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» (ИУ)
КАФЕДРА «Информационная безопасность» (ИУ8)

Лабораторная работа № 8

ПО КУРСУ

«Алгоритмические языки»

на тему «Синхронизация потоков»

Студент ИУ8-25 (Группа) В.В.Гоза (И. О. Фамилия)

Преподаватель: В. В. Соборова (И.О. Фамилия)

2022 г.

Условие:

Для своего варианта лабораторной работы № 7 обеспечить синхронизацию потоков:

1. Обеспечить печать имени потока и значения в одну строку без возможных разрывов, продемонстрировать два варианта реализации: использование mutex и использование блокировки.

2. С помощью условной переменной обеспечить, чтобы главный поток дожидался завершения дочерних потоков (дочерние потоки перед завершением оповещают главный поток, главный поток принимает эти оповещения). Главный поток после приема оповещения от каждого дочернего потока печатает об этом событии сообщение.

Программа:

Часть 1:

```
#include <iostream>
#include <string>
#include <ctime>
#include <cstdlib>
#include <future>
#include <thread>
#include <mutex>

std::mutex mut;


double arr_fill(double *numbers, int size) {
    srand(time(NULL));
    for (int i = 0; i < size; i++) {
        int ran = rand() % 1000;
        numbers[i] = ran + 1;
        std::cout << numbers[i] << " ";
    }
    std::cout << std::endl;
    return *numbers;
}

void sorter(double *numbers, int starting, int ending, const std::string
name) {
    int temp = 0;
    std::lock_guard<std::mutex>lock(mut);
    for (int i = starting; i < ending - 1; i++) {
        int min = i;
        for (int j = i + 1; j < ending; j++) {
            if (numbers[j] < numbers[min]) {
                min = j;
```

```
            }
        }
        temp = numbers[i];
        numbers[i] = numbers[min];
        numbers[min] = temp;
    }
    for (size_t i = starting; i < ending; ++i) {
        std::cout << name << " ";
        std::this_thread::sleep_for(std::chrono::milliseconds(10));
        std::cout << numbers[i] << std::endl;
    }
}

int main() {
    int size = 12;
    double *arr_async = new double[size];
    double *arr_thread = new double[size];
    std::cout << "Array for async" << std::endl;
    arr_fill(arr_async, size);
    std::cout << std::endl;
    std::future<void> async_1 = std::async(sorter, arr_async, 0, 6,
"async_1");
    std::future<void> async_2 = std::async(sorter, arr_async, 5, 12,
"async_2");
    async_1.get();
    async_2.get();
    std::future<void> async_3 = std::async(sorter, arr_async, 0, 12,
"async_3");
    async_3.get();
    std::cout << "Array for async after sort" << std::endl;
    for (int i = 0; i < size; i++) {
        std::cout << arr_async[i] << " ";
}
    std::cout << std::endl << std::endl;
    std::cout << "Array for thread" << std::endl;
    arr_fill(arr_thread, size);
    std::cout << std::endl;
        std::thread thread_1(sorter, arr_thread, 0, 6, "thread_1");
        std::thread thread_2(sorter, arr_thread, 5, 12, "thread_2");
        thread_1.join();
        thread_2.join();
        std::thread thread_3(sorter, arr_thread, 0, 12, "thread_3");
        thread_3.join();
    std::cout << "Array for thread after sort" << std::endl;
    for (int i = 0; i < size; i++) {
        std::cout << arr_thread[i] << " ";
    }
    std::cout << std::endl;
    return 0;
}
```

Вывод программы:

```
Array for async
214 267 373 683 36 543 886 720 369 136 784 97

async_1 36
async_1 214
async_1 267
async_1 373
async_1 543
async_1 683
async_2 97
async_2 136
async_2 369
async_2 683
async_2 720
async_2 784
async_2 886
async_3 36
async_3 97
async_3 136
async_3 214
async_3 267
async_3 369
async_3 373
async_3 543
async_3 683
async_3 720
async_3 784
async_3 886
Array for async after sort
36 97 136 214 267 369 373 543 683 720 784 886
```

```
Array for thread
499 77 892 951 206 999 131 575 370 350 24 89

thread_1 77
thread_1 206
thread_1 499
thread_1 892
thread_1 951
thread_1 999
thread_2 24
thread_2 89
thread_2 131
thread_2 350
thread_2 370
thread_2 575
thread_2 999
thread_3 24
thread_3 77
thread_3 89
thread_3 131
thread_3 206
thread_3 350
thread_3 370
thread_3 499
thread_3 575
thread_3 892
thread_3 951
thread_3 999
Array for thread after sort
24 77 89 131 206 350 370 499 575 892 951 999
```

