МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут комп'ютерних наук та інформаційних технологій
Кафедра програмного забезпечення

ЗВІТ

до лабораторної роботи №3
на тему: «Створення та керування процесами засобами API в операційній системі WINDOWS.»
з дисципліни: «Операційні системи»

Лектор:
ст. викладач кафедри ПЗ
Грицай О. Д.

Виконав:
ст. гр. ПЗ-22
Чаус О. М.

Прийняла:
ст. викладач кафедри ПЗ
Грицай О. Д.

« ____ » _____ 2022 р.

∑= _____ .....:

Львів-2022

Тема роботи: Створення та керування процесами засобами API в операційній системі WINDOWS.

Мета роботи: Ознайомитися з багатопоточністю в ОС Windows. Навчитися працювати з процесами, використовуючи WinAPI-функції.

Теоретичні відомості

Створення Win32 процесу здійснюється викликом однієї з функцій: CreateProcess, CreateProcessAsUser (для Win NT/2000) і CreateProcessWithLogonW (починаючи з Win2000).

Завершення процесів у Win32 API

Завершення процесів виконує функція ExitProcess(): VOID ExitProcess (UINT exitcode);
exitcode - код повернення процесу. Наприклад:
ExitProcess (100); // вихід з кодом 100
Завершення іншого процесу виконує функція TerminateProcess().

Визначення класу пріоритету

DWORD GetPriorityClass(HANDLE hProcess);
Функція повертає клас пріоритету або 0.

Встановити пріоритет процесу

BOOL SetPriorityClass(HANDLE hProcess, DWORD dwPriorityClass);
Повертає не нуль в разі успіху і нуль коли помилка
Призупинити інший потік можна за допомогою функції
DWORD SuspendThread(HANDLE hThread);
Отримати інформацію про час виконання процесу можна через функцію GetProcessTimes.

Завдання для виконання лабораторної роботи

1. Створити окремий процес, і здійснити в ньому розв'язок задачі згідно варіанту у відповідності до порядкового номера у журнальному списку (підгрупи).
2. Реалізувати розв'язок задачі у 2-ох, 4-ох, 8-ох процесах. Виміряти час роботи процесів за допомогою функцій WinAPI. Порівняти результати роботи в одному і в багатьох процесах.
3. Для кожного процесу реалізувати можливість його запуску, зупинення, завершення та примусове завершення («вбиття»).
4. Реалізувати можливість зміни пріоритету виконання процесу.
5. Продемонструвати результати виконання роботи, а також кількість створених процесів у "Диспетчері задач", або подібних утилітах (н-д, ProcessExplorer)

   Табулювати функцію arctg x, задану розкладом в ряд Тейлора, в області її визначення на відрізку від A до B (кількість кроків не менше 100 000 –задається користувачем).

Хід роботи

1. Здійснив розв'язок індивідуальної задачі та створив окремий процес, який його виконав(зображення 1-2).
Код розв'язку індивідуальної задачі:

```
#include <iostream>
#include <iomanip>
#include <cmath>
#include <vector>
#include <chrono>

#define EPS 0.0001
```

