Σειρα εργασιων 3

ΣΤΑΜΟΥΛΟΣ ΑΛΕΞΑΝΔΡΟΣ

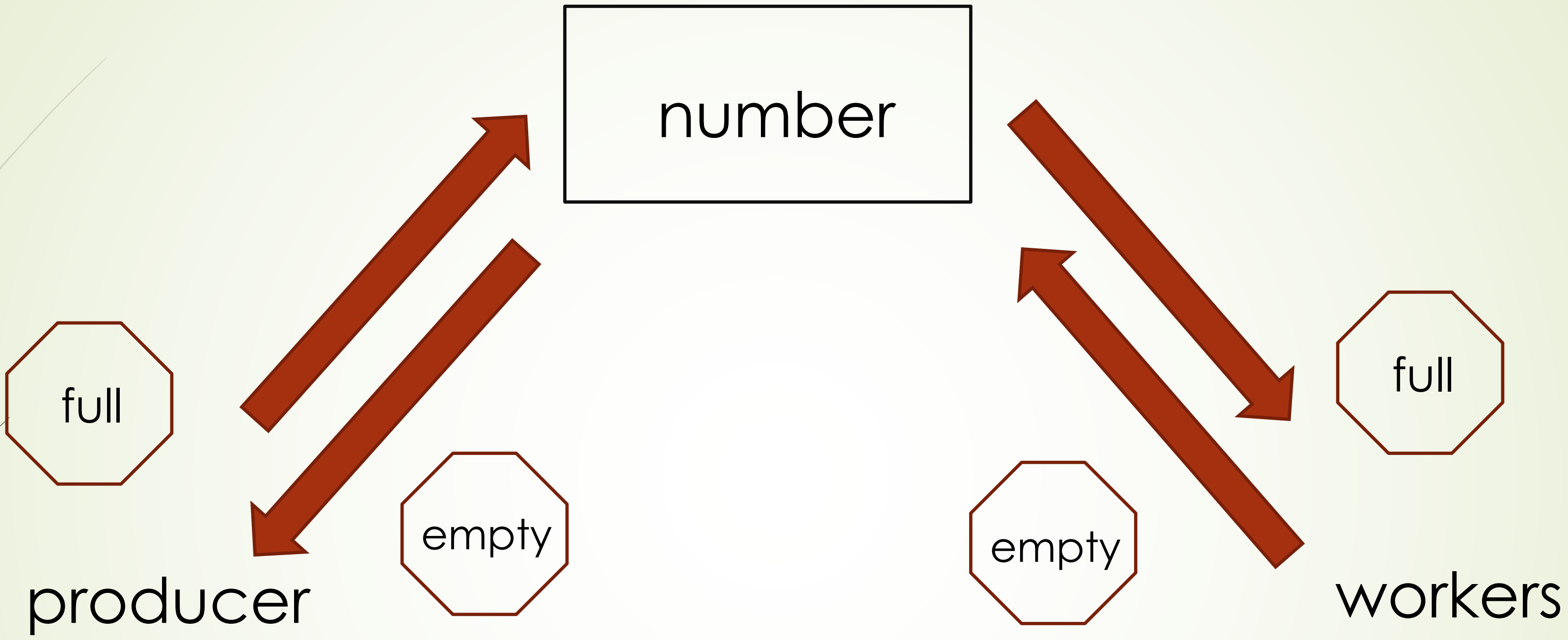
ΚΥΡΙΤΣΗΣ ΒΑΣΙΛΕΙΟΣ

Ασκηση 1

```
Count = 0; init(mutex) ,init(full), init(count)
```

```
While(1){
    produce()
    mutex lock
    while(count = 1)
        cond wait(full)
    put number
    count ++
    if(count == 1)
        cond wait(empty)
    mutex unlock
}
```

```
While(1){
    mutex lock
    while(count = 0)
        cond wait(empty)
    job = number
    count --
    if(count == 0)
        cond wait(full)
    mutex unlock
    process value
}
```


number
full
empty
producer
full
empty
workers

Ασκηση 2

```
Thread car(){
    mutex lock
    While(can’t enter)
        cond wait (q[0])
    mutex unlock
    cross()
    if (can wake someone from ur lane)
        wake (q[0])
    else if(can wake someone from the other lane)
        wake(q[1])
    else if (no one waiting)
        reset()
    mutex unlock
}
```

Conditions for waiting

- `cars == capacity`

bridge is full

- `cars > 0 && direction != bridge_direction`

there are some cars on the bridge but the have opposite direction

- `crosses+cars >= 2*capacity && waiter[1-direction]!=0)`

count crosses to avoid starvation when 2*bridge size cars have crossed the bridge change direction

Condition when exiting

Crossed condition for changing direction

- `waiter[direction] > 0 && ( cars+crosses<2*capacity || waiter[1-direction]==0)`

Occurs when cars are waiting on ur lane and the crosses cond is ok

- `cars == 0 && waiter[1-direction] > 0 &&`

`(waiter[direction]==0 || waiter[1-direction]+crosses >= 2*capacity )->crosses cond`

Occurs when ur the last of ur lane and someone is waiting on the other side

- `cars == 0 && waiter[1-direction] == 0 && waiter[direction] == 0`

Occurs where ur the last of ur lane but no one is watitng so u reset the bridge

Ασκηση 3

Waiting // flag that the car is waiting

```
Rollecoaster(){
    while(){
    mutex lock
    for(i < capacity)
        cond signal(board)
    waiting = 1
    cond wait(full)
    mutex unlock

    ride()

    mutex lock
    for(i < capacity)
        cond signal(unboard)
    cond wait(full)
    mutex unlock
}
```

If the car is waiting u don't want to wait for the car

Last passenger to board

Last passenger to unboard

```
Passenger(){
    mutex lock
    if(waiting == 0)
        cond wait(board)
    boarded++
    if(boarded == car_size){
        cond signal(&full)
        waiting = 0;
    }
    cond wait(&unboard)
    boarded--
    if(boarded == 0)
        cond signal(full)
    mutex_unlock
}
```

Ασκηση 4

```
Woman(){
    Woman enters()
    Spend time on wc
    Woman exits()
}
Man(){
    Man enters()
    Spend time on wc
    Man exits()
}
```

```
Man enters(){
    mutex lock
    while(inside == capacity || (inside>0 && turn!=0) || (waiter[1]!=0))
        cond wait(q[0])
    mutex unlock
}


Man exit(){
    mutex lock
    if(waiter[0]>0 && waiter[1]==0)
        cond signal(q[0])
    else if(inside==0 && waiter[1]>0)
        for(i=0; i<capacity; i++)
            cond signal(q[1])
    mutex unlock
}
```

```
Woman enters(){
    mutex lock
    while(inside == capacity || (inside>0 && turn!=1))
        cond wait(q[1])
    mutex unlock
}

Wonan exits(){
    mutex lock
    if(waiter[1]>0)
        cond signal(&q[1]);
    else if(inside==0){
        for(i=0; i<capacity; i++)
            cond signal(&q[0]);
    }
    mutex unlock
}
```