Задания к лабораторной работк №3.

*При реализации **синхронизации** процессов использовать функции ожидания сигнального состояния объекта только с **равным нулю или бесконечности интервалом** ожидания. Каждый отдельный процесс открывать в **отдельном консольном окне**.*

3.1. Написать программы для консольного процесса Boss (Резидент) и консольных процессов Scout (Шпион). *Для моделирования передачи сообщений ввести специальные события, которые обозначают «точку» и «тире», конец сеанса.*

Процесс **Boss**:

запрашивает у пользователя количество процессов Scout, которые он должен запустить; запускает заданное количество процессов Scout;
принимает от каждого процесса Scout сообщение и выводит его на консоль в одной строке. Принимать сообщение может **только от одного процесса**, передача остальных сообщений от других процессов должна блокироваться с помощью мьютекса;
завершает свою работу.

Процесс **Scout** :

запрашивает с консоли символы: «-», «.» (событие *«тире»,* событие *«точка»)* и передает соответствующие события процессу Boss;
завершает свою работу, когда будет введён символ, обозначающий конец ввода сообщений.

4.2. Написать программы для консольного процесса Boss (Резидент) и консольных процессов Scout (Шпион). *Для моделирования передачи сообщений ввести специальные события, которые обозначают «1» , «2» и конец сеанса для процессов* **Scout**

Процесс **Boss**:

запрашивает у пользователя количество процессов **Scout**, которые он должен запустить; запускает заданное количество процессов **Scout**;
принимает от каждого процесса **Scout** сообщение и выводит его на консоль в одной строке. Принимать сообщение может **только от двух процессов**, передача остальных сообщений от других процессов должна блокироваться с помощью мьютексов;
завершает свою работу.

Процесс **Scout** :

запрашивает с консоли сообщения, состоящее из *«1» , «2»*, и передает их (по одному) процессу Boss;
завершает свою работу.

4.3. Написать программы для консольного процесса Boss и консольных процессов Parent, Child. *Для моделирования передачи сообщений ввести специальные события, которые обозначают любые 4-е цифры и конец сеанса для процессов* **Parent и Child.**

Процесс **Boss**:

запрашивает у пользователя количество процессов **Parent** и количество процессов **Child**, которые он должен запустить;
запрашивает кол-во сообщений, отправленных **Parent и Child**
запускает заданное количество процессов **Parent, Child**;

отправляет сообщения для процессов **Parent, Child** Отправить сообщение может **только трём процессам из всех процессов Child и Parent**, передача остальных сообщений от других процессов должна блокироваться с помощью мьютексов;
завершает свою работу.

Процесс **Parent**:

получает сообщение, от процесса Boss и выводит его на консоль; завершает свою работу.

Процесс **Child** :

получает сообщение, от процесса Boss и выводит его на консоль; завершает свою работу.

3.4. Написать программы для консольного процесса Boss (Резидент) и консольных процессов Scout (Шпион).
Для моделирования передачи сообщений ввести специальные события, которые обозначают любые 4-е цифры.

Процесс **Boss**:

запрашивает у пользователя количество процессов Scout, которые он должен запустить; запрашивает у пользователя пароль (3 цифры);

запускает заданное количество процессов Scout;

принимает от каждого процесса Scout сообщение и выводит его на консоль в одной строке. Принимать сообщение может **только от трёх процессов**, передача остальных сообщений от других процессов должна блокироваться;

если приходит сообщение, с цифрой не из пароля, то выводит на консоль текст "ошибка"; завершает свою работу.

Процесс **Scout** :

запрашивает с консоли сообщение, состоящее из цифр, и передает их (по одному) процессу Boss;

завершает свою работу.

