

Міністерство освіти і науки України  
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра ЕОМ



Звіт  
з лабораторної роботи № 1  
з дисципліни «Системне програмне забезпечення»  
на тему: «Керування процесами та потоками»

Виконав: ст. гр. КІ-302

Радевич-Винницький Я.А.

Перевірила: викладач

Ногаль М.В.

**Мета роботи:** Навчитися керувати процесами і потоками в середовищі операційної системи, розробляти програми керування процесами і потоками.

**Варіант:** 18.

Розробити дві програми. Перша обчислює суму непарних чисел від L до U. Обчислення суми оформити як функцію потоку. Друга програма запускає першу як новостворений процес. Обидві програми мають виводити інформацію про усі запущені процеси і потоки (дескриптор та ідентифікатор).

**Виконання завдання:**

1. Створено перший програмний проект, який реалізує обчислення суми непарних чисел в заданому діапазоні масиву. Обчислення оформлено у вигляді функції потоку.

Код файлу main.cpp:

*Лістинг 1*

```
#include <stdio.h>
#include <windows.h>

struct ThreadArgs {
    int* array;
    int array_size;
    int l;
    int u;
};

DWORD WINAPI sumOddsNumbers(LPVOID lpParam) {
    struct ThreadArgs* threadArgs = (struct ThreadArgs*)lpParam;
    int* array = threadArgs->array;
    int size = threadArgs->array_size;

    int l = threadArgs->l;
    int u = threadArgs->u;

    int sum = 0;

    for (int i = l; i <= u; i++) {
        if (array[i] % 2 != 0) {
            sum += array[i];
        }
    }
    return sum;
}

int main() {
    int n;
    int l;
    int u;
    printf("Enter the size of the array: ");
    scanf_s("%d", &n);

    printf("Enter L: ");
    scanf_s("%d", &l);

    printf("Enter U: ");
```

```

scanf_s("%d", &u);

int* dynamicArray = (int*)malloc(n * sizeof(int));
for (int i = 0; i < n; i++) {
    dynamicArray[i] = rand() % 50;
}

printf("\nOriginal array: \n");
for (int i = 0; i < n; i++) {
    printf("%d ", dynamicArray[i]);
}
printf("\n");

printf("\nArray from L to U: \n");
for (int i = l; i <= u; i++) {
    printf("%d ", dynamicArray[i]);
}
printf("\n");

struct ThreadArgs threadArgs;
threadArgs.array = dynamicArray;
threadArgs.array_size = n;
threadArgs.l = l;
threadArgs.u = u;

HANDLE hThread;
DWORD dwThreadId;

hThread = CreateThread(
    NULL,
    0,
    sumOddsNumbers,
    &threadArgs,
    0,
    &dwThreadId);

if (hThread == NULL) {
    fprintf(stderr, "Error creating thread (%lu).\n", GetLastError());
    return 1;
}

WaitForSingleObject(hThread, INFINITE);
DWORD dwExitCode;
GetExitCodeThread(hThread, &dwExitCode);

printf("\nSum of odd numbers from L to U: %lu\n", dwExitCode);

printf("\n");

printf("\nThread ID: %lu\n", dwThreadId);
printf("Thread Handle: %p\n\n", hThread);
printf("Main process ID: %lu\n", GetCurrentProcessId());
printf("Main thread ID: %lu\n", GetCurrentThreadId());
printf("Main process Handle: %p\n", GetCurrentProcess());
printf("Main thread Handle: %p\n", GetCurrentThread());

CloseHandle(hThread);
free(dynamicArray);

getchar(); getchar();

return 0;
}

```

```
Microsoft Visual Studio Debu X + -
Enter the size of the array: 15
Enter L: 3
Enter U: 9

Original array:
41 17 34 0 19 24 28 8 12 14 5 45 31 27 11

Array from L to U:
0 19 24 28 8 12 14

Sum of odd numbers from L to U: 19

Thread ID: 14728
Thread Handle: 00000000000000B4

Main process ID: 17612
Main thread ID: 18240
Main process Handle: FFFFFFFFFFFFFFFF
Main thread Handle: FFFFFFFFFFFFFFFE

C:\3 course 2 sem\SPZ\L1\spz-lab1\x64\Debug\spz-lab1.exe (process 17612) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

*Рис. 1 – Результат роботи програми*

2. Створено другий програмний проект, який запускає файл spz-lab1.exe.

Код файлу main.cpp:

*Лістинг 2*

```
#include <stdio.h>
#include <windows.h>

int main() {
    LPCWSTR lpApplicationName = L"C:\\3 course 2 sem\\SPZ\\L1\\spz-
lab1\\x64\\Debug\\spz-lab1.exe";
    PROCESS_INFORMATION pi;
    STARTUPINFO si;
    int status = 0;

    ZeroMemory(&si, sizeof(si));
    si.cb = sizeof(si);
    ZeroMemory(&pi, sizeof(pi));

    si.dwFlags = STARTF_USESHOWWINDOW;
    si.wShowWindow = SW_SHOWDEFAULT;

    if (!CreateProcess(
        lpApplicationName,
        NULL,
        NULL,
        NULL,
        FALSE,
        CREATE_NEW_CONSOLE,
        NULL,
        NULL,
        &si,
        &pi)) {
        fprintf(stderr, "CreateProcess failed (%lu).\n", GetLastError());
        return 1;
    }
}
```

```

    }

    printf("Child process started...\n");

    printf("\nChild process ID: %lu\n", pi.dwProcessId);
    printf("Child thread ID: %lu\n", pi.dwThreadId);
    printf("Child process Handle: %p\n", pi.hProcess);
    printf("Child thread Handle: %p\n", pi.hThread);

    printf("\nParent process ID: %lu\n", GetCurrentProcessId());
    printf("Parent thread ID: %lu\n", GetCurrentThreadId());
    printf("Parent process Handle: %p\n", GetCurrentProcess());
    printf("Parent thread Handle: %p\n", GetCurrentThread());

    WaitForSingleObject(pi.hProcess, INFINITE);

    GetExitCodeProcess(pi.hProcess, (LPDWORD)&status);
    printf("\nChild process finished with status %d\n", status);

    CloseHandle(pi.hProcess);
    CloseHandle(pi.hThread);

    return 0;
}

```

```

C:\3 course 2 sem\SPZ\L1\spz >
Child process started...

Child process ID: 20056
Child thread ID: 20732
Child process Handle: 0000000000000068
Child thread Handle: 0000000000000060

Parent process ID: 5688
Parent thread ID: 15024
Parent process Handle: FFFFFFFFFFFFFFFF
Parent thread Handle: FFFFFFFFFFFFFFFE

Enter the size of the array: 15
Enter L: 3
Enter U: 14

Original array:
41 17 34 0 19 24 28 8 12 14 5 45 31 27 11

Array from L to U:
0 19 24 28 8 12 14 5 45 31 27 11

Sum of odd numbers from L to U: 138

Thread ID: 19848
Thread Handle: 00000000000000A8

Main process ID: 20056
Main thread ID: 20732
Main process Handle: FFFFFFFFFFFFFFFF
Main thread Handle: FFFFFFFFFFFFFFFE

```

*Рис. 2 – Результат роботи програми*

**Висновок:** під час виконання лабораторної роботи було розроблено програми для роботи з процесами та потоками. Перша програма обчислює суму непарних чисел за вказаним діапазоном, а друга запускає першу як новостворений процес. Обидві програми виводять інформацію про усі запущені процеси і потоки.