Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ  «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра ЕОМ



Звіт

з лабораторної роботи № 2

з дисципліни «Системне програмне забезпечення»

на тему: «Організація взаємодії між процесами та потоками»

Виконав: ст. гр. КІ-302

Радевич-Винницький Я.А.

Перевірила: викладач

Ногаль М.В.

**Мета роботи:** Навчитися планувати процеси і потоки в середовищі операційної системи, розробляти програми планування процесів і потоків.**Варіант:** 18.

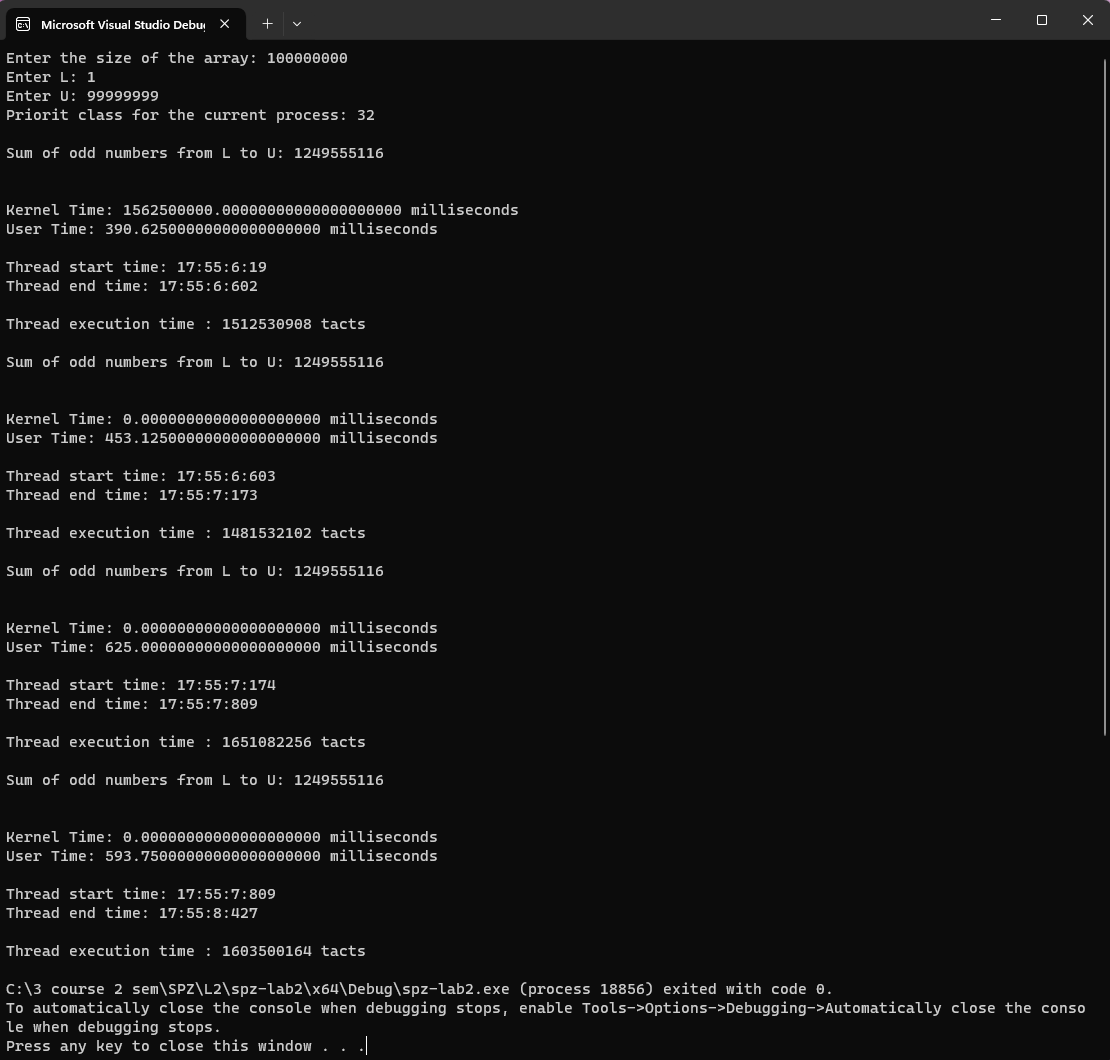
Розробити програму, яка обчислює суму непарних чисел від L до U. Обчислення суми оформити як функцію потоку. Запустити потік на виконання з декількома рівнями пріоритету, визначити час виконання потоку за допомогою функції GetThreadTimes(). Запустити програму декілька раз з різними вхідними даними, результати оформити у вигляді таблиці.**Виконання завдання:**