4.5. Написать программы для консольного процесса Boss и консольных процессов Parent, Child. *Для моделирования передачи сообщений ввести специальные события, которые обозначают «A» , «B» и конец сеанса для процессов* **Parent и Child.**

Процесс **Boss**:

запрашивает у пользователя количество процессов **Parent** и количество процессов **Child**, которые он должен запустить;

запускает заданное количество процессов **Parent, Child**;

запрашивает кол-во сообщений, полученных от **Parent или Child**

принимает от каждого процесса **Parent, Child** сообщение и выводит сообщение и кто его отправил на консоль в одной строке. Принимать сообщение может **только от одного процесса Child и одного процесса Parent**, передача остальных сообщений от других процессов должна блокироваться с помощью мьютексов;

завершает свою работу.

Процесс **Parent**:

- запрашивает с консоли сообщения, состоящее из *«A»* и передает их (по одному) процессу Boss;
- завершает свою работу.

Процесс **Child**:

- запрашивает с консоли сообщения, состоящее из *«B» »* и передает их (по одному) процессу Boss;
- завершает свою работу.

4.6. Написать программы для консольного процесса **Administrator** и консольных процессов **Reader и Writer**.
Для моделирования передачи сообщений ввести специальные события, которые обозначают сообщение "A" , сообщение "B", и конец сеанса для процессов **Reader и Writer**.

Одновременно принимать и отправлять сообщения могут **только два процесса Writer** и **два процесса Reader**, передача остальных сообщений от других процессов должна блокироваться с помощью мьютексов;

Процесс **Administrator**:

запрашивает у пользователя количество процессов **Writer(Reader)**;

запрашивает у пользователя кол-во отправленных (полученных) сообщений для процессов **Writer (Reader)**;

запускает заданное количество процессов **Reader и Writer**;

принимает от каждого процесса **Writer** сообщение и выводит на консоль, затем отправляет его процессу **Reader.**

принимает от каждого процесса **Reader и Writer** сообщение о завершении сеанса и выводит его на консоль в одной строке.

завершает свою работу.

Процесс **Writer**:

запрашивает с консоли сообщения, состоящее из *"A" , "B",* и передает их (по одному) процессу **Administrator**;

передает сообщение о завершении сеанса процессу **Administrator**; завершает свою работу.

Процесс **Reader**:

принимает сообщение от процесса **Administrator**;

выводит на консоль сообщение;

передает сообщение о завершении сеанса процессу **Administrator**; завершает свою работу.

4.7. Написать программы для консольного процесса Boss и консольных процессов Parent, Child. *Для моделирования передачи сообщений ввести специальные события, которые обозначают «A» , «B», «C» , «D» и конец сеанса для процессов* **Parent и Child.**

Процесс **Boss**:

запрашивает у пользователя количество процессов **Parent** и количество процессов **Child**, которые он должен запустить;

запускает заданное количество процессов **Parent, Child**;

запрашивает кол-во сообщений, принятых от **Parent или Child**

принимает от каждого процесса **Parent, Child** сообщение и выводит сообщение и кто его отправил на консоль в одной строке. Принимать сообщение может **только от двух процессов Child и одного процесса Parent**, передача остальных сообщений от других процессов должна блокироваться с помощью мьютексов;

завершает свою работу.

Процесс **Parent**:

- запрашивает с консоли сообщения, состоящее из *«A» , «B»* и передает их (по одному) процессу Boss;
- завершает свою работу.

Процесс **Child**:

- запрашивает с консоли сообщения, состоящее из *«C» , «D»* и передает их (по одному) процессу Boss;
- завершает свою работу.

4.8. Написать программы для консольного процесса **Administrator** и консольных процессов **Reader и Writer**.

Для моделирования передачи сообщений ввести специальные события, которые обозначают сообщение "A" , сообщение "B", и конец сеанса для процессов **Reader и Writer**.

Одновременно принимать и отправлять сообщения могут **только один процесс Writer** и **один процесс Reader**, передача остальных сообщений от других процессов должна блокироваться с помощью мьютексов;

Процесс **Administrator**:

запрашивает у пользователя количество процессов **Reader и Writer**, которые он должен запустить;

запрашивает у пользователя кол-во отправленных сообщений для процесса **Writer** и кол-во принятых сообщений для процесса **Reader(соответствие сообщений проверить и подкорректировать по формуле)**;

запускает заданное количество процессов **Reader и Writer**;

принимает от каждого процесса **Reader и Writer** сообщение о завершении сеанса и выводит его на консоль в одной строке.

