**Лабораторная работа №1**

**Задача.** Написать программу для консольного процесса, который состоит из двух потоков: main и worker. Поток **main** должен выполнить следующие действия:

1. Создать массив(тип массива согласно варианту) чисел, размерность и элементы которого вводятся с консоли( или сгенерировать случайно).
2. Для некоторых вариантов ввести дополнительные парамтры : k,a,b, х.
3. Ввести время для остановки и запуска потока worker.
4. Создать поток worker, передать в поток данные: размер маcсива, масив и т.д.
5. Приостановить поток **worker(**SuspendThread), затем через некоторое время снова запустить поток.
6. Уметь создавать поток командой \_beginthreadex
7. Дождаться завершения потока worker.
8. Вывести на консоль результат работы потока worker
9. Завершить работу.

Поток **worker** должен выполнить следующую работу:

Вывести отрицательные элементы кратне 9 из массива. Тип элементов - unsigned short. После каждой итерации цикла - «спать» 30 миллисекунд. Завершить свою работу.

**Дополнительное задание:**

* + 1. Добавить третий поток **Count;**
    2. Создать поток **Count** в потоке main, в подвешенном состоянии.
    3. Запустить поток **Count**. Поток **Count** выполняет:

Выводит на консоль числа фибоначчи, по возрастанию.

**Лабораторная работа №2**

**Задача**. Написать программы двух консольных процессов Parent и Child, которые выполняют следующие действия.

## Процесс Parent:

* + Ввести размер массива, ввести элементы массива;
  + Формирует командную строку, которая содержит информацию об размерности массива, элементах и т.д. (согласно индивидуальному варианту);
  + Для консоли дочернего прооцесса устанавливает визуальные настройки

Установить ширину окна для Сhild или заголовок окна.

* + Запускает дочерний процесс Child, которому через командную строку передается информация об размерности массива, элементах и т.д. (согласно варианту);

## Процесс Child:

Сортировка методом “пузырька”. Полученный массив вывести. Тип элементов - символы.

**Дополнительное задание:**

1. завершить процесс с помощью функции TerminateProcess
2. завершить процесс Parent с помощью функции ExitProcess;
3. Запустить 2-й процесс **Count из Parent**. У процесса **Count** менять приоритет. Проецесс **Count** выводит на консоль числа фибоначчи, по возрастанию.

**Лабораторная работа №3.**

Написать программу для консольного процесса, который состоит из трёх потоков: **main** , **work, и третьего** (см. варианты)..

## Объекты синхронизации :

***Критическая секция №1-*** синхронизация потока ***work и*** вывод в потоке ***main*** (для вывода части массива в ***main***)***; Событие*** устанавливает поток ***main*** для потока ***SumElement*** *(сигнализирует о начале суммирования).*

***Критическая секция №2-*** синхронизация ***SumElement и*** *потока* ***main (***вывод ***результа SumElement); Поток*** *main* ***должен выполнить следующие действия:***

* + - Инициализировать необходимые события и критические секции.
    - создать массив, размерность и элементы которого вводятся пользователем с консоли;
    - вывести размерность и элементы исходного массива на консоль;
    - запустить поток **work (** в подвешенном состоянии);
    - ввести число k;
    - запустить поток **work**
    - запустить поток **SumElement**;
    - Приостановить работу потока на 50 мс (Sleep)
    - Получить от потока **work** сигнал о начале суммирования (момент запуска произойдёт после того, будут выведены на консоль k элементов) (*использовать* ***критическую секцию****)*;
    - вывести на экран элементы массива (итогового до k элементов);
    - известить поток **SumElement** о начале суммирования (момент запуска произойдёт после того, будут выведены на консоль k элементов) (*использовать* ***событие***);.
    - вывести на экран результат работы потока **SumElement**;
    - вывести на экран элементы массива (итогового после k элементов);

Поток **work** должен выполнить следующие действия:

