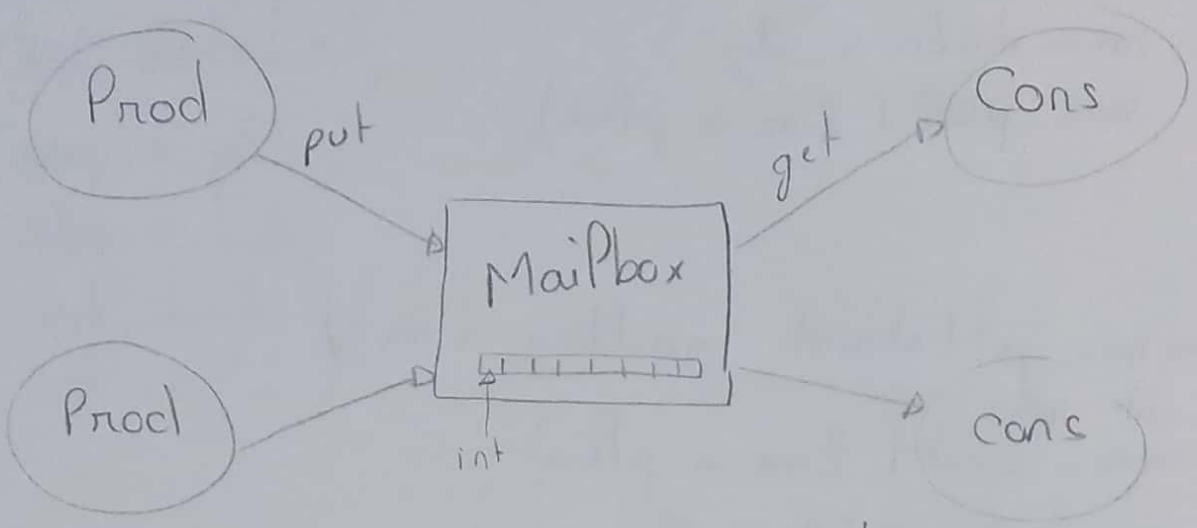
TD5 : Producteur / Consommateur



On ne sait pas si les producteurs et les consommateur vont à la même vitesse

Q1&2.

```
struct mailbox {
    int data;
    sem_t plein;
    sem_t vide;

}
void mbox_init(struct mailbox * m){
    sem_init(&m->vide, 0, 1);
    sem_init(&m->plein, 0, 0);

}
```

```
void mbox_put(struct mailbox *m, int d){
    sem_wait(&m->vide);
    m->data = d;
    sem_post(&m->plein);
}

int mbox_get(struct mailbox *m){
    int res;
    sem_wait(&m->plein);
    res = m->data;
    sem_post(&m->vide);
    return res;
}
```

Q1.3

```
#define N

struct mailbox {
  int data[N];
  sem_t plein;
  sem_t vide;
  int head;
  int tail;
}

void mbox_init (struct mailbox * m){
    sem_t (&m -> plein, 0, N);
    sem_t (&m -> vide, 0, 0);
    m -> head = 0;
    m -> tail = 0;
}

void mbox_put (struct mailbox *m, int d){
    sem_wait
    m -> data [m -> head] = d;
    m -> head = ((m -> head) + 1) % N;
    sem_post (&m -> plein);
}

int mbox_get (struct mailbox *m) {
    int res;
    sem_wait (&m -> plein);
    res = m -> data[m -> tail];
    m -> tail = ((m -> tail) + 1) % N;
    return res;
}
```

Q1.4

```
struct mailbox {
    idem Q1.3
    +
    sem_t mutex;
}

void mbox_init (____ *m) {
    idem Q1.3
    +
    sem_init (&m->mutex, 0, 1);
}

void mbox_put (____, ____) {
    sem_wait (&m->vide);
    sem_wait (&m->mutex);
    m->data[m->head] = d;
    m->head = (m->head+1) % N;
    sem_post (&m->mutex);
    sem_post (&m->plein);
}

int m___get (____) {
    int res;
    sem_wait (&m->plein);
    sem_wait (&m->mutex);
    |;
    sem_wait (&m->mutex);
    sem_post (&m->vide);
    return res;
```

toujours à 1 avec un mutex