завершает свою работу.

Процесс **Writer**:

запрашивает с консоли сообщения, и передает их (по одному) процессу **Reader**; передает сообщение о завершении сеанса процессу **Administrator**;

завершает свою работу.

Процесс **Reader**:

принимает сообщение от процесса **Writer**; выводит на консоль сообщение;

передает сообщение о завершении сеанса процессу **Administrator**; завершает свою работу.

4.9. Написать программы для консольного процесса **Boss** и консольных процессов **Employee**. *Для моделирования передачи сообщений ввести специальные события, которые «0» , «1», «2», «3» и конец сеанса для процессов* **Employee** .

Процесс **Boss**:

запрашивает у пользователя количество процессов **Employee**, которые он должен запустить; запускает заданное количество процессов **Employee**;

принимает от каждого процесса **Employee** сообщение и выводит его на консоль в одной строке. Принимать сообщение может **только от трёх процессов**, передача остальных сообщений от других процессов должна блокироваться с помощью мьютексов;

завершает свою работу.

Процесс **Employee**:

запрашивает с консоли сообщения, состоящее из *«0» , «1», «2», «3»*, конец сеанса работы и передает (по одному) его процессу **Boss**;
завершает свою работу.

4.10 Написать программы для консольного процесса **Administrator** и консольных процессов **Reader и Writer**.
Для моделирования передачи сообщений ввести специальные события, которые обозначают сообщение "A" , сообщение "B", и конец сеанса для процессов **Reader и Writer**.
Одновременно принимать и отправлять сообщения могут **только два процесса Writer** и **два процесса Reader**, передача остальных сообщений от других процессов должна блокироваться с помощью мьютексов;

Процесс **Administrator**:

запрашивает у пользователя количество процессов **Reader и Writer**, которые он должен запустить;
запрашивает у пользователя кол-во отправленных сообщений для процесса **Writer**. Кол-во принятых сообщений для процесса **Reader** вычислить**. (соответствие сообщений проверить и подкорректировать по формуле)**;
запускает заданное количество процессов **Reader и Writer**;
принимает от каждого процесса **Reader и Writer** сообщение о завершении сеанса и выводит его на консоль в одной строке.
завершает свою работу.

Процесс **Writer**:

запрашивает с консоли сообщения, и передает их (по одному) процессу **Reader**; передает сообщение о завершении сеанса процессу **Administrator**;
завершает свою работу.

Процесс **Reader**:

принимает сообщение от процесса **Writer**; выводит на консоль сообщение;
передает сообщение о завершении сеанса процессу **Administrator**; завершает свою работу.

4.11 Написать программы для консольного процесса **Administrator** и консольных процессов **Reader и Writer**.
Для моделирования передачи сообщений ввести специальные события, которые обозначают сообщение "A" , сообщение "B", и конец сеанса для процессов **Reader и Writer**.
Одновременно принимать и отправлять сообщения могут **только один процесс Writer** и **два процесса Reader**, передача остальных сообщений от других процессов должна блокироваться с помощью мьютексов;

Процесс **Administrator**:

запрашивает у пользователя количество процессов **Reader и Writer**, которые он должен запустить;
запрашивает у пользователя кол-во отправленных сообщений для процесса **Writer**. Кол-во принятых сообщений для процесса **Reader** вычислить**. (соответствие сообщений проверить и подкорректировать по формуле)**;
запускает заданное количество процессов **Reader и Writer**;
принимает от каждого процесса **Reader и Writer** сообщение о завершении сеанса и выводит его на консоль в одной строке.
завершает свою работу.

Процесс **Writer**:

запрашивает с консоли сообщения, и передает их (по одному) процессу **Reader**; передает сообщение о завершении сеанса процессу **Administrator**;
завершает свою работу.

Процесс **Reader**:

принимает сообщение от процесса **Writer**; выводит на консоль сообщение;
передает сообщение о завершении сеанса процессу **Administrator**; завершает свою работу.