МIНIСТЕРСТВО ОСВIТИ I НАУКИ УКРАЇНИ

НАЦIОНАЛЬНИЙ ТЕХНIЧНИЙ УНIВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛIТЕХНIЧНИЙ IНСТИТУТ»

Кафедра прикладної математики

ПОВIДОМЛЕННЯ ТА ЧЕРГИ ПОВIДОМЛЕНЬ В ОС WINDOWS

лабораторна робота №2 з дисциплiни:

«Операцiйнi системи»

|  |  |
| --- | --- |
| Виконав: | Керівник: |
| студент групи КМ-63 | *асистент Громова В. В.* |
| *Артеменко Я.К.* |  |

Київ — 2019

# **ЗМІСТ**

[**Вступ** 2](#_Toc3146962)

[**1 ПОСТАНОВКА ЗАДАЧІ** 3](#_Toc3146963)

[**2** **ОПИС ПРОГРАМИ** 4](#_Toc3146964)

[**ВИСНОВКИ** 5](#_Toc3146965)

[**Додаток А (скріншоти результату)** 6](#_Toc3146966)

[**Додаток Б (текст програми)** 6](#_Toc3146967)

# **Вступ**

Потік (thread) — це базова одиниця в рамках процесу, якiй ОС видiляє процесорний час. Потiк може виконувати будь-яку частину коду процесу, у тому числi частини коду, якi в даний момент виконує інший потiк. Усi потоки процесу мають спiльний вiртуальний адресовий простiр та системнi ресурси. Окрiм того, кожний процес пiдтримує опрацьовувачi виключних ситуацiй (exception handlers), планувальний пріоритет (scheduling priority), локальну пам’ять (local storage), унiкальний iдентифiкатор, а також низку структур, якi ОС використовує для збереження контексту потоку пiд час його простоювання. Контекст потоку (thread context) включає значення регiстрiв процесора, системний стек (kernel stack), блок середовища потоку, а також користувацький стек (user stack) в адресовому просторi процесу, якому належить потiк. Потоки також можуть мати власний безпековий контекст.

# **1 ПОСТАНОВКА ЗАДАЧІ**

У рамках виконання лабораторної роботи потрiбно:

а) ознайомитися з теоретичними вiдомостями, викладеними в роздiлi 2;

б) написати будь-якою мовою програмування програму, яка повинна:

1) за потреби створювати додатковi потоки, окрiм головного;

2) регулювати доступ потокiв до деякого ресурсу (файлу, графiчного об’єкта, статичної змiнної i т.п.); зокрема, у програмi повинно бути передбачено можливiсть як синхронного, так i асинхронного доступу потокiв до ресурсу (на вимогу користувача);

3) регулювати прiоритети потокiв на вимогу користувача;

4) за потреби зупиняти та вiдновлювати потоки;

5) використовувати тiльки засоби Windows API i не повинна використовувати стандартнi процедури MFC, спецiальнi класи для реалiзацiї багатопотокових аплiкацiй тощо;

в) вiдлагодити програму в ОС Windows версiї XP та вище;

г) пiдготувати звiт лабораторної роботи вiдповiдно до вимог роздiлу 3.

Типова програма, яку можна написати в рамках даної лабораторної роботи, може передбачати наявнiсть одного iнтерфейсного потоку, який здiйснює спiлкування з користувачем та керування iншими потоками, а також декiлькох робочих потокiв, якi виконують фоновi задачi (перерахунок даних, друк тексту, уведення/виведення, пошук файлiв тощо).

*В рамках даної лабораторної роботи* було написано програму, яка імітує складні обрахунки якихось величин, для яких потрібно багато часу, в одному потоці, коли виконується інший потік (головний).

# **ОПИС ПРОГРАМИ**

В межах даної лабораторної роботи було розроблено програму, яка імітує складний процес під час виконання іншого.

*Загальні відомості про програму:*

Програма була написана мовою C з використання режиму багато поточності за допомогою бібліотеки thread.

Всього було використано три потоки(один головний і інший робочий), кожен з яких виконує окрему задачу:

1. Головний потік забезпечує відкриття головного вікна та виконання певних дій;
2. Другий потік забезпечує імітацію виконання складних обчислень та виводить їх результат.

