```cpp
/** PCO 2015 YTA Buffer2Conso. Tampon simple, 1 producteur - 2 consommateurs.
 * --- 1ere solution ---
 * Objectif: parfaitement fonctionnel.
 * Commentaire: pour un put, 6 appels de synchro nécessaires.
 * Non-scalable pour nombre de lectures N!=2, p.ex. 1000 -> problème d'optimisation
template<typename T> class Buffer2Conso : public AbstractBuffer<T> {
public:
    Buffer2Conso();
    virtual ~Buffer2Conso();
    virtual void put(T item);
    virtual T get(void);
private:
    T element;
    QSemaphore* waitEmpty;
    QSemaphore* waitFull;
    QSemaphore* mutex;
}

Buffer2Conso::Buffer2Conso() {
    waitEmpty = new QSemaphore();
    waitFull = new QSemaphore();
    mutex = new QSemaphore();
    waitEmpty->release(); // ok pour écriture (ouvert)
    waitEmpty->release();
    mutex->release(); // libre
}

Buffer2Conso::~Buffer2Conso() {
    delete waitEmpty;
    delete waitFull;
    delete mutex;
}

void Buffer2Conso::put(T item) {
    // mutex empeche 2 put de faire un deadlock entre eux.
    // celui qui a commencé doit attendre que 2 lectures aient lieu.
    mutex->acquire(); // prelude
    waitEmpty->acquire();
    waitEmpty->acquire();
    mutex->release();

    element = item;  // section critique

    waitFull->release(); // postlude
    waitFull->release();
}

T Buffer2Conso::get(void) {
    T item;
    waitFull->acquire(); // prelude
    item = element; // section critique
    waitEmpty->release(); // postlude
    return item;
}
```