* + - запросить у пользователя временной интервал, требуемый для отдыха после подготовки одного элемента в массиве;
    - Сортировка методом “пузырька”. Элементы - вещественные числа двойной точности.
    - после каждого готового элемента отдыхать в течение заданного интервала времени;
    - известить поток **main** о начале суммирования (момент запуска произойдёт после того, будут отсортированы k элементов) (*использовать* ***критическую секцию****)*;

Поток **SumElement** должен выполнить следующие действия (*Для синхронизации с потоком main****,*** *использовать* ***событие***

## и критическую секцию*!*):

* + - ждёт от потока **main** сигнал о начале суммирования (*использовать* ***событие***);
    - выполнить суммирование элементов итогового массива до заданной позиции k;
    - вывести итоговую сумму.
* известить(*использовать* ***критическую секцию****)* поток **main** о выводе результата

**Лабораторная работа №4.**

*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****. Использовать функцию WaitForMultipleObject для ожидания одного из группы событий.*

*ПЕРЕДАЧА СООБЩЕНИЙ : Отправить сообщение, например, А или В от одного процесса другому, в данном задании означает : создаем события соответсвующие сообщениям А и В. Затем вводится одно из сообщений (А или В) с консоли в одном процессе и устанавливается соответсвующее событие в сигнальное состояние. В другом процессе ожидается одно из событии и выводится на консоль соответствующее сообщение..*

АКТИВНЫЙ процесс**- процесс, который может отправить сообщение, введённое с консоли и получить сообщение.**

Написать программы для консольного процесса **Administrator** и консольных процессов **Reader и Writer**. *Для моделирования передачи сообщений ввести специальные события(***c автоматическим сбросом)***, которые обозначают сообщение “A”(два события) , сообщение “B” (два события), сообщение “С”(два события и конец сеанса для процессов* **Reader и Writer***.*

Одновременно принимать и отправлять сообщения могут **только ОДИН АКТИВНЫЙ процесс Writer(использовать мьютекс)** и **ОДИН АКТИВНЫЙ процесса Reader(использовать семафор)**, передача остальных сообщений от других процессов должна временно блокироваться (находиться в режиме ожидания).

Процесс **Administrator**:

* + - Инициализация объектов синхронизации;
    - запрашивает у пользователя количество процессов **Writer( Reader)**;
    - запрашивает у пользователя кол-во отправленных сообщений процессом **Writer(**и принятых процессом

## 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**:

* + - синхронизировать работу процессов **Writer** с помощью **мьютекса**
    - передачу сообщений реализовать с помощью **событий**
    - запрашивает с консоли сообщения, состоящее ( *“A” или “B” или “С”)* и передает их (по одному) процессу **Administrator**;
    - передает сообщение о завершении сеанса процессу **Administrator**;
    - завершает свою работу.

Процесс **Reader**:

* + - синхронизировать работу процессов **Reader** с помощью **семафора**
    - передачу сообщений реализовать с помощью **событий**
    - принимает сообщения от процесса **Administrator**;
    - выводит на консоль сообщение;
    - передает сообщение о завершении сеанса процессу **Administrator**;
    - завершает свою работу.

**Лабораторная работа №5**

Написать программы для консольных процессов **Server и Sum,** которые обмениваются сообщениями по анонимному каналу.

### Одновременно сообщение может передаваться только одним из процессов.

**Процесс- Server**, который выполняет следующие действия:

* + - Размер массива вводится с консоли. Тип массива: **double**
    - Генерирует элементы массива
    - Запускает процесс **Sum**.
    - Запрашивает с консоли число N.
    - Передает число N, размер массива процессу **Sum**
    - Передаёт массив процессу **Sum**.
    - Получает массив от процесса- **Sum** .
    - Выводит переданную и полученную информацию по каналу на консоль.

**Процесс-Sum**, который выполняет следующие действия.

* + - Получает число N, размер массива, массив по анонимному каналу от процесса-сервера
    - Находит числа в массиве >N
    - Выводит полученный массив на консоль.
    - Вычисляет сумму квадратов чисел масcива, больших N
    - Передаёт квадраты элементов массива по анонимному каналу процессу-серверу.
    - Передаёт cумму по анонимному каналу процессу-серверу.
    - Выводит сумму на консоль.

