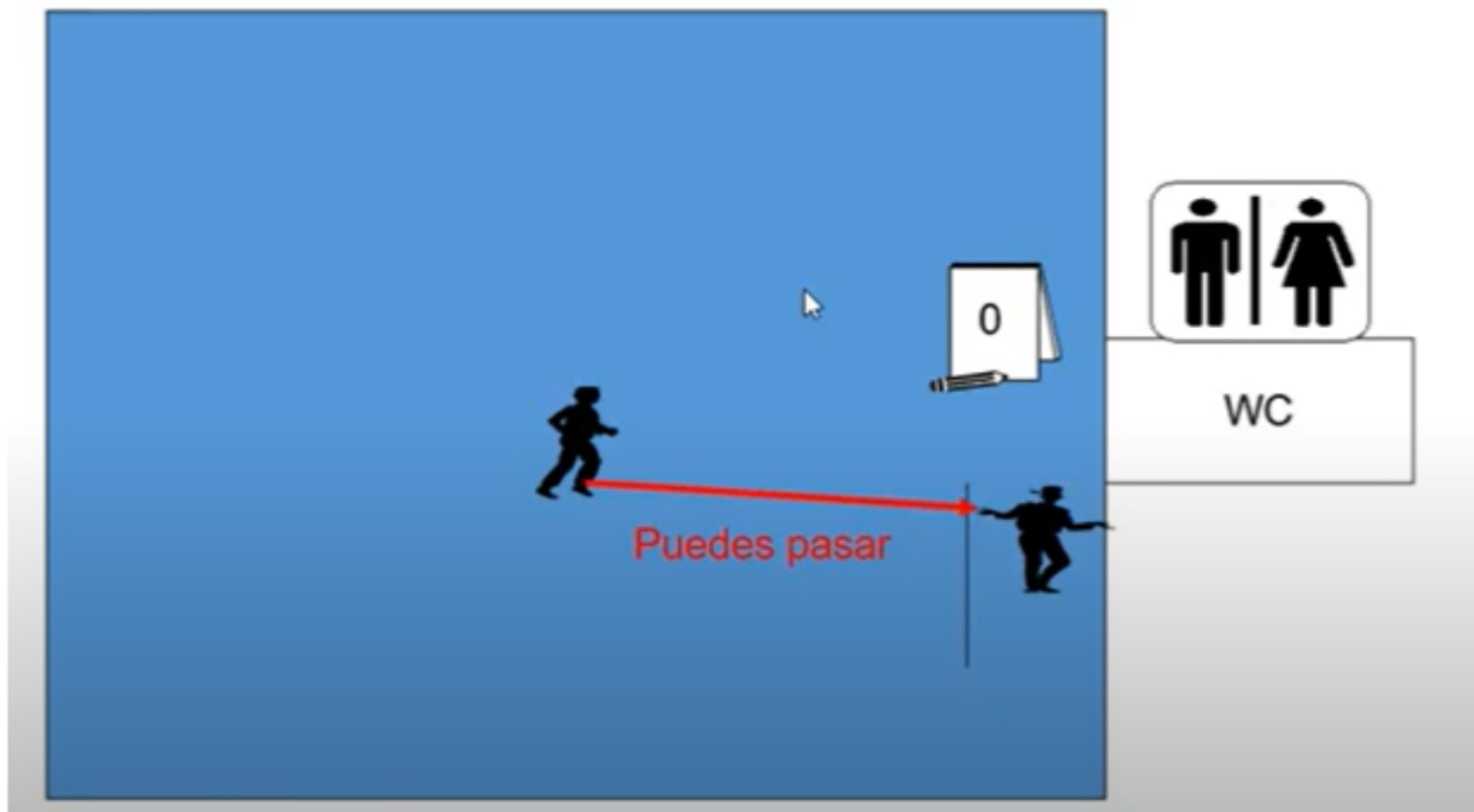
**El problema de
los borrachos
de la cantina
con semáforos
enteros
(Utilizando
valores -)**