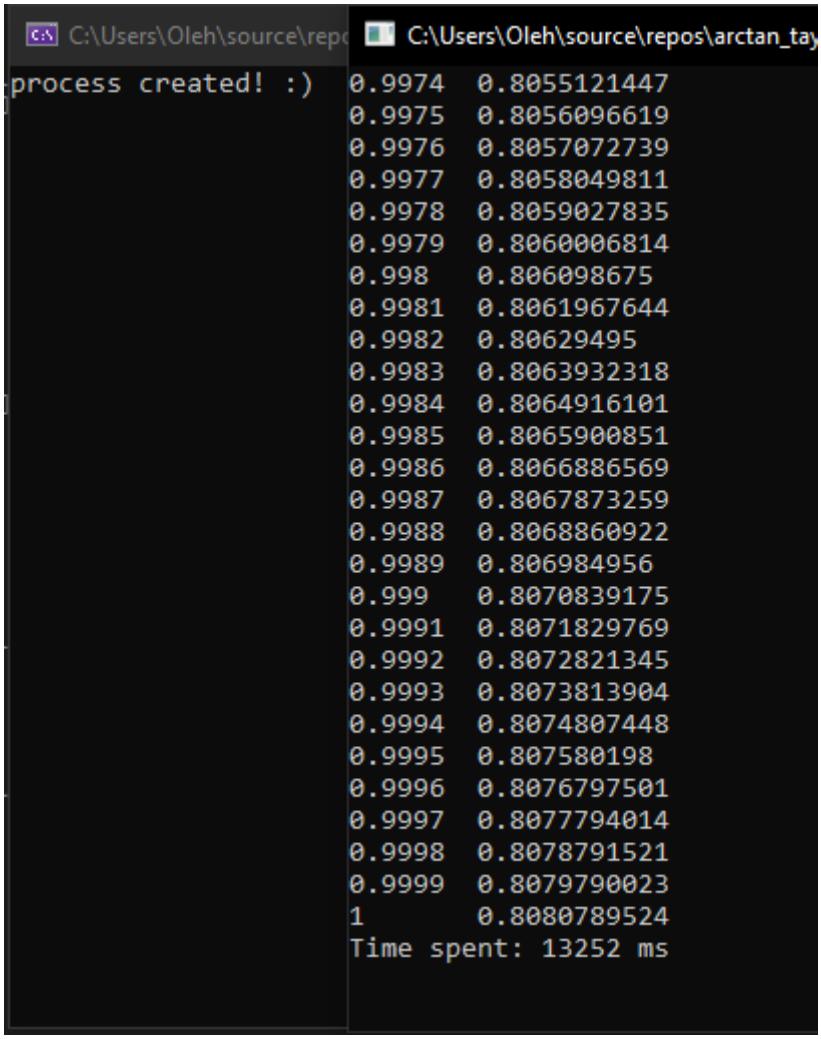
```
struct Arctan {
	double x;
	double arctanx;
};

int main(int argc, const char** argv) {
	auto start_time = std::chrono::system_clock::now();
	std::cout << std::setprecision(10);
	double A = atof(argv[1]);
	double B = atof(argv[2]);
	int steps = atoi(argv[3]);
	std::vector<Arctan> info(steps + 1);
	double step = (B - A) / steps;
	for (std::size_t i = 0; i < info.size(); i++) {
		info[i].x = A + (step * i);
		double sum = 0;
		double iter_value = EPS + 1;
		int k = 0;
		while (abs(iter_value) > EPS) {
			iter_value = (pow(-1, k) * pow(info[i].x, 1 + 2 * k)) / (1 + 2 * k);
			k++;
			sum += iter_value;
			if (k > 10) break;
		}
		info[i].arctanx = sum;
		std::cout << info[i].x << "\t" << info[i].arctanx << "\t" << "\n";
	}
	auto exit_time = std::chrono::system_clock::now();
	std::cout << "Time spent: " <<
std::chrono::duration_cast<std::chrono::milliseconds>(exit_time - start_time).count() << "
ms\n";
	putchar('\n');
	getchar();
	}
```

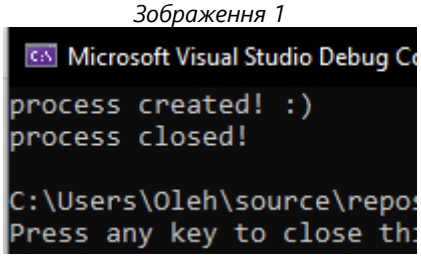
Код програми, що створює процес:

```
#include <iostream>
#include <Windows.h>
#include <chrono>
int main() {
	STARTUPINFO si;
	PROCESS_INFORMATION pi;
	ZeroMemory(&si, sizeof(STARTUPINFO));
	ZeroMemory(&pi, sizeof(PROCESS_INFORMATION));
	si.cb = sizeof(STARTUPINFO);
	std::wstring command_line =
L"C:\\Users\\Oleh\\source\\repos\\arctan_taylor\\Debug\\arctan_taylor.exe 0 1 10000";
	if (CreateProcess(NULL, (LPWSTR)(command_line.c_str()), NULL, NULL, NULL,
		CREATE_NEW_CONSOLE | HIGH_PRIORITY_CLASS, NULL, NULL, &si, &pi)) {
		std::cout << "process created! :)\n";
		WaitForSingleObject(pi.hThread, INFINITE);
		CloseHandle(pi.hProcess);
		CloseHandle(pi.hThread);
		std::cout << "process closed!\n";
	}
	else std::cout << "failed to create process ;(";

	return 0;
}
```