Часть 2:

```cpp
#include <iostream>
#include <string>
#include <ctime>
#include <cstdlib>
#include <future>
#include <thread>
#include <mutex>
#include <condition_variable>

std::mutex mut;
std::condition_variable reporter;
bool finished_1st = false;
bool finished_2nd = false;
bool finished_3rd = false;

double arr_fill(double *nums, int size) {
    srand(time(NULL));
    for (int i = 0; i < size; i++) {
        int ran = rand() % 1000;
        nums[i] = ran + 1;
        std::cout << nums[i] << " ";
    }
    std::cout << std::endl;
    return *nums;
}

void sorter(double *nums, int start_point, int end_point, const std::string
name) {
    int temp = 0;
    for (int i = start_point; i < end_point - 1; i++) {
        int min = i;
        for (int j = i + 1; j < end_point; j++) {
            if (nums[j] < nums[min]) {
                min = j;
            }
        }
        temp = nums[i];
        nums[i] = nums[min];
        nums[min] = temp;
    }
    for (size_t i = start_point; i < end_point; ++i) {
        std::cout << name << " ";
        std::this_thread::sleep_for(std::chrono::milliseconds(10));
        std::cout << nums[i] << std::endl;
    }
    if (name == "threads_1" or name == "asyncs_1") {
        finished_1st = true;
        reporter.notify_one();
    }
    if (name == "threads_2" or name == "asyncs_2") {
        finished_2nd = true;
        reporter.notify_one();
    }
    if (name == "threads_3" or name == "asyncs_3") {
        finished_3rd = true;
        reporter.notify_one();
    }
}

int main() {
    int size = 12;
```

```
    double *arr_async = new double[size];
    double *arr_thread = new double[size];
    std::cout << "Array for async" << std::endl;
    arr_fill(arr_async, size);
    std::cout << std::endl;
    {
        std::unique_lock<std::mutex> lock(mut);
        std::future<void> async_1 = std::async(sorter, arr_async, 0, 6,
"asyncs_1");
        while (!finished_1st) {
            reporter.wait(lock);
        }
        finished_1st = false;
        std::cout << "\nFirst async ready\n" << std::endl;

        std::future<void> async_2 = std::async(sorter, arr_async, 6, 12,
"asyncs_2");
        while (!finished_2nd) {
            reporter.wait(lock);
        }
        finished_2nd = false;
        std::cout << "\nSecond async ready\n" << std::endl;

        std::future<void> async_3 = std::async(sorter, arr_async, 0, 12,
"asyncs_3");
        while (!finished_3rd) {
            reporter.wait(lock);
        }
        finished_3rd = false;
        std::cout << "\nThird async ready\n" << std::endl;

        std::cout << "Array for async after sort" << std::endl;
        for (int i = 0; i < size; i++) {
            std::cout << arr_async[i] << " ";
        }
        std::cout << std::endl << std::endl;
    }
    std::cout << "Array for thread" << std::endl;
    arr_fill(arr_thread, size);
    std::cout << std::endl;
    {
        std::unique_lock<std::mutex> lock(mut);
        std::thread thread_1(sorter, arr_thread, 0, 6, "threads_1");
        while (!finished_1st) {
            reporter.wait(lock);
        }
        thread_1.join();
        std::cout << "\nFirst thread ready\n" << std::endl;

        std::thread thread_2(sorter, arr_thread, 6, 12, "threads_2");
        while (!finished_2nd) {
            reporter.wait(lock);
        }
        thread_2.join();
        std::cout << "\nSecond thread ready\n" << std::endl;

        std::thread thread_3(sorter, arr_thread, 0, 12, "threads_3");
        while (!finished_3rd) {
            reporter.wait(lock);
        }
        thread_3.join();
        std::cout << "\nThird thread ready\n" << std::endl;
        std::cout << "Array for thread after sort" << std::endl;
        for (int i = 0; i < size; i++) {
```

```
            std::cout << arr_thread[i] << " ";
        }
        std::cout << std::endl;
    }
}
```

Вывод программы:

```
Array for async
190 95 680 219 26 701 835 198 118 442 55 461

asyncs_1 26
asyncs_1 95
asyncs_1 190
asyncs_1 219
asyncs_1 680
asyncs_1 701

First async ready

asyncs_2 55
asyncs_2 118
asyncs_2 198
asyncs_2 442
asyncs_2 461
asyncs_2 835

Second async ready

asyncs_3 26
asyncs_3 55
asyncs_3 95
asyncs_3 118
asyncs_3 190
asyncs_3 198
asyncs_3 219
asyncs_3 442
asyncs_3 461
asyncs_3 680
asyncs_3 701
asyncs_3 835

Third async ready

Array for async after sort
26 55 95 118 190 198 219 442 461 680 701 835
```

```
Array for thread
190 95 680 219 26 701 835 198 118 442 55 461

threads_1 26
threads_1 95
threads_1 190
threads_1 219
threads_1 680
threads_1 701

First thread ready

threads_2 55
threads_2 118
threads_2 198
threads_2 442
threads_2 461
threads_2 835

Second thread ready

threads_3 26
threads_3 55
threads_3 95
threads_3 118
threads_3 190
threads_3 198
threads_3 219
threads_3 442
threads_3 461
threads_3 680
threads_3 701
threads_3 835

Third thread ready

Array for thread after sort
26 55 95 118 190 198 219 442 461 680 701 835
```