**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ**
**“ЛЬВІВСЬКА ПОЛІТЕХНІКА”**

**Кафедра систем штучного інтелекту**

**Лабораторна робота № 3**

з дисципліни

**«Операційні системи»**

по темі :

**“ Виконання задачі в декількох потоках в ОС Windows”**

Виконав:

студент групи КН-209

Ярчак А.В

Викладач:

Кривенчук Ю. П.

**Львів – 2019р.**

**Мета.** Навчитись реалізовувати розпаралелювання алгоритмів за допомогою багато-поточності в ОС Windows з використанням функцій WinAPI.

**Завдання.**

1. Реалізувати заданий алгоритм в окремому потоці.
2. Виконати розпаралелювання заданого алгоритму на 2, 4, 8 потоків.
3. Реалізувати можливість зміни пріоритету певного потоку.
4. Результати виконання роботи відобразити у звіті.

**Індивідуальні завдання.**

6. Створити масив N елементів і відсортувати його елементи у порядку ***спадання*** за допомогою методу ***«вибірки»***. Елементи масиву згенерувати випадковим чином за допомогою вбудованих функцій.

**Код програми:**

```
#include <iostream>
#include <windows.h>
#include <chrono>
#include <conio.h>

using namespace std;
using namespace std::chrono;

typedef struct THREAD_PARAMS{
    int *ptr, start, end;
    THREAD_PARAMS(int arr[], int s, int e):ptr(arr),start(s),end(e){}
}THREAD_PARAMS;

int priority[7] = {THREAD_PRIORITY_TIME_CRITICAL, THREAD_PRIORITY_HIGHEST,
THREAD_PRIORITY_ABOVE_NORMAL,
                   THREAD_PRIORITY_NORMAL, THREAD_PRIORITY_BELOW_NORMAL,
THREAD_PRIORITY_LOWEST, THREAD_PRIORITY_IDLE};


DWORD WINAPI Thread(LPVOID lpParam)
{
    int *arr = ((THREAD_PARAMS *) lpParam)->ptr;
    int start = ((THREAD_PARAMS *) lpParam)->start;
    int end = ((THREAD_PARAMS *) lpParam)->end;
    int i, key, j;
    for (i = start; i < end; i++) {
        key = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] < key) {
            arr[j + 1] = arr[j];
            j = j - 1;
            Sleep(1);
        }
        arr[j + 1] = key;
    }
}

int main()
{
    cout << "=== An Array Top-Down Insert Sort ===" << endl;

    int size;
    cout << "Enter array size:";
    cin >> size;

    int *arr = new int[size];
```

```
    for (int i = 0; i < size; i++) arr[i] = rand() % 11000 - 1000;

    cout << "Sort by 1 thread. Please wait..." << endl;

    auto start = high_resolution_clock::now();
    HANDLE hThread = CreateThread(nullptr, 0, Thread, (PVOID)(new
THREAD_PARAMS(arr, 0, size - 1)), 0, nullptr);
        WaitForSingleObject(hThread, INFINITE);
    auto stop = high_resolution_clock::now();
    cout << "Excellent! Work time:" << double(duration_cast<microseconds>(stop-
start).count())/1000000 << endl;

    CloseHandle(hThread);

    for (int i = 0; i < size; i++) arr[i] = rand() % 9000 + 1000;

    int count;
    cout << "\nNow try to sorting by few threads" << endl << "Enter count of
threads: ";
    cin >> count;

    int step = size / count;

    cout << "Sort by " << count << " threads. Please wait..." << endl;

    HANDLE threads[count];
    for (int i = 0; i < count; i++)
        threads[i] = CreateThread(nullptr, 0, Thread, (PVOID)(new
THREAD_PARAMS(arr, i*step, i == count-1? size-1 : i*step+step-1)),
CREATE_SUSPENDED, nullptr);

    while(true)
    {
        int command;
        cout << "1 - Change thread priority" << endl << "2 - Run threads"<<
endl<< "->";
        cin >> command;
        switch(command)
        {
            case 1:
                while(_getch() != 's')
                {
                    cout << "Select thread number (1-" << count << "): ";
                    int number;
                    cin >> number;
                    cout << "Select priority(1-critical 2-highest 3-above 4-
normal 5-below 6-lowest 7-idle): ";
                    int set;
                    cin >> set;
                    SetThreadPriority(threads[number-1],priority[set - 1]);
                }
                break;
            case 2:
                start = high_resolution_clock::now();
                for(HANDLE i: threads) ResumeThread(i);
                goto link;
            default:break;
        }
    }
    link:
    WaitForMultipleObjects(DWORD(count),threads,true,INFINITE);
    stop = high_resolution_clock::now();
    cout << "Excellent! Work time: " << double(duration_cast<microseconds>(stop-
start).count())/1000000 << endl;
```

```
    for(HANDLE i: threads) CloseHandle(i);

    return 0;
}
```

**Результат:**

```
F:\Programs\CLion\Projects\OS\lab1\cmake-build-debug\lab1.exe
=== An Array Top-Down Insert Sort ===
Enter array size:300
300
Sort by 1 thread. Please wait...
Excellent! Work time:45.9382

Now try to sorting by few threads
Enter count of threads:3
 3
Sort by 3 threads. Please wait...
1 - Change thread priority
2 - Run threads
->1
1

Select thread number (1-3):1
 1
Select priority(1-critical 2-highest 3-above 4-normal 5-below 6-lowest 7-idle):1
 1

Select thread number (1-3):2
 2
Select priority(1-critical 2-highest 3-above 4-normal 5-below 6-lowest 7-idle):7
 7

Select thread number (1-3):3
 3
Select priority(1-critical 2-highest 3-above 4-normal 5-below 6-lowest 7-idle):3
 3
s
1 - Change thread priority
2 - Run threads
->
2
2
Excellent! Work time: 6.40988

Process finished with exit code 0
```

**Висновок:** на цій лабораторній роботі я навчився покращив навички, знання і вміння користуватись бібліотекою WinAPI, а іменно, навчився працювати з потоками, створювати один або більше, вимірювати час їх роботи та змінювати приорітет.