**Лабораторная работа №5.**

***Цель работы:*** получить практические навыки создания многопоточных приложений.

***Содержание работы:***

1. Разработать многопоточную программу для указанного варианта.
2. В приложении должен быть главный поток, порожденные потоки выполняющие определенный процесс и поток, выводящий на экран результаты.

***Формат сдачи лабораторной работы:***

1. Отчет с листингами программ и скриншотами выполнения программы.
2. Вывод по лабораторной работе.
3. Отчеты высылать на почту преподавателя с указанием темы: JAVA <Номер группы>_<ФИО>_Лаб<номер лабораторной>
4. При возникновении вопросов связываться на занятии, либо через телеграмм или почту.

***Варианты заданий:***

1. В директории лежат входные текстовые файлы, проименованные следующим образом: in_<N>.dat, где N - натуральное число. Каждый файл состоит из двух строк. В первой строке - число, обозначающее действие, а во второй - числа с плавающей точкой, разделенные пробелом. Действия могут быть следующими: 1 – сложение, 2 – умножение, 3 - сумма квадратов

   Необходимо написать многопоточное приложение, которое выполнит требуемые действия над числами и сумму результатов запишет в файл out.dat. Название рабочей директории передается в виде аргумента рабочей строки. В реализации приветствуется использование полиморфизма и паттернов проектирования.
2. Разработайте многопоточное приложение, выполняющее вычисление произведения матриц A (m×n) и B (n×k). Элементы cij матрицы произведения C = A×B вычисляются параллельно p однотипными потоками. Если некоторый поток уже вычисляет элемент cij матрицы C, то следующий приступающий к вычислению поток выбирает для расчета элемент cij+1, если j<k, и ci+1k, если j=k. Выполнив вычисление элемента матрицы-произведения, поток проверяет, нет ли элемента, который еще не рассчитывается. Если такой элемент есть, то приступает к его расчету. В противном случае отправляет (пользовательское) сообщение о завершении своей работы и приостанавливает своё выполнение. Главный поток, получив сообщения о завершении вычислений от всех потоков, выводит результат на экран и запускает поток, записывающий результат в конец файла-протокола. В каждом потоке должна быть задержка в выполнении вычислений (чтобы дать

возможность поработать всем потокам). Синхронизацию потоков между собой организуйте через критическую секцию или мьютекс.

3. Написать многопоточное приложение эмулирующее работу библиотеки. В библиотеке доступны к чтению несколько книг. Некоторые из них можно выдавать на руки некоторые только в читальный зал. Посетители могут брать одновременно по несколько книг на руки и в читальный зал.
4. Написать клиент для работы порта. Корабли могут заходить в порт для разгрузки / загрузки контейнеров. Количество контейнеров, находятся в текущий момент в порту и на корабле, должно быть неотъемлемой и не превышать заданную грузоподъемность судна и емкость порта. В порту работает несколько причалов. В одном причале может стоять только один корабль. Корабль может загружаться у причала, разгружаться или выполнять оба действия.
5. Напишите программу, которая каждую секунду отображает на экране данные о времени, прошедшем от начала сессии, а другой её поток выводит сообщение каждые 5 секунд. Предусмотрите возможность ежесекундного оповещения потока, воспроизводящего сообщение, потоком, отсчитывающим время. Не внося изменений в код потока-"хронометра", добавьте ещё один поток, который выводит на экран другое сообщение каждые 7 секунд.
6. Эмулировать работу следующего процесса «Warcraft». Заданное количество юнитов добывают золото равными порциями из одной шахты, задерживаясь в пути на случайное время, до ее истощения. Работа каждого юнита реализуется в порожденном процессе (потоке).
7. Эмулировать работу следующего процесса «Винни-Пух и пчелы». Заданное количество пчел добывают мед равными порциями, задерживаясь в пути на случайное время. Винни-Пух потребляет мед порциями заданной величины за заданное время и столько же времени может прожить без питания. Работа каждой пчелы реализуется в порожденном процессе (потоке).
8. Бег с препятствиями. Создается условная карта трассы в виде матрицы, ширина которой соответствует количеству бегунов, а высота – фиксирована, содержащей произвольное количество единиц (препятствий) в произвольных ячейках. Стартующие бегуны (процессы, потоки) перемещаются по трассе и при встрече с препятствием задерживаются на фиксированное время. По достижении финиша бегуны сообщают свой номер.

