**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 № 7

# ПО КУРСУ

# «Алгоритмические языки»

# на тему «Изучение потоковой многозадачности »

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

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

2022 г.

## Условие:

Вариант 3

Реализовать программу, в которой кроме главного создается три отдельных потока: первый поток сортирует первую половину вещественного массива, второй поток сортирует вторую половину вещественного массива, третий поток запускается после завершение первых двух, он сортирует массив полностью (уже частично отсортированный). Каждый поток имеет свое имя (например, thread1, thread2, thread3), и печатает отсортированный массив, перед печатью каждого значения элемента массива поток должен напечатать с новой строки свое имя. После завершения дочерних потоков главный поток выдает сообщение об окончании работы. Имена потоков и массив передаются в потоковую функцию через ее параметры, использовать одну потоковую функцию для всех трех потоков. Массив предварительно до запуска потоков заполняются числами с использованием ГПСЧ. Для сортировки использовать метод прямого выбора.

1. Запустить программу несколько раз при одних и тех же исходных данных, посмотреть, как меняются результаты вывода. Сделать выводы.

---

2. Между печатью имени потока и значением установить небольшую задержку, например, 10 мс. Посмотреть, как меняются результаты вывода. Сделать выводы.

## Программа:

```
//Вариант 3
#include <iostream>
#include <ctime>
#include <cstdlib>
#include <string>
#include <future>
#include <thread>
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 *numbers, int starting, int ending, const std::string name) {
      int temp = 0;
      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:

### В первый раз без задержки:

Array for async:

638 360 274 876 891 263 739 567 761 391 792 343

async_1 263

async_1 274

async_1 360

async_1 638

async_1 876

async_1 891

async_2 343

async_2 391

async_2 567

async_2 739

async_2 761

async_2 792

async_2 891

async_3 263

async_3 274

async_3 343

async_3 360

async_3 391

async_3 567

async_3 638

async_3 739

async_3 761

async_3 792

async_3 876

async_3 891

Array for async after sort:

263 274 343 360 391 567 638 739 761 792 876 891

Array for thread:

638 360 274 876 891 263 739 567 761 391 792 343

thread_1 263

thread_1 274

thread_1 360

thread_1 638

thread_1 876

thread_1 thread_2 891

343

thread_2 391

thread_2 567

thread_2 739

thread_2 761

thread_2 792

thread_2 891

thread_3 263

thread_3 274

thread_3 343

thread_3 360

thread_3 391

thread_3 567

thread_3 638

thread_3 739

thread_3 761

thread_3 792

thread_3 876

thread_3 891

Array for thread after sort:

263 274 343 360 391 567 638 739 761 792 876 891

## Во второй раз без задержки:

Array for async:

315 710 418 819 411 905 618 743 613 763 698 709

async_1 315

async_1 411

async_1 418

async_1 710

async_1 819

async_1 905

async_2 613

async_2 618

async_2 698

async_2 709

async_2 743

async_2 763

async_2 905

async_3 315

async_3 411

async_3 418

async_3 613

async_3 618

async_3 698

async_3 709

async_3 710

async_3 743

async_3 763

async_3 819

async_3 905

Array for async after sort:

315 411 418 613 618 698 709 710 743 763 819 905

Array for thread:

315 710 418 819 411 905 618 743 613 763 698 709

thread_1 315

thread_1 411

thread_1 418

thread_1 710

thread_1 819

thread_1 905

thread_2 613

thread_2 618

thread_2 698

thread_2 709

thread_2 743

thread_2 763

thread_2 905

thread_3 315

thread_3 411

thread_3 418

thread_3 613

thread_3 618

thread_3 698

thread_3 709

thread_3 710

thread_3 743

thread_3 763

thread_3 819

thread_3 905

Array for thread after sort:

315 411 418 613 618 698 709 710 743 763 819 905

## В третий раз с задержкой:

Array for async:

599 823 715 316 852 793 641 543 855 843 667 181

async_1 async_2 181

async_2 316

async_1 543

async_2 599

async_1 641

async_2 715

async_1 667

async_2 793

async_1 843

async_2 823

async_1 852

async_2 181

855

async_3 181

async_3 316

async_3 543

async_3 599

async_3 641

async_3 667

async_3 715

async_3 793

async_3 823

async_3 843

async_3 852

async_3 855

Array for async after sort:

181 316 543 599 641 667 715 793 823 843 852 855

Array for thread:

884 633 234 583 22 601 886 398 856 706 907 173

thread_1 thread_2 22

173

thread_1 thread_2 234

thread_1 398

thread_2 583

thread_1 706

thread_2 856

thread_2 601

thread_1 884

633

thread_2 thread_1 886

173

thread_2 907

thread_3 22

thread_3 173

thread_3 234

thread_3 398

thread_3 583

thread_3 601

thread_3 633

thread_3 706

thread_3 856

thread_3 884

thread_3 886

thread_3 907

Array for thread after sort:

22 173 234 398 583 601 633 706 856 884 886 907