Лабораторная работа №2

Сдать часть 1 до 29.10(часть 2 до 05.11)

Тема: «Синхронизация потоков и управление ресурсами».

Общее задание:

Синхронизация потоков при помощи критических секций и событий. Разрешение ресурсных конфликтов.

Задача: Написать программу, моделирующую работу нескольких потоков-исполнителей (marker), которые конкурируют за общий ресурс (массив), под управлением главного потока (main).

Часть 1

Роль потока main (Управляющий поток):

1. Создать и инициализировать нулями общий массив целых чисел (размер вводится с консоли).
2. Запросить у пользователя количество потоков marker, которые нужно создать.
3. Создать необходимые для синхронизации объекты:
    a. Критическую секцию для защиты доступа к общему массиву.
    b. Массив событий, которые потоки marker будут использовать, чтобы сообщить main о невозможности продолжать работу.
    c. Массив событий для сигнализации потокам marker о необходимости завершиться.
    d. Событие для одновременного запуска всех потоков marker.
4. Запустить N потоков marker. Каждому потоку передать его порядковый номер, размер массива и необходимые объекты синхронизации.
5. Дать общий сигнал на начало работы всем потокам marker.
6. Войти в главный управляющий цикл:

a. Ждать, пока все активные потоки marker не сообщат о невозможности продолжать работу (используя массив событий).

b. Вывести на консоль текущее состояние массива.

c. Запросить у пользователя номер потока marker, который следует завершить.

d. Подать сигнал на завершение выбранному потоку.

e. Дождаться, пока этот поток действительно завершится.

f. Вывести на консоль состояние массива после того, как завершенный поток очистил свои отметки.

g. Подать сигнал на продолжение работы всем оставшимся потокам marker.

Когда все потоки marker будут завершены, освободить все ресурсы и завершить программу.

Роль потока marker (Рабочий поток):

После запуска ожидать сигнала к началу работы от main.

Инициализировать свой генератор случайных чисел (srand) своим порядковым номером.

Войти в основной рабочий цикл:

a. Сгенерировать случайный индекс в пределах размера массива (rand() % array_size).

b. Войти в критическую секцию для эксклюзивного доступа к массиву.

c. Проверить, свободна ли ячейка с этим индексом (array[index] == 0).

* Если свободна: "поспать" 5 мс, записать в ячейку свой порядковый номер, "поспать" еще 5 мс.

* Если занята: выйти из цикла и перейти к шагу 4.

d. Покинуть критическую секцию.

e. Продолжить цикл с шага 3a.

Если рабочий цикл прерван (не удалось найти свободную ячейку):

a. Вывести на консоль свой номер, количество сделанных отметок и индекс ячейки, на которой произошла блокировка.

b. Подать сигнал потоку main о том, что работа приостановлена.

c. Ждать одного из двух сигналов от main: на продолжение или на завершение работы.

При получении сигнала на продолжение: вернуться к выполнению основного рабочего цикла (пункт 3).

При получении сигнала на завершение:

a. Войти в критическую секцию.

b. Найти в массиве все ячейки, отмеченные своим номером, и обнулить их.

c. Покинуть критическую секцию.

d. Завершить свою работу.

Часть 2

Для написания тестов рекомендуется использовать один из фреймворков для модульного тестирования C++, например Google Test.

Потребуется модифицировать вашу программу так, чтобы основную логику можно было вызывать из тестового окружения (например, вынести логику main в отдельную функцию, принимающую параметры вместо интерактивного ввода).

Базовая функциональность (Unit/Integration Tests)

Тест 1.1: Корректная маркировка одним потоком.

Сценарий: Запустить программу с 1 потоком marker и массивом из 10 элементов.

Проверка: Дождаться, когда поток сообщит о невозможности работы. Убедиться, что все 10 элементов массива помечены числом 1.
Тест 1.2: Корректная очистка при завершении.
Сценарий: Повторить сценарий теста 1.1, а затем подать потоку сигнал на завершение.
Проверка: После завершения потока убедиться, что все элементы массива снова равны нулю.

Проверка синхронизации (Concurrency Tests)
Тест 2.1: Отсутствие гонки за ресурс.
Сценарий: Запустить 10 потоков marker с небольшим массивом (например, 20 элементов). Дать им поработать до полной блокировки.
Проверка: Проверить содержимое массива. Каждый элемент должен быть либо 0, либо одним из номеров потока (1..10). В массиве не должно быть "мусорных" значений, которые могли бы возникнуть при одновременной записи. Проверить, что общее количество помеченных ячеек равно размеру массива.
Тест 2.2: Корректное поочередное завершение.
Сценарий: Запустить 5 потоков с массивом на 30 элементов. Дождаться их блокировки. Затем в цикле 5 раз:
Подать сигнал на завершение одному из потоков (например, последнему активному).
Дождаться его завершения.
Промежуточная проверка: Убедиться, что из массива исчезли только отметки завершенного потока, а отметки остальных остались на месте.
Подать остальным сигнал на продолжение.
Проверка: В конце теста массив должен быть полностью пустым (все нули).