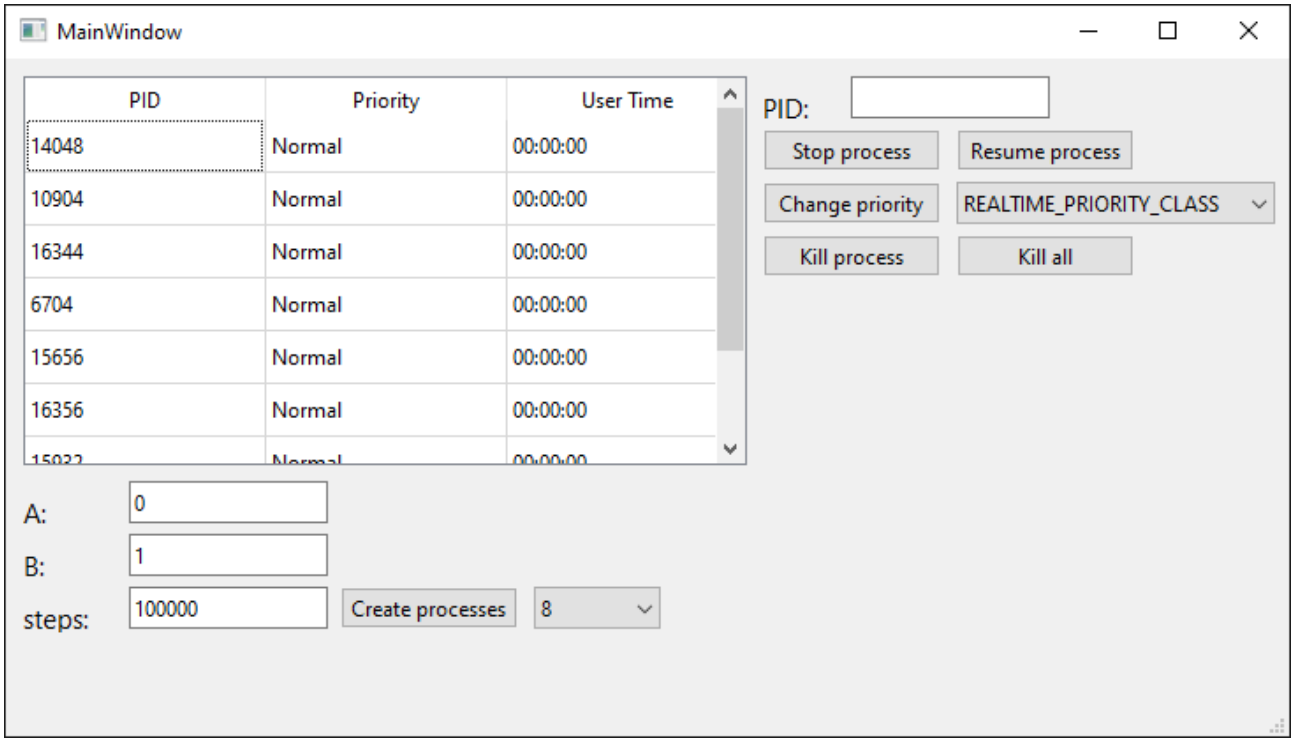


Зображення 1



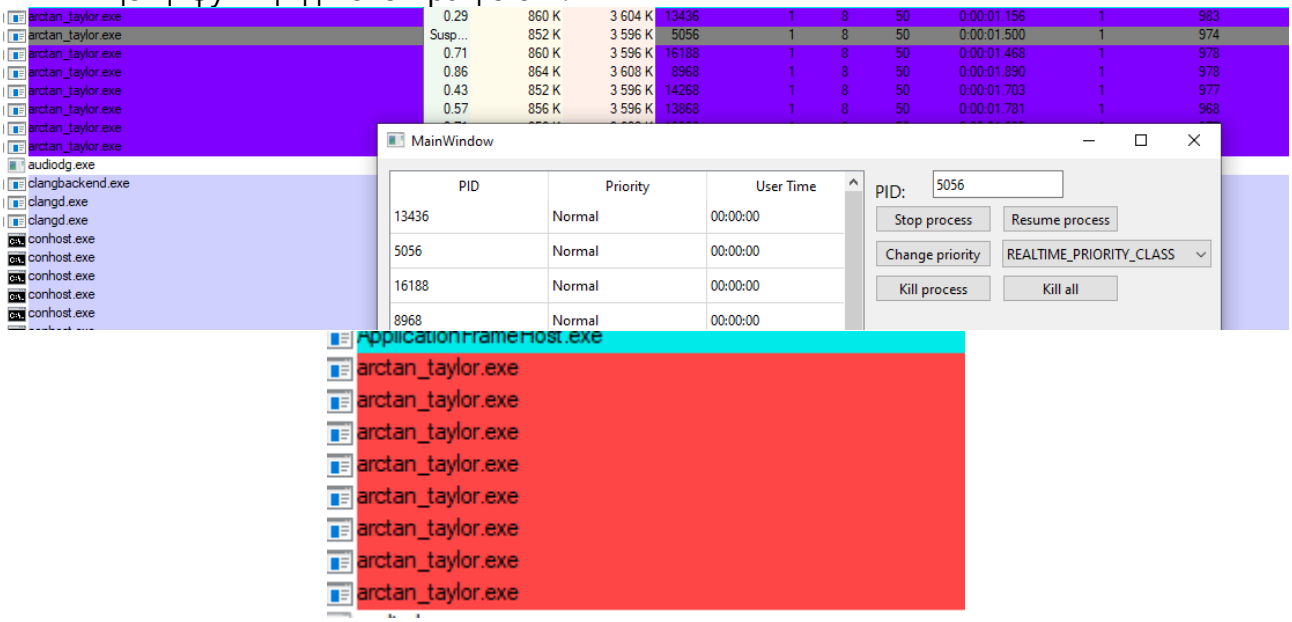
Зображення 2

2. Реалізував розв’язок задачі у 2-ох, 4-ох, 8-ох процесах. Виміряв час роботи процесів за допомогою функцій WinAPI.

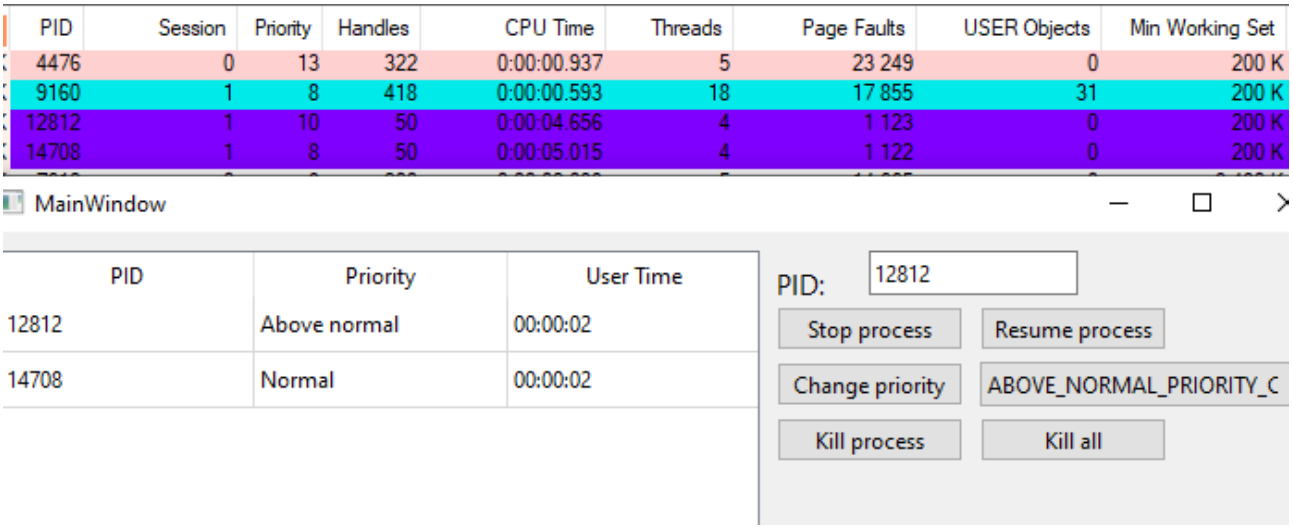


Інтерфейс програми, час виконання процесу

3. Для кожного процесу реалізував можливість його запуску, зупинення, та завершення. На зображеннях за допомогою утиліти Process Explorer показав, що ці функції дійсно працюють.



4. Реалізував можливість зміни пріоритету виконання процесу.



Висновок: під час виконання лабораторної роботи ознайомився з WinAPI, створенням процесів, функціями керування, такі як зміна пріоритету, зупинка, запуск та завершення. Отримав досвід розпаралелення програми на декілька процесів та написав невелику утиліту для демонстрації.