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: Ивченко Анна Владимировна

Группа: М8О-208Б-20

Вариант:8

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/Anetta123/OS/tree/main/os\_lab3/src

**Постановка задачи**

Задача: Есть К массивов одинаковой длины. Необходимо сложить эти массивы. Необходимо предусмотреть стратегию, адаптирующуюся под количество массивов и их длину (по количеству операций).

**Общие сведения о программе**

Для реализации поставленной задачи нам нужны следующие библиотеки:  
<iostream> -  [заголовочный файл](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BE%D1%87%D0%BD%D1%8B%D0%B9_%D1%84%D0%B0%D0%B9%D0%BB) с классами, функциями и переменными для организации ввода-вывода в языке программирования [C++](https://ru.wikipedia.org/wiki/C%2B%2B).

<chrono> - для функций, работающих со временем.

<thread> - для работы с потоками.

<vector> - для работы с динамическими массивами - векторами.

<string> - для работы со строками.

<sstream> - заголовочный файл с классами, функциями и переменными для организации работы со строками через интерфейс потоков.

Для работы с потоками используем библиотеку thread в C++. Я создаю вектор потоков и заполняю его ими по необходимости. В моей реализации потоки работают с регулярными фунцкиями, а именно с фунцкией sum, как раз и производящей нужные вычисления. С помощью библиотеки chrono я замеряю время выполнения нужных вычислений для сравнения между собой запуска программы с разным количеством потоков.

**Общий метод и алгоритм решения**

В программе есть такие переменные: k – количество массивов и n – их длина. После программе подается на вход количество потоков. Объем работы между потоками разделяем поровну в зависимости от их количества. Таким образом каждый поток производит суммирования всех массивов только на диапазоне с определенными для него индексами. Каждый поток запускается с регулярной функцией sum, которая производит суммирования всех массивов на определенном диапазоне индексов.

**Исходный код**

#include <iostream>

#include <thread>

#include <vector>

#include <string>

#include <sstream>

#include <chrono>

using namespace std;

void Sum(int left, int right, const std::vector<std::vector<int>>& v, std::vector<int>& ans) {

for (int i = left; i < right; ++i) {

for (int j = 0; j < v.size(); ++j) {

ans[i] += v[j][i];

}

}

}

int main(){

int k = 10000; // Number of Arrays

int n = 10000; // Length of Arrays

std::vector<std::vector<int>> v(k,std::vector<int>(n));

std::vector<int> ans(v[0].size());

for (size\_t i = 0; i < k; ++i) {

for (size\_t j = 0; j < n; ++j) {

v[i][j] = 1;

}

}

std::cout << "Enter number of threads: "<<std::endl;

int threads\_num;

std::cin >> threads\_num;

if (threads\_num > n) {

threads\_num = n;

}

if (threads\_num > 8) {

threads\_num = 8;

}

auto begin = std::chrono::steady\_clock::now(); // Initial moment of time

if (threads\_num == 0) {

for (size\_t i = 0; i < n; ++i) {

for (size\_t j = 0; j < k; ++j) {

ans[i] += v[j][i];

}

}

} else {

std::vector<std::thread> th;

int left = 0;

int delta = n / threads\_num;

int right = delta;

th.reserve(threads\_num);

for (int i = 0; i < threads\_num; ++i) {

th.emplace\_back(Sum, left, right,std::ref(v), std::ref(ans));

left = right;

right = right + delta;

if (i == threads\_num - 2) {

right = n;

}

}

for (int i = 0; i < threads\_num; ++i) {

th[i].join();

}

}

auto end = std::chrono::steady\_clock::now(); // End moment of time

for (size\_t i = 0; i < n; ++i) {

std::cout << ans[i] << " ";

}

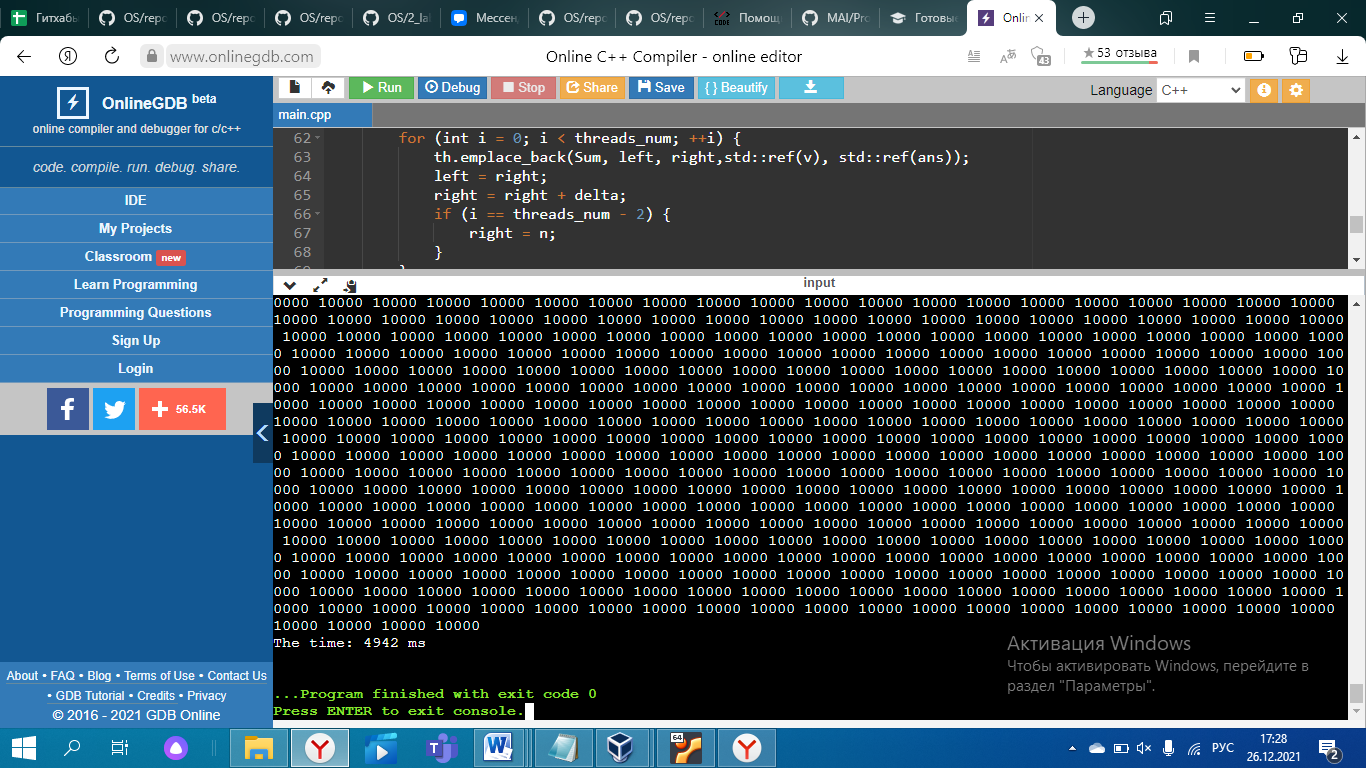
auto elapsed\_ms = std::chrono::duration\_cast<std::chrono::milliseconds>(end - begin);

std::cout << "\nThe time: " << elapsed\_ms.count() << " ms\n";

}

**Демонстрация работы программы**

Тест 1:



**Выводы**

В ходе проделанной работе, были получены знания и уменя в работе с потоками Linux. Были усвоены многие тонкости работы с потоками.