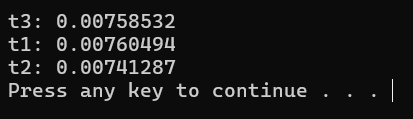
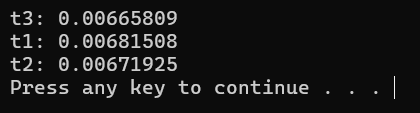
Лабораторная работа №2  
Соколов Арсений

**Тест программы (см Приложение 1)**

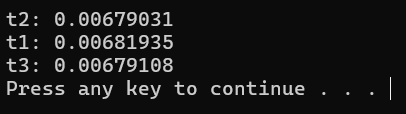
Первый запуск в *Debug*:



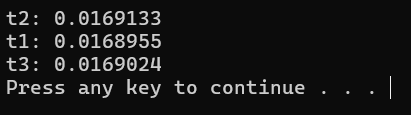
Второй запуск в *Debug*:



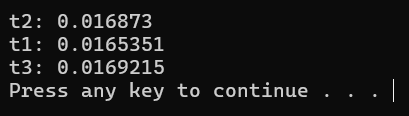
Третий запуск в *Debug*:



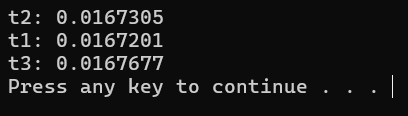
Первый запуск в *Release*:



Второй запуск в *Release*:



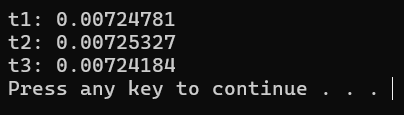
Третий запуск в *Release*:



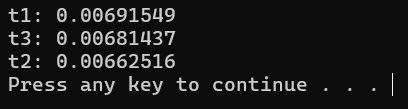
**Объяснение:** порядок создания потоков в коде не гарантирует порядок выполнения потоков, *поэтому* *в выводе потоки идут не по порядку их создания*. Операционная система может запустить их в любом порядке. Разные потоки могут завершить разное количество итераций в цикле за секунду из-за различий в загрузке процессора, планировании потоков операционной системой и других факторов, *поэтому результаты вычислений в каждом потоке отличаются*.

**Тест программы без *std::mutex m* (см Приложение 2)**

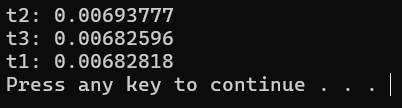
Первый запуск в *Debug*:



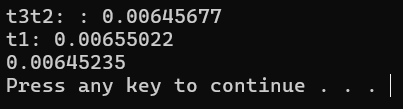
Второй запуск в *Debug*:



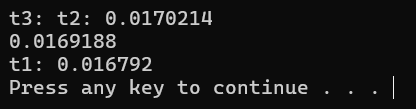
Третий запуск в *Debug*:



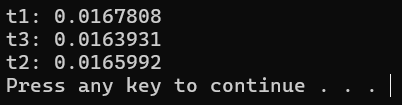
Четвертый запуск в *Debug*:



Первый запуск в *Release*:



Второй запуск в *Release*:



**Объяснение:** при удалении мьютекса из кода, возникает ситуация, известная как "гонка данных" (race condition) – несколько потоков пытаются одновременно получить доступ и изменить общие данные без синхронизации доступа. В нашем случае, несколько потоков могут одновременно попытаться выполнить операцию вывода в консоль. Поскольку операция вывода в консоль не является атомарной, то различные потоки могут перекрываться в выводе, *поэтому иногда получается “страшный” вывод*.

**ПРИЛОЖЕНИЕ 1**

***Программа «What’s going on?»***

#include <thread>

#include <mutex>

#include <string>

#include <iostream>

#include <cmath>

#include <chrono>

std::mutex m;

void Func(std::string name)

{

long double i = 0;

auto start = std::chrono::high\_resolution\_clock::now();

while (std::chrono::high\_resolution\_clock::now() - start < std::chrono::seconds(1)) {

i += pow(10, -9);

}

m.lock();

std::cout << name << ": " << i << std::endl;

m.unlock();

}

int main()

{

std::cout << std::endl;

std::thread thread1(Func, "t1");

std::thread thread2(Func, "t2");

std::thread thread3(Func, "t3");

thread1.join();

thread2.join();

thread3.join();

system("pause");

return 0;

}

**ПРИЛОЖЕНИЕ 2**

***Программа «What’s going on?» без* *std::mutex m***

#include <thread>

#include <mutex>

#include <string>

#include <iostream>

#include <cmath>

#include <chrono>

void Func(std::string name)

{

long double i = 0;

auto start = std::chrono::high\_resolution\_clock::now();

while (std::chrono::high\_resolution\_clock::now() - start < std::chrono::seconds(1)) {

i += pow(10, -9);

}

std::cout << name << ": " << i << std::endl;

}

int main()

{

std::cout << std::endl;

std::thread thread1(Func, "t1");

std::thread thread2(Func, "t2");

std::thread thread3(Func, "t3");

thread1.join();

thread2.join();

thread3.join();

system("pause");

return 0;

}