**El problema de
los borrachos
de la cantina
con semáforos
enteros
(Utilizando
valores -)**




**El problema de
los borrachos
de la cantina
con semáforos
enteros
(Utilizando
valores -)**



Semáforos generales (con enteros)

```
struct SEMAPHORE {  
    unsigned int contador;  
    unsigned int bloqueados;  
    queue cola_de_bloqueados;  
} s;
```



```
Wait(s):  
    if s.contador==0 then  
    {  
        s.bloqueados++;  
        poner este proceso en s.cola_de_bloqueados;  
        bloquear este proceso;  
    }  
    else  
        s.contador--;
```

```
Signal(s):  
    if s.bloqueados==0 then  
        s.contador++;  
    else  
    {  
        quitar un proceso P de s.cola_de_bloqueados;  
        poner el proceso P en la cola de listos  
        s.bloqueados--;  
    }
```

Pseudocódigo

```
main()
{
    cobegin {
        P(0);P(1);P(2)... P(N);
    }
}
```

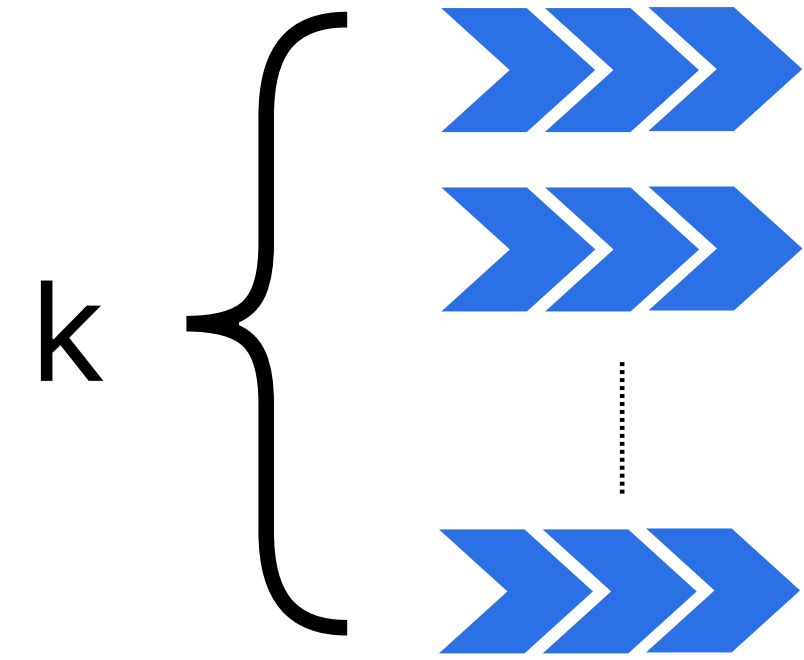


```
Semaphore S;

Process P(int i)
{
    while(1)
    {
        wait(S) ;
        CS
        signal(S) ;
        RS
    }
}
```

Ejemplo

Ejecutar K procesos en una sección crítica



Vamos a suponer que yo tengo un ancho de banda limitado a 100 MB, y yo tengo procesos que para que se ejecuten bien necesitan 50 MB y yo tengo K procesos en ejecución, ¿Qué necesito hacer para que todos los procesos que se van a ejecutar pueda asegurar que tendrán sus 50 MB?

Lo que necesito hacer es limitar la cantidad de procesos, en este caso necesitaría limitar a que dos procesos están ocupando la sección crítica.

Nuestra solución es usar un semáforo entero que inicializamos con el valor de 2, con esto permitimos que dos procesos compartan la sección crítica al mismo tiempo

Utilizando semáforos

Limitando a k procesos

P(0)	P(1)	P(2)	S.count
			2
wait(S);			1
	wait(S);		0
		wait(S);	-1
CS			
	CS		
signal(S);			0
		CS	
	signal(S);		1
		signal(S);	2

Ejemplo

Cocineros y meseros

Tenemos los procesos

- cocinero()
- mesero()

que iniciarán concurrentemente.

Posteriormente el cocinero preparará la comida y finalmente el mesero servirá la comida.

```
Process cocinero()  
{  
    preparar_comida();  
}
```

```
Process mesero()  
{  
    servir_comida();  
}
```

Cocineros y meseros

Los procesos iniciarán concurrentemente, es decir al mismo tiempo arrancan los dos. Lo que ocurre es que el mesero puede ir a servir la comida mientras el cocinero cocina la comida, sin embargo hay un detalle; el mesero no debe repartir la comida hasta que no esté preparada. Entonces necesitamos una restricción:

`Preparar_comida()` en `cocinero()` sea ejecutado antes que `servir_comida()` en `mesero()`.

