Минобрнауки России  
Федеральное государственное автономное образовательное  
Учреждение высшего образования  
«Санкт-Петербургский государственный электротехнический  
Университет им. В.И. Ульянова (Ленина)»  
(СПГЭТУ «ЛЭТИ»)  
Факультет компьютерных технологий и информатики

Кафедра вычислительной техники  
  
  
  
**Отчет по лабораторной работе №3**

**Тема:«** **Процессы и потоки»**

по дисциплине «Операционные системы»

Выполнил студент гр.9307 Брызгалова Е.А.

Принял: к.т.н., доцент Тимофеев А.В.

Санкт-Петербург  
2021 г.

Оглавление

[Цель: 3](#_Toc89490133)

[Исследовать механизмы создания и управленияпроцессами и потоками в ОС Windows. 3](#_Toc89490134)

[Задание 1.1. 3](#_Toc89490135)

[Указания к выполнению. 3](#_Toc89490136)

[Протокол работы программы 3](#_Toc89490137)

[График зависимости времени от числа потоков. 4](#_Toc89490138)

[Выводы по заданию 1 4](#_Toc89490139)

[Задание 1.2 4](#_Toc89490140)

[Указания к выполнению 4](#_Toc89490141)

[Протокол работы программы 5](#_Toc89490142)

[График зависимости времени от числа потоков. 5](#_Toc89490143)

[Выводы по заданию 2 5](#_Toc89490144)

[Приложение 5](#_Toc89490145)

[Исходный код 5](#_Toc89490146)

# Цель:

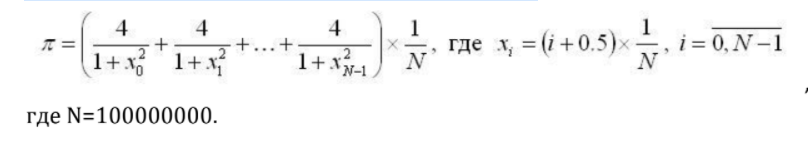
# Исследовать механизмы создания и управленияпроцессами и потоками в ОС Windows.

# Задание 1.1.

## Указания к выполнению.

Создайте приложение, которое вычисляет число pi с точностью

N знаков после запятой по следующей формуле:



## Протокол работы программы

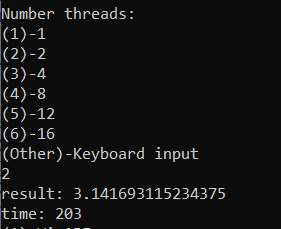


Рисунок 1. Пример запуска

## График зависимости времени от числа потоков.

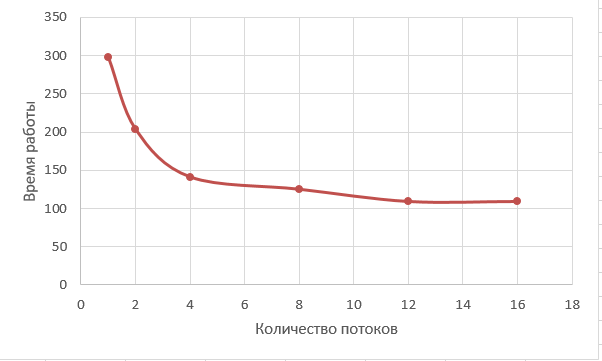


Рисунок 2. График

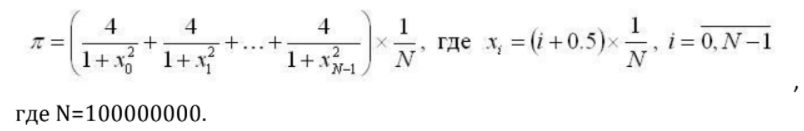
## Выводы по заданию 1

В ходе выполнения первого задания лабораторной работы было реализовано многопоточное приложение с использованием функций Win32 API. Из графика видно, что увеличение числа потоков приводит к уменьшению времени работы программы.

# Задание 1.2

## Указания к выполнению

Реализация многопоточного приложения с использованием технологии OpenMP. Создайте приложение, которое вычисляет число пи с точностью N знаков после запятой по следующей формуле:



## Протокол работы программы

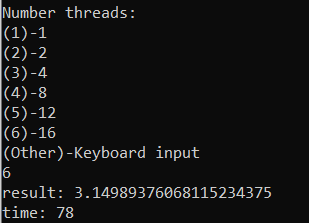


Рисунок 3. Протокол работы программы

## График зависимости времени от числа потоков.

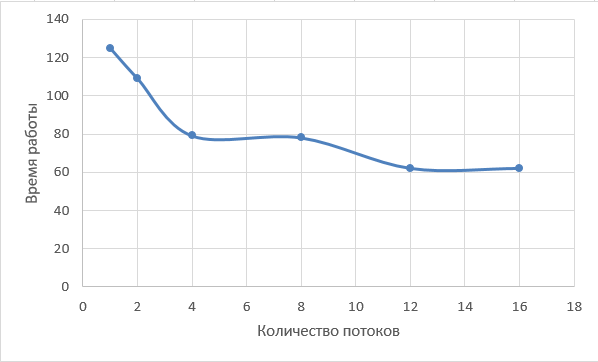


Рисунок 4. График

## Выводы по заданию 2

В ходе выполнения второго задания лабораторной работы было реализовано многопоточное приложение с использованием технологии OpenMP. Использование технологии OpenMP уменьшает время работы программы и упрощает написание кода.

