ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
Факультет компьютерных наук
Департамент программной инженерии

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

ЗАДАЧА О ЧИТАТЕЛЯХ И ПИСАТЕЛЯХ.

Вариант 3.

Автор пояснительной записки, _____ Доржсурэн Тоголдор
студент ФКН,
“Программная инженерия”,
группа БПИ193

Москва 2020

1. ПОСТАНОВКА ЗАДАЧИ

Текст задания: Базу данных разделяют два типа процессов – читатели и писатели. Читатели выполняют транзакции, которые просматривают записи базы данных, транзакции писателей и просматривают и изменяют записи. Предполагается, что в начале БД находится в непротиворечивом состоянии (т.е. отношения между данными имеют смысл). Каждая отдельная транзакция переводит БД из одного непротиворечивого состояния в другое. Для предотвращения взаимного влияния транзакций процесс-писатель должен иметь исключительный доступ к БД. Если к БД не обращается ни один из процессов-писателей, то выполнять транзакции могут одновременно сколько угодно читателей. Создать многопоточное приложение с потоками-писателями и потоками-читателями. Реализовать решение, используя семафоры.

2. АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ

Для решения как, было указано в задаче использовались семафоры для разделения общей БД между потоками-писателей и потоками-читателей с исключительным доступом для писателей.

База данных в рамках задачи реализована классом <Database> с параметром <T> для типа объекта в контейнере.

```
template<typename T>
class DataBase{
    public:
        static const int capacity = 100;
        T data[capacity];
};
```

В задаче используются два общих семафора:

```
sem_t mutex;            //ensures mutual exclusion
sem_t write_read_sem;   //both reader and writer semaphore
```

Метод потоков-писателей использует семафор <write_read_sem> для критической секции:

```
//Writer thread function to work with databases of <int> type
void* writer_int_transaction(void* param) {
    int index = *((int*)param);
    for(int j = 0; j < total_loops; j++) {
        // writer requests for critical section
        sem_trywait(&write_read_sem);

        // performs the write
        int i = rand() % database.capacity;
        database.data[i] += 1;
        printf("Time: %d. Writer #%d -- wrote value %d at [%d] in database.\n",
        clock(), index + 1, database.data[i], i);

        // leaves the critical section
        sem_post(&write_read_sem);
    }
    return nullptr;
}
```

Метод потоков-читателей использует оба семафора <write_read_sem> и <mutex> для критической секции и для уведомления потоков-писателей, проверяет наличие активных читателей с помощью счётчика <read_count>:

```
//Reader thread function to work with databases of <int> type
void* reader_int_transaction(void* param) {
    int index = *((int*)param);
    for(int j = 0; j < total_loops; j++) {
        // Reader waiting
        sem_trywait(&mutex);

        // Increase number of readers
        read_count++;

        // there is atleast one reader in the critical section
        // this blocks all writers
        if (read_count == 1)
            sem_trywait(&write_read_sem);

        // the critical section
        sem_post(&mutex);
        int i = rand() % database.capacity;
        printf("Time: %d. Reader #%d -- read value %d at [%d] in database.\n", clock(), index + 1, database.data[i], i);
        sem_trywait(&mutex);   // a reader wants to leave

        read_count--;

        // no reader is left in the critical section
        if (read_count == 0)
            sem_post(&write_read_sem);  // writers can enter

        sem_post(&mutex); // reader leaves
    }
    return nullptr;
}
```

3. ОПИСАНИЕ ОБЛАСТИ ДОПУСТИМЫХ ЗНАЧЕНИЙ ВХОДНЫХ ПАРАМЕТРОВ

Программа при запуске запрашивает у пользователя 3 входных параметра:

- количество потоков-писателей (1 < целое число < 5).
- количество потоков-читателей (1 < целое число < 5).
- количество циклов в методах потоков (1 < целое число < 500).

ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

Пример 1

```
Readers-Writers Problem!
Thread functions work on infinite loops, so abort the program in order to quit.
Enter number of writer threads: 2
Enter number of reader threads: 4
Enter number of loops in thread functions: 4
Time: 9713. Writer #1 -- wrote value 36 at [41] in database.
Time: 9713. Writer #2 -- wrote value 37 at [41] in database.
Time: 9747. Writer #2 -- wrote value 78 at [67] in database.
Time: 9789. Writer #2 -- wrote value 7 at [34] in database.
Time: 9797. Writer #2 -- wrote value 30 at [0] in database.
Time: 9714. Reader #3 -- read value 37 at [41] in database.
Time: 9828. Reader #3 -- read value 78 at [67] in database.
Time: 9841. Reader #3 -- read value 7 at [34] in database.
Time: 9713. Reader #1 -- read value 37 at [41] in database.
Time: 9889. Reader #1 -- read value 78 at [67] in database.
Time: 9894. Reader #1 -- read value 7 at [34] in database.
Time: 9714. Reader #4 -- read value 37 at [41] in database.
Time: 9738. Writer #1 -- wrote value 77 at [67] in database.
Time: 9943. Writer #1 -- wrote value 8 at [34] in database.
Time: 9713. Reader #2 -- read value 37 at [41] in database.
Time: 9978. Reader #2 -- read value 78 at [67] in database.
Time: 9992. Reader #2 -- read value 8 at [34] in database.
Time: 9928. Reader #4 -- read value 78 at [67] in database.
Time: 9865. Reader #3 -- read value 30 at [0] in database.
Time: 9961. Writer #1 -- wrote value 31 at [0] in database.
Time: 9713. Reader #1 -- read value 69 at [96] in database.
Time: 9913. Reader #1 -- read value 30 at [0] in database.
Time: 10013. Reader #2 -- read value 31 at [0] in database.
Time: 10030. Reader #4 -- read value 8 at [34] in database.
Time: 10202. Reader #4 -- read value 31 at [0] in database.
Time: 10079. Reader #1 -- read value 92 at [32] in database.
Time: 10228. Reader #1 -- read value 51 at [83] in database.
Time: 10244. Reader #1 -- read value 92 at [32] in database.
```

Пример 2

```
Readers-Writers Problem!
Thread functions work on infinite loops, so abort the program in order to quit.
Enter number of writer threads: 2
Enter number of reader threads: 4
Enter number of loops in thread functions: 3
Time: 10669. Writer #1 -- wrote value 40 at [41] in database.
Time: 10669. Writer #2 -- wrote value 41 at [41] in database.
Time: 10687. Writer #2 -- wrote value 94 at [67] in database.
Time: 10693. Writer #2 -- wrote value 51 at [34] in database.
Time: 10669. Reader #1 -- read value 56 at [93] in database.
Time: 10708. Reader #1 -- read value 41 at [89] in database.
Time: 10716. Reader #1 -- read value 46 at [32] in database.
Time: 10678. Writer #1 -- wrote value 93 at [67] in database.
Time: 10742. Writer #1 -- wrote value 52 at [34] in database.
Time: 10669. Reader #2 -- read value 41 at [41] in database.
Time: 10760. Reader #2 -- read value 94 at [67] in database.
Time: 10769. Reader #2 -- read value 52 at [34] in database.
```

Пример 3

```
Readers-Writers Problem!
Thread functions work on infinite loops, so abort the program in order to quit.
Enter number of writer threads: 8
Wrong input. It must be in this bounds: [1, 5]
Enter number of writer threads: 5
Enter number of reader threads: 10
Wrong input. It must be in this bounds: [1, 5]
Enter number of reader threads: -1
Wrong input. It must be in this bounds: [1, 5]
Enter number of reader threads: 3
Enter number of loops in thread functions: 1000
Wrong input. It must be in this bounds: [1, 500]
Enter number of loops in thread functions: 0
Wrong input. It must be in this bounds: [1, 500]
Enter number of loops in thread functions: 1
Time: 25476. Writer #1 -- wrote value 94 at [41] in database.
Time: 25476. Writer #2 -- wrote value 95 at [41] in database.
Time: 25476. Writer #3 -- wrote value 96 at [41] in database.
Time: 25476. Writer #4 -- wrote value 97 at [41] in database.
Time: 25476. Writer #5 -- wrote value 98 at [41] in database.
Time: 25476. Reader #1 -- read value 40 at [24] in database.
```

4. СПИСОК ИСТОЧНИКОВ

1. Семафоры, онлайн ресурс. Доступ: https://learnc.info/c/pthreads_semaphores.html
2. Producer–consumer problem, Доступ: https://en.wikipedia.org/wiki/Producer%E2%80%93consumer_problem
3. Многопоточное программирование. Синхронизация, онлайн ресурс. Доступ: http://softcraft.ru/edu/comparch/practice/thread/02-sync/