Cocineros y meseros

```
semaphore synch;
```

```
main()
{
    initsem(synch,0);
    cobegin {
        cocinero();
        mesero();
    }
}
```



```
Process cocinero()
{
    preparar_comida();
    signal(synch);
}
```



```
Process mesero()
{
    wait(synch);
    servir_comida();
}
```

Pasos

Cocineros y meseros

`cocinero()`

`mesero()`

`synch.count=0`

`wait(synch)`

Pasos


Cocineros y meseros

`cocinero()`

`mesero()`

`synch.count=0`

`wait(synch)`



Pasos

Cocineros y meseros

cocinero()

`synch.count=-1`

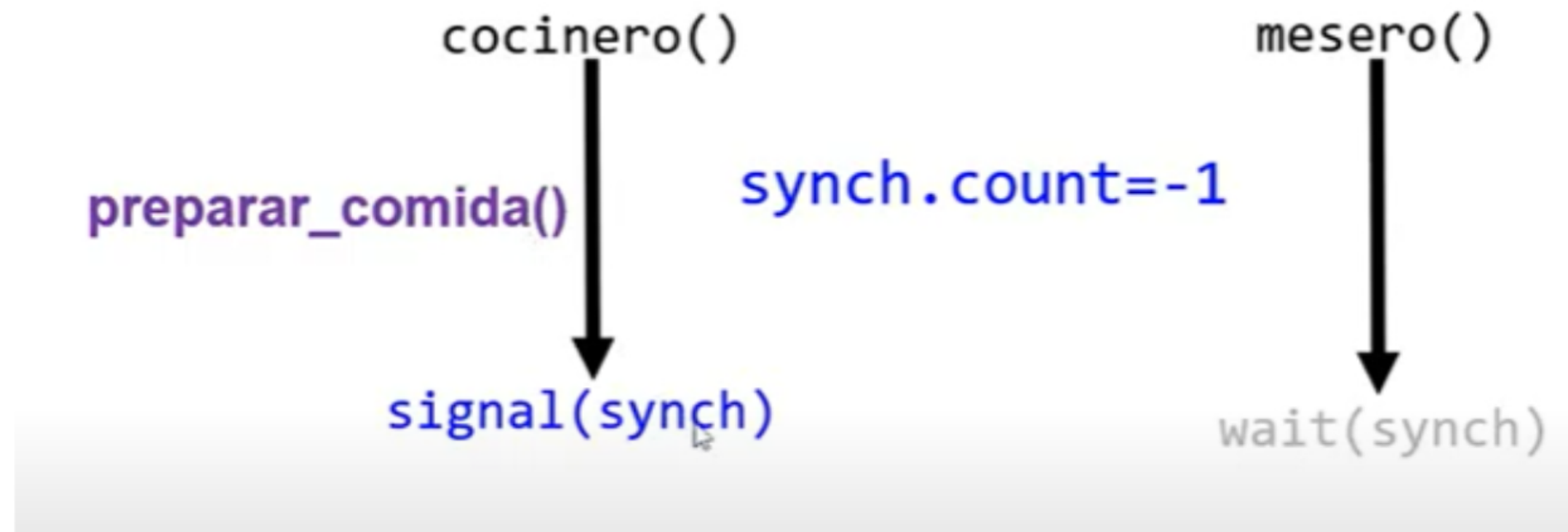
mesero()

`wait(synch)`

A thick black vertical arrow points from the text 'mesero()' down to the text 'wait(synch)'. The arrow starts at the 'o' in 'mesero()' and ends at the 't' in 'wait(synch)'. The background of the slide is white, with a blue header bar at the top left and a light gray bar at the bottom.