# Приложение

## Исходный код

#pragma region libs

#include <iostream>

#include <iomanip>

#include <windows.h>

#include <string>

#include <omp.h>

using namespace std;

#pragma endregion

const long N = 10000000;

const long block = 40;

DWORD tlsIndex;

CRITICAL\_SECTION CriticalSection;

float sum;

DWORD numoftr;

HANDLE\* threads;

int Threads()

{

int c,t;

cout << "\nNumber threads:\n(1)-1\n(2)-2\n(3)-4\n(4)-8\n(5)-12\n(6)-16\n(Other)-Keyboard input\n";

cin >> c;

switch (c)

{

case 1:

t = 1;

break;

case 2:

t = 2;

break;

case 3:

t = 4;

break;

case 4:

t = 8;

break;

case 5:

t = 12;

break;

case 6:

t = 16;

break;

default:

cin >> t;

break;

}

return t;

}

DWORD WINAPI CalculatingFunc(LPVOID lpParameter)

{

int i = 0;

long curpr = \*(reinterpret\_cast<long\*>(lpParameter)), pr = curpr;

float pi ;

TlsSetValue(tlsIndex, (LPVOID)&pi);//Сохранение значения в слоте локального хранилища (TLS) потока вызывающего потока для указанного индекса TLS. У каждого потока процесса есть свой слот для каждого индекса TLS.

do

{

i++;

pi = 0;

for (long j = pr; j < pr + block && j < N; ++j)

pi += ((4.0 / (1.0 + (j + 0.5) / N \* (j + 0.5) / N))/ N);

TlsSetValue(tlsIndex, (LPVOID)&pi);

EnterCriticalSection(&CriticalSection);//Потоки одного процесса могут использовать объект критической секции для синхронизации взаимного исключения.

{

sum += pi;

pr = numoftr \* block \* i + curpr;

}

LeaveCriticalSection(&CriticalSection);

} while (pr < N);

return 0;

}

void first()

{

int t1,t2;

numoftr = Threads();

threads = new HANDLE[numoftr];

DWORD\* lpThreadId = new DWORD[numoftr];

tlsIndex = TlsAlloc();

InitializeCriticalSection(&CriticalSection);

for (int i = 0; i < numoftr; ++i)//создание потоков

threads[i] = CreateThread(nullptr,//Указатель на структуру SECURITY\_ATTRIBUTES, которая определяет, может ли возвращаемый дескриптор быть унаследован дочерними процессами. Если lpThreadAttributes имеет значение NULL, дескриптор не может быть унаследован.

0,//Начальный размер стека в байтах. Система округляет это значение до ближайшей страницы. Если этот параметр равен нулю, новый поток использует размер исполняемого файла по умолчанию.

CalculatingFunc,//Указатель на функцию, определяемую приложением, которая должна выполняться потоком. Этот указатель представляет начальный адрес потока.

new long{ i \* block },//Указатель на переменную, которую нужно передать потоку.

CREATE\_SUSPENDED,//Поток создается в приостановленном состоянии и не запускается, пока не будет вызвана функция ResumeThread .

lpThreadId + i);//Указатель на переменную, которая получает идентификатор потока. Если этот параметр равен NULL , идентификатор потока не возвращается.

t1 = GetTickCount();

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

ResumeThread(threads[i]);

WaitForMultipleObjects//Ожидает, пока один или все указанные объекты не перейдут в сигнальное состояние или пока не истечет интервал времени ожидания.

(numoftr,

threads,

TRUE, //функция возвращается, когда сигнализируется состояние всех объектов в массиве еркуфвы

INFINITE);// функция вернется только тогда, когда будут указаны указанные объекты

t2 = GetTickCount();

cout.precision(20);

cout << "result: "<< sum << endl;

cout << "time: " << t2 - t1 << endl;

DeleteCriticalSection(&CriticalSection);

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

CloseHandle(threads[i]);

free(threads);

TlsFree(tlsIndex);

}

float second()

{

int t1, t2;

numoftr = Threads();

float pi = 0;

t1 = GetTickCount();

#pragma omp parallel /\*Определяет параллельную область, которая является кодом, который будет выполняться несколькими потоками параллельно.\*/for /\*Приводит к тому, что работа, выполненная в for цикле внутри параллельной области, делится между потоками.\*/ reduction(+: pi)//

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

pi += ((4.0 / (1.0 + (i + 0.5) / N \* (i + 0.5) / N))/ N);

t2 = GetTickCount();

cout.precision(20);

cout << "result: " << pi << endl;

cout << "time: " << t2 - t1 << endl;

return pi;

}

int main()

{

int c;

do{

sum = 0;

cout << "(1)-WinAPI" << "\n(2)-OpenMp" << "\n(3)-Exit\n";

cin >> c;

switch (c)

{

case 1:

first();

break;

case 2:

second();

break;

case 3:

return 0;

default:

cout << "Wrong enterence";

}

}while (true);

}