МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

КАФЕДРА АВТОМАТИЗИРОВАННЫХ СИСТЕМ УПРАВЛЕНИЯ

**ОТЧЕТ ПО**

**ЛАБОРАТОРНОЙ РАБОТЕ №2**

**Дисциплина «Параллельное программирование»**

Факультет: **АВТФ**

Группа**: АВТ-812**

Студенты:                                                                   Преподаватель:

**Беннер Богдан                                                          Ландовский В.В.**

**Пунченко Евгений**

**Раченков Максим**

Новосибирск

2020 год

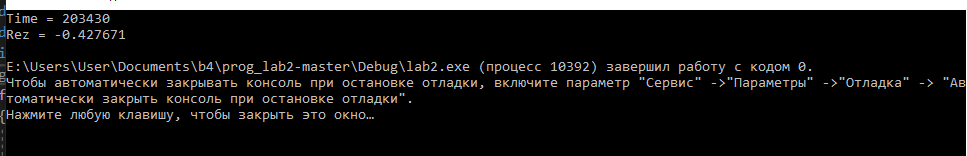
Разрыв страницы

**Цель работы**: реализовать программу, подсчитывающую интеграл методом левых прямоугольников, используя потоки, с использованием библиотеки Open Mp.

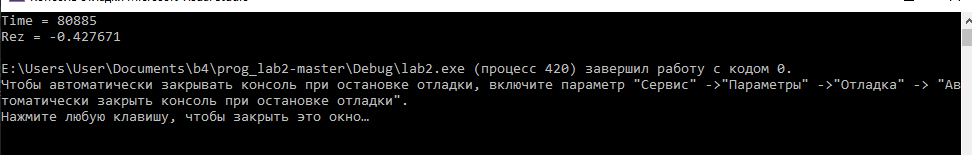
**Ход работы**

Для знакомства с поточным программированием, мы решили использовать элементарную функцию y = cos(x) в интервале 0..6000.

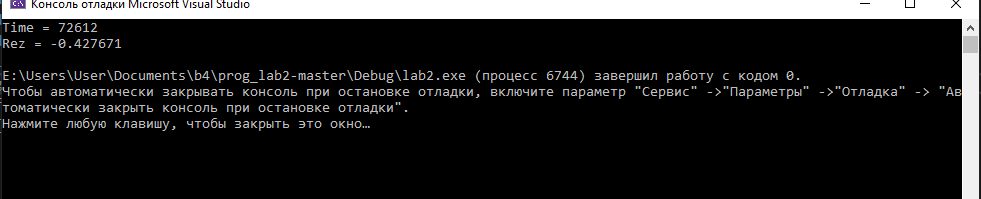
Результат работы программы с одним потоком:



Результат работы программы с тремя потоками:



Результат работы программы с шестью потоками:



Исходя из вышеперечисленного , можно сделать вывод  :  при достаточно трудной задаче, чем больше потоков , тем быстрее выполняется программа . Но при малой сложности задачи программа с потоками будет работать медленнее.

Код программы :

#include <iostream>

#include <ctime>

#include <chrono>

struct MyData

{

double a;

double b;

double e;

double (\*function)(double x);

double sum;

};

MyData generateData(double a, double b, double e, double (\*function)(double x), double sum)

{

MyData myData;

myData.a = a;

myData.b = b;

myData.e = e;

myData.function = function;

myData.sum = sum;

return myData;

}

double calculate(void\* args)

{

auto time1 = std::chrono::high\_resolution\_clock::now();

MyData myData = \*(MyData\*)args;

double a = myData.a;

double b = myData.b;

double e = myData.e;

double h = e;

double x = a;

double sum = 0;

int finish = (b - a) / h;

#pragma omp parallel for num\_threads(6) reduction(+:sum)

for(int i=0;i< finish;i++)

{

sum += h \* myData.function(i\*e);

}

auto time2 = std::chrono::high\_resolution\_clock::now();

auto duration = std::chrono::duration\_cast<std::chrono::microseconds>(time2 - time1).count();

auto avrDur = duration;

std::cout << "Time = " << avrDur << std::endl;

return sum;

}

double cosinus(double x)

{

return cos(x);

}

int main()

{

srand(time(NULL));

const int N = 1;

double sum = 0;

double sums[N];

MyData data[N];

double a = 0, b = 6000;

data[0] = generateData(a, b, 0.001, cos, 0);

double rez=calculate(&data[0]);

std::cout <<"Rez = "<< rez << std::endl;

return 0;

};