Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Лабораторная работа № 5_2

по курсу «Распределение параллельных и распределённых программ»

«Синхронизация потоков»

Студент группы ИУ9-51Б Горбунов А. Д.

Преподаватель Царёв А. С.

Москва 2024

1 Код программы

Файл main.cpp:

```
#include <iostream>
#include <thread>
#include <vector>
#include <shared_mutex>
#include <mutex>
#include <unordered_set>
#include <random>
#include <list>

using namespace std;

shared_mutex list_mutex;
list<int> shared_list;
unordered_set<int> unique_numbers;

void generate_and_insert(int num_threads, int num_numbers) {
    random_device rd;
    mt19937 gen(rd());
    uniform_int_distribution<> dis(0, 1000);

    for (int i = 0; i < num_numbers; i++) {
        int number = dis(gen);

        shared_lock<std::shared_mutex> read_lock(list_mutex);
        if (unique_numbers.find(number) == unique_numbers.end()) {
            read_lock.unlock();

            unique_lock<std::shared_mutex> write_lock(list_mutex);
            if (unique_numbers.find(number) == unique_numbers.end()) {
                shared_list.push_back(number);
```

```
                unique_numbers.insert(number);
            }
        }

    }
}

bool check_for_duplicates() {
    shared_lock<shared_mutex> read_lock(list_mutex);
    unordered_set<int> temp_set;
    for (const auto& number : shared_list) {
        if (temp_set.find(number) != temp_set.end())
            return false;
        temp_set.insert(number);
    }
    return true;
}

void print_list() {
    shared_lock<shared_mutex> read_lock(list_mutex);
    cout << "List contents: ";
    for (const auto& number : shared_list)
        cout << number << " ";
    cout << std::endl;
}

int main() {
    int num_threads = 4;
    int num_numbers_per_thread = 10000;

    vector<thread> threads;
    for (int i = 0; i < num_threads; i++)
        threads.emplace_back(generate_and_insert, num_threads, num_numbers_pe
```

```
    for (auto& thread : threads)
        thread.join();

    print_list();

    if (check_for_duplicates())
        cout << "No duplicates found in the list." << endl;
    else
        cout << "Duplicates found in the list." << endl;

    return 0;
}
```