Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №3 по курсу**

**«Операционные системы»**

**Тема работы**

**“Потоки”**

Студент: Зубко Дмитрий Валерьевич

Группа: М8О-208Б-20

Вариант: 17

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

**Репозиторий**

<https://github.com/usernameMAI/OS/tree/main/os_lab3>

**Постановка задачи**

## Цель работы

Целью является приобретение практических навыков в:

* Управление потоками в ОС
* Обеспечение синхронизации между потоками

## Задание

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме.

Вариант 17. Найти в большом целочисленном массиве минимальный элемент

**Общие сведения о программе**

Программа написана на языке С++. Для компиляции требуется указать ключ –pthread.

**Общий метод и алгоритм решения**

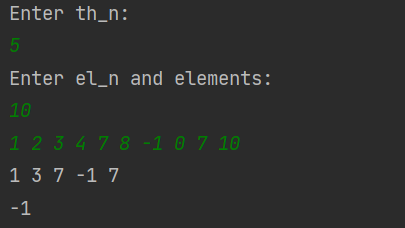
При запуске программы требуется указать количество потоков *th\_n* и количество элементов *el\_n* в массиве, где будет происходить поиск минимального элемента. Этот массив разбивается на частей. В каждой части разные потоки находят минимальный элемент и заносят его в отдельный массив. В конце программы просматривается этот отдельный массив и выводится минимальный элемент. Также выводятся минимальные элементы в каждой из частей исходного массива чисел.

Создаём поток с помощью thread(функция, параметры функции).

**Исходный код**

#include <iostream>  
#include <thread>  
#include <vector>  
  
using namespace std;  
  
void min\_elem(vector<int> const& elements, vector<int>& min, const int l, const int r, const int i) {  
 int min\_el = elements[l];  
  
 for (int j = l; j <= r; ++j)  
 {  
 if (elements[j] < min\_el)  
 min\_el = elements[j];  
 }  
// cout << this\_thread::get\_id() << endl;  
 min[i] = min\_el;  
}  
  
int main() {  
 int th\_n, el\_n;  
 cout << "Enter th\_n: \n";  
 cin >> th\_n;  
 cout << "Enter el\_n and elements: \n";  
 cin >> el\_n;  
  
 vector<int> elements(el\_n);  
 vector<int> min(th\_n);  
 vector<thread> threads(th\_n);  
  
 for (int i = 0; i < el\_n; ++i)  
 cin >> elements[i];  
  
 int l = 0;  
 int delta = el\_n / th\_n;  
 int last\_id = th\_n - 1;  
  
 for (int i = 0; i < th\_n; ++i)  
 {  
 if (i != last\_id)  
 {  
 threads[i] = thread(min\_elem, ref(elements), ref(min), l, l + delta - 1, i);  
 l += delta;  
 } else  
 {  
 threads[i] = thread(min\_elem, ref(elements), ref(min), l, el\_n - 1, i);  
 }  
 }  
  
 for (int i = 0; i < th\_n; ++i)  
 {  
 threads[i].join();  
 }  
  
 for (int i = 0; i < th\_n; ++i)  
 cout << min[i] << ' ';  
  
 cout << endl;  
  
 int ans = min[0];  
 for (int i = 1; i < th\_n; ++i)  
 if (min[i] < ans)  
 ans = min[i];  
  
 cout << ans << endl;  
}

**Демонстрация работы программы**



**Выводы**

Лабораторная работа ознакомила и научила меня работать с потоками. Я изучил некоторые функции из библиотеки thread. Понял, что при выполнении сложных задач можно распараллеливать их для увеличения производительности.