Функції, які використовує програма:

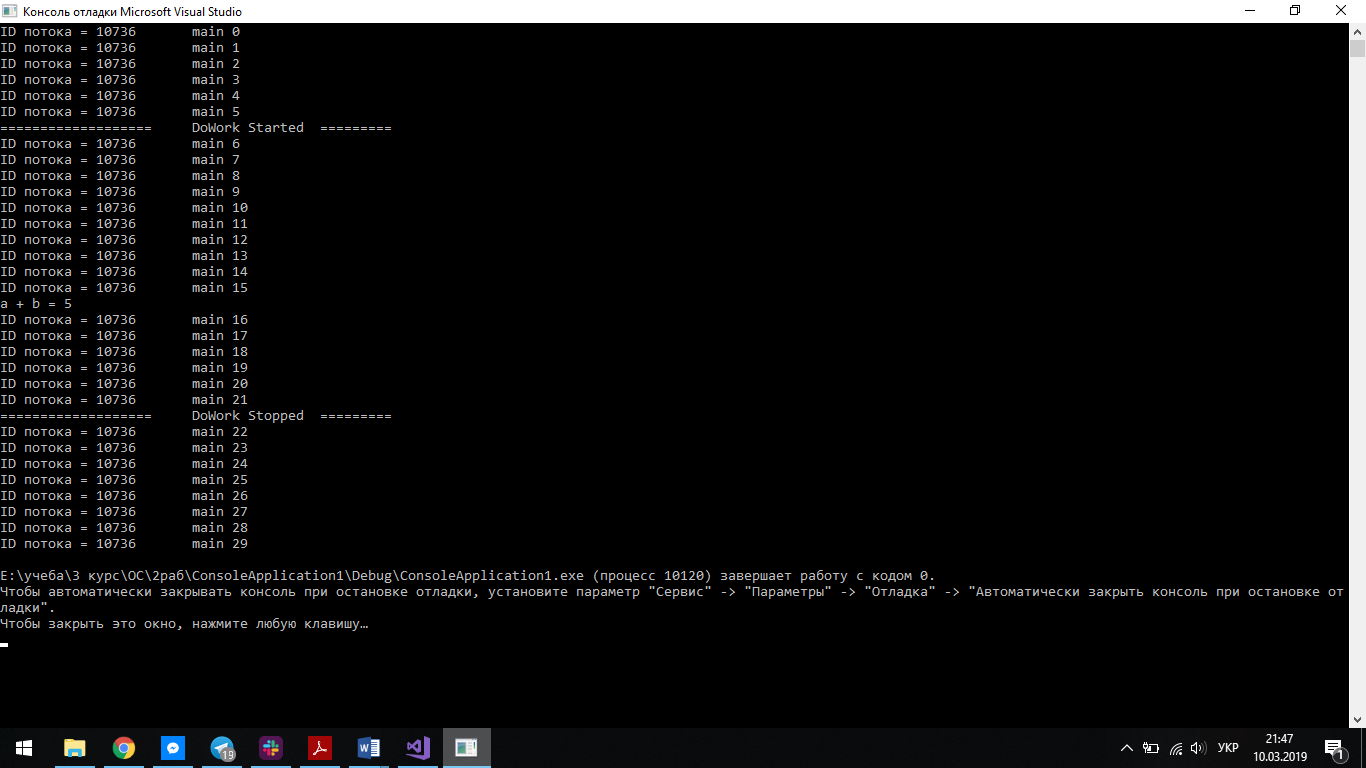
1. **main –** є головною функцією, яка є окремим потоком і слугує для виводу ID – потоку та виклику функції dowork;
2. **dowork** – функція, яка імітує складні обчислення і є другим потоком.

# **ВИСНОВКИ**

На цій лабораторній роботі було здобуто навички по використанню потоків, його базові принципи та його переваги. Також було створено програму, яка демонструє роботу потоків.

# 

# **Додаток А (скріншоти результату)**



# **Додаток Б (текст програми)**

#include "pch.h"

#include <iostream>

#include <thread>

#include <chrono>

using namespace std;

void dowork(int a, int b)

{

this\_thread::sleep\_for(chrono::milliseconds(3000));

cout << "===================\t" << "DoWork Started\t=========" << endl;

this\_thread::sleep\_for(chrono::milliseconds(5000));

cout << "a + b = " << a + b << endl;

this\_thread::sleep\_for(chrono::milliseconds(3000));

cout<< "===================\t" << "DoWork Stopped\t=========" << endl;

}

int main()

{

setlocale(LC\_ALL, "ru");

thread th(dowork,2,3);

for (size\_t i = 0; i<30; i++)

{

cout << "ID потока = " << this\_thread::get\_id() << "\tmain " << i << endl;

this\_thread::sleep\_for(chrono::milliseconds(500));

}

th.join();

return 0;

}