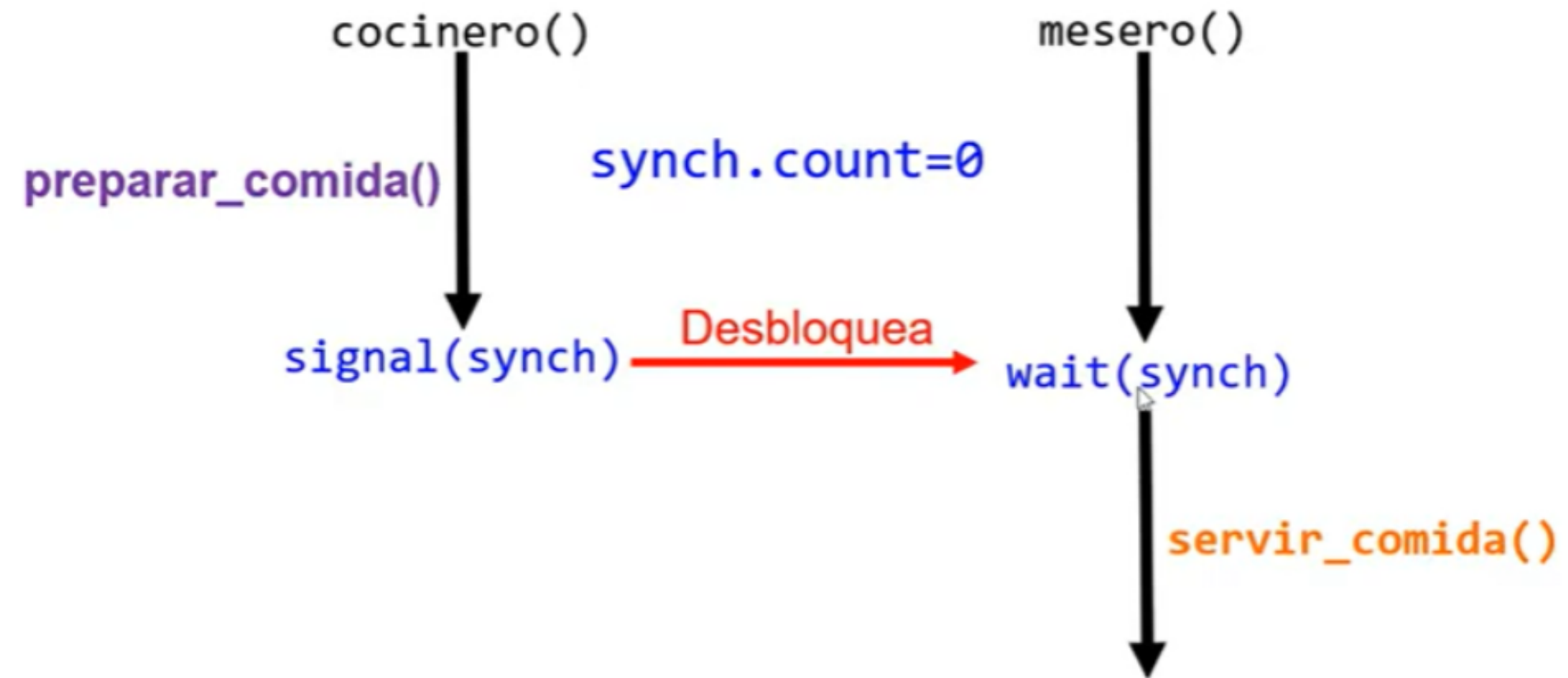
Pasos

Cocineros y meseros



Pasos

Cocineros y meseros



Glosario

CONCURRENCIA

Eventos cuyo orden de ejecución es no determinista

SINCRONIZACIÓN

Es un método de administración de los procesos que garantiza la exclusión mutua en la concurrencia.

CONDICIÓN DE CARRERA

Categoría de errores de programación que involucra a dos procesos que fallan al comunicarse su estado mutua, llevando a resultados inconsistentes

OPERACIÓN ATÓMICA

Manipulación de datos que requiere la garantía de que se ejecutará como una sola unidad de ejecución, o fallará completamente, sin resultados o estados parciales

Glosario

Sección crítica

El área de código que requiere ser protegida de accesos simultáneos donde se realiza la modificación de datos compartidos

Recurso compartido

Es un recurso al cual se puede tener accesos desde más de un proceso. En muchos casos esto es una variable en memoria pero podría ser archivos, periféricos, etc.

Espera ocupada

Cuando un proceso repetidamente verifica una condición, tal como esperar una entrada de teclado o si el ingreso a una sección crítica está habilitado.