9. Противостояние нескольких команд. Каждая команда увеличивается на случайное количество бойцов и убивает случайное количество бойцов участника. Борьба каждой команды реализуется в отдельном процессе (потоке).
10. Несколько потоков работают с общим одноэлементным буфером. Потоки делятся на "писателей", осуществляющих запись сообщений в буфер, и "читателей", осуществляющих извлечение сообщений из буфера. Только один поток может осуществлять работу с буфером. Если буфер свободен, то только один писатель может осуществлять запись в буфер. Если буфер занят, то только один читатель может осуществлять чтение из буфера. После чтения буфер освобождается и доступен для записи. В качестве буфера используется глобальная переменная, например, типа string. Работа приложения заканчивается после того, как все сообщения писателей через общий буфер будут обработаны читателями.
11. Задача об "обедающих философах", предложенная Э. Дейкстрой. Опуская некоторые художественные подробности задачи, задача представляется в следующей формулировке.

    Пять философов приглашены на обед. Их посадили за круглый стол, перед каждым из них стоит блюдо спагетти, справа от блюда лежит вилка. Как полагается истинным философам, гости не столько едят, как предаются размышлениям. Когда философ вспоминает о еде, то он берет свою вилку, но вскоре обнаруживает, что есть спагетти одной вилкой невозможно. Если вилка соседа свободна, то он берет вилку соседа и ест, пока опять не погружается в размышления, освобождая обе вилки. Если же вилка соседа занята, то философ ожидает ее освобождения, не выпуская свою вилку.
12. Реализовать многопоточное приложение, реализующее вывод всех четных слов из списка файлов. Для каждого файла создается новый поток, но общее число потоков не должно превышать 10.
13. Реализовать многопоточное приложение, реализующее поиск подстроки в файлах. Список файлов передается в качестве параметра командной строки. Для каждого файла выделяется отдельный поток. Для вывода результатов поиска в консоль создается отдельный поток, считывающий данные по мере поступления из разделяемого списка объектов класса SearchResult, имеющего следующего поля «имя файла», «индекс вхождения».
14. Реализовать многопоточное приложение, производящее тестирование целых чисел типа long на простоту в заданном диапазоне. Для этого создается фиксированное количество потоков равное 10-15% от количества элементов, каждый из которых

случайным образом обращается к непроверенному элементу последовательности, проверенные элементы удаляются из списка.

15. Бег с препятствиями. Создается условная карта трассы в виде матрицы, ширина которой соответствует количеству бегунов, а высота – фиксирована, содержащей произвольное количество единиц (препятствий) в произвольных ячейках. Стартующие бегуны (процессы, потоки) перемещаются по трассе и при встрече с препятствием задерживаются на фиксированное время. По достижении финиша бегуны сообщают свой номер.
16. Написать многопоточное приложение эмулирующее работу библиотеки. В библиотеке доступны к чтению несколько книг. Некоторые из них можно выдавать на руки некоторые только в читальный зал. Посетители могут брать одновременно по несколько книг на руки и в читальный зал.
17. Задача об "обедающих философах", предложенная Э. Дейкстрой. Опуская некоторые художественные подробности задачи, задача представляется в следующей формулировке. Пять философов приглашены на обед. Их посадили за круглый стол, перед каждым из них стоит блюдо спагетти, справа от блюда лежит вилка. Как полагается истинным философам, гости не столько едят, как предаются размышлениям. Когда философ вспоминает о еде, то он берет свою вилку, но вскоре обнаруживает, что есть спагетти одной вилкой невозможно. Если вилка соседа свободна, то он берет вилку соседа и ест, пока опять не погружается в размышления, освобождая обе вилки. Если же вилка соседа занята, то философ ожидает ее освобождения, не выпуская свою вилку.
18. Эмулировать работу следующего процесса «Warcraft». Заданное количество юнитов добывают золото равными порциями из одной шахты, задерживаясь в пути на случайное время, до ее истощения. Работа каждого юнита реализуется в порожденном процессе (потоке).