**Лабораторная работа №6**

## Сдать до 8.05.

**переделать лаб. №5, изменить для именованных каналов.**

**Объекты синхронизации не использовать.**

**Процессы должны работать на разных компьютерах!**

**Имя канала на сервере : . (точка - имя сервера) в имени канала (например: \\\\.\\pipe\\demo\_pipe ).**

**Имя компьютера, на котором ваше приложение подключается к каналу: fpmi506pc14 (например)**

**Лабораторная работа №7**

**Тема: "Разработка монитора для синхронизации параллельного доступа к кольцевой очереди(стеку)" (варианты смотреть ниже).**

1. Написать монитор для управления параллельным доступом к очереди или стеку (***Согласно вариантам - см. ниже***)**:**

## Интерфейс монитора для очереди:

class MonitorQueuer

{

MonitorQueue(int nSize); // nSize - размер очереди

~MonitorQueue();

void AddHead (type & nElement); // добавить элемент в «голову» очереди char RemoveTail(); // удалить элемент из «хвоста» очереди

};

## Интерфейс монитора для стека:

class MonitorStack

{

MonitorStack(int nSize); // nSize - размер стека

~MonitorStack();

void Push(type & nElement); // добавить элемент в стек char Pop(); // удалить элемент из стека

};

1. Кольцевая очередь (стек) реализуется массивом. Размер кольцевой очереди задается в конструкторе.
2. Если поток обращается к методу Add… (Push) монитора, а очередь (стек) оказывается в этот момент полной, то монитор переводит поток в состояние ожидания до тех пор, пока в очереди (в стеке) не окажутся свободные ячейки.
3. Если поток обращается к методу Remove (Pop) монитора, а очередь(стек) оказывается в этот момент пустой, то монитор переводит поток в состояние ожидания до тех пор, пока в очереди (в стеке) не окажется новый элемент.

## Дополнительные требования:

Для тестирования монитора написать программу для консольного процесса, который состоит из потока main и нескольких потоков consumer и producer.

Поток **main** должен выполнять следующие действия:

* + создать монитор очереди (стека), размер которой вводится пользователем с клавиатуры;
  + [Уточнения ] создать объекты синхронизации:

1. для ограничения доступа потоков к верхушке стека или к двум концевым элементам очереди;
2. для блокировки потоков producer - если стек(очередь) заполнен(а);
3. для блокировки потоков consumer - если стек(очередь) пуст(а);
   * ввести с клавиатуры количество потоков producer и количество потоков consumer, которые он должен запустить;
   * для каждого из потоков producer и consumer ввести количество продуктов (тип массива в индивидуальном варианте), которые эти потоки должны соответственно произвести и потребить;
   * создать требуемое количество потоков producer и consumer;
   * передать в качестве параметров каждому потоку producer кол-во продуктов (целое число и сам продукт (элемент ), который поток должен произвести (записать в очередь\стек);
   * передать в качестве параметра каждому потоку consumer положительное целое число, которое обозначает количество продуктов, которое поток должен потребить (удалить из очереди\стека);
   * завершить свою работу после окончания работы всех потоков producer и consumer.

Поток **producer** должен выполнять следующие действия:

* произвести (записать в очередь\стек) требуемое количество экземпляров продукта ;
* после добавления очередного числа в очередь(стек) выводить на консоль сообщение следующего вида: "Произведен элемент (продукт) N ", где N ,записано в очередь\стек.

Поток **consumer** должен выполнять следующие действия:

* потребить (удалить из очереди\стека) заданное количество продуктов;
* при извлечении очередного символа из очереди (стека), выводить на консоль сообщение следующего вида: "\tУпотреблен элемент (продукт) N ", где N – элемент (продукт) который извлечен из очереди\стека.

# Варианты индивидуальных заданий:

Доступ к стеку. Элемент (продукт) -double.