1. Створено перший програмний проект, який реалізує обчислення суми непарних чисел в заданопу діаназоні 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);  HANDLE hProcess = GetCurrentProcess();  int priorityClass = GetPriorityClass(hProcess);  printf("Priorit class for the current process: %d\n", priorityClass);  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;  for (int priority = THREAD\_PRIORITY\_IDLE; priority <= THREAD\_PRIORITY\_TIME\_CRITICAL;  priority += 10) {  hThread = CreateThread(  NULL,  0,  sumOddsNumbers,  &threadArgs,  0,  &dwThreadId);  if (hThread == NULL) {  fprintf(stderr, "Error creating thread (%lu).\n", GetLastError());  return 1;  }  SetThreadPriority(hThread, priority);  ResumeThread(hThread);  WaitForSingleObject(hThread, INFINITE);  DWORD dwExitCode;  GetExitCodeThread(hThread, &dwExitCode);  printf("\nSum of odd numbers from L to U: %lu\n", dwExitCode);  printf("\n");  FILETIME creationTime, exitTime, kernelTime, userTime;  if (GetThreadTimes(hThread, &creationTime, &exitTime, &kernelTime, &userTime)) {  ULARGE\_INTEGER kernelTimeInt, userTimeInt;  kernelTimeInt.LowPart = kernelTime.dwLowDateTime;  kernelTimeInt.HighPart = kernelTime.dwHighDateTime;  userTimeInt.LowPart = userTime.dwLowDateTime;  userTimeInt.HighPart = userTime.dwHighDateTime;  printf("\nKernel Time: %.20f milliseconds\n", kernelTimeInt.QuadPart \* 1e4);  printf("User Time: %.20f milliseconds\n", userTimeInt.QuadPart \* 1e-4);  SYSTEMTIME systemTime;  FileTimeToSystemTime(&creationTime, &systemTime);  printf("\nThread start time: %u:%u:%u:%u\n", systemTime.wHour,  systemTime.wMinute, systemTime.wSecond, systemTime.wMilliseconds);  FileTimeToSystemTime(&exitTime, &systemTime);  printf("Thread end time: %u:%u:%u:%u\n", systemTime.wHour,  systemTime.wMinute, systemTime.wSecond, systemTime.wMilliseconds);  ULONGLONG ThreadCycleTime;  QueryThreadCycleTime(hThread, &ThreadCycleTime);  printf("\nThread execution time : %llu tacts\n", ThreadCycleTime);  }  CloseHandle(hThread);  }  return 0;  } |



*Рис. 1 – Результат роботи програми при масиві розміром 100000000 елементів*

Результати виконання програми:

Таблиця 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Значення n, Пріоритет | -15 | -5 | 5 | 15 |
| 1000 | 0 мс. | 0 мс. | 0 мс. | 0 мс. |
| 100000 | 0 мс. | 0 мс. | 0 мс. | 0 мс. |
| 100000000 | 375 мс. | 500 мс. | 531 мс. | 640 мс. |

**Висновок:** під час виконання лабораторної роботи було розроблено програму для роботи з процесами та потоками. Програма обчислює суму непарних чисел за вказаним діапазоном. Було запущено потік на виконання з декількома рівнями пріоритету, визначено час виконання потоку за допомогою функції GetThreadTimes(). Запущено програму декілька раз з різними вхідними даними, результати оформлено у вигляді таблиці.