**ЛАБОРАТОРНА РОБОТА #1. СТВОРЕННЯ ПОТОКІВ. КЛАС THREAD. ПРІОРИТЕТНІ ТА ФОНОВІ ПОТОКИ. ПРІОРИТЕТИ ПОТОКІВ**

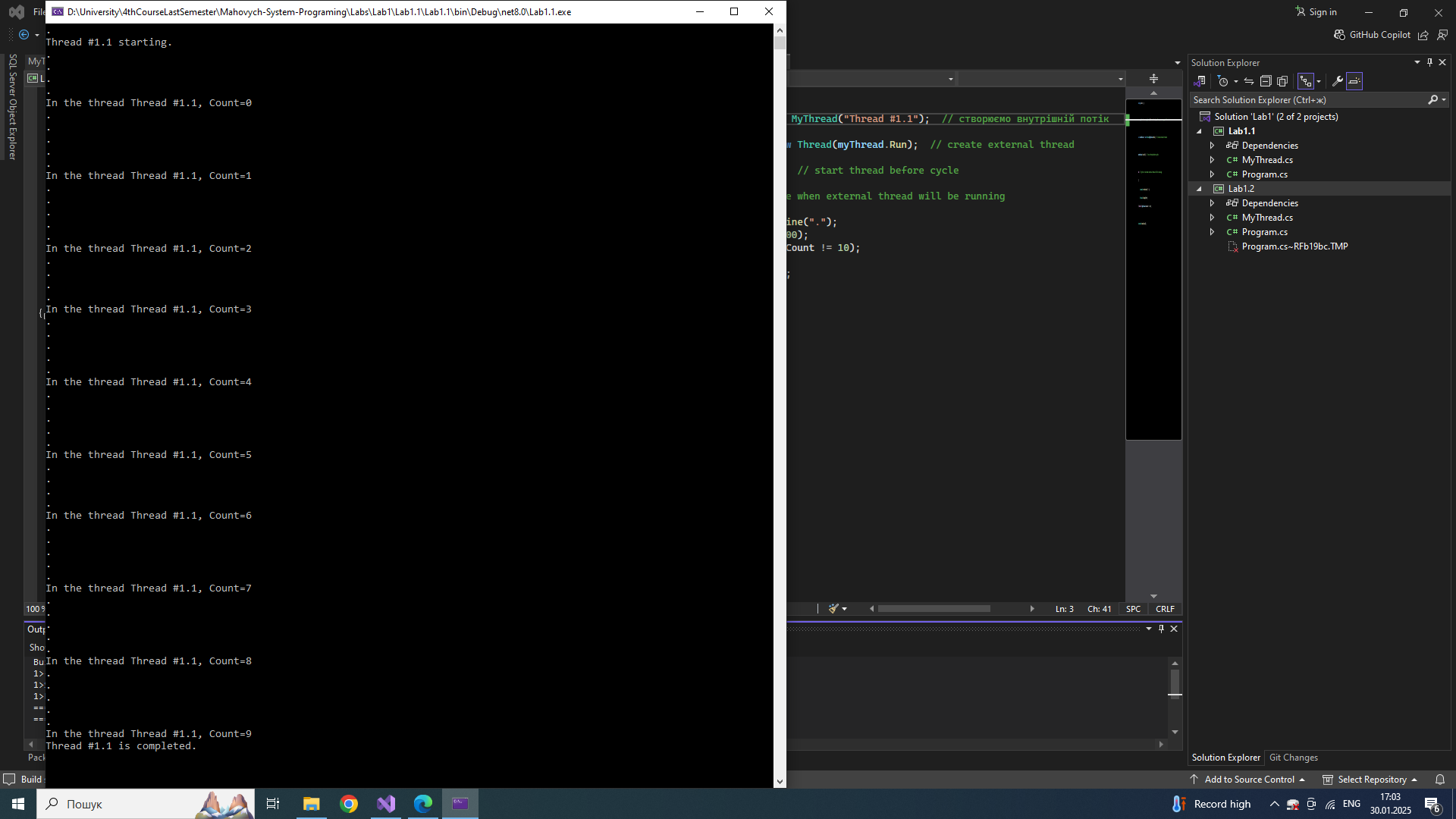
Іваницька Анна, МІТ 41, Варіант 1

**Завдання для виконання:**

1. Повторити задачі, розв’язані у тексті.
2. Створити багатопотокову програму, задати різні пріоритети для потоків. Підрахувати розподіл процесорного часу між потоками у %. *Варіант 1)*Потоків – 3 шт. (пріоритети: середній, вище середнього, найвищий).
3. Високий рівень: Реалізувати програму із попереднього завдання з використанням графічного інтерфейсу та можливістю динамічного задання кількості та пріоритетності потоків; відображення ходу обчислень та виведення результатів.
4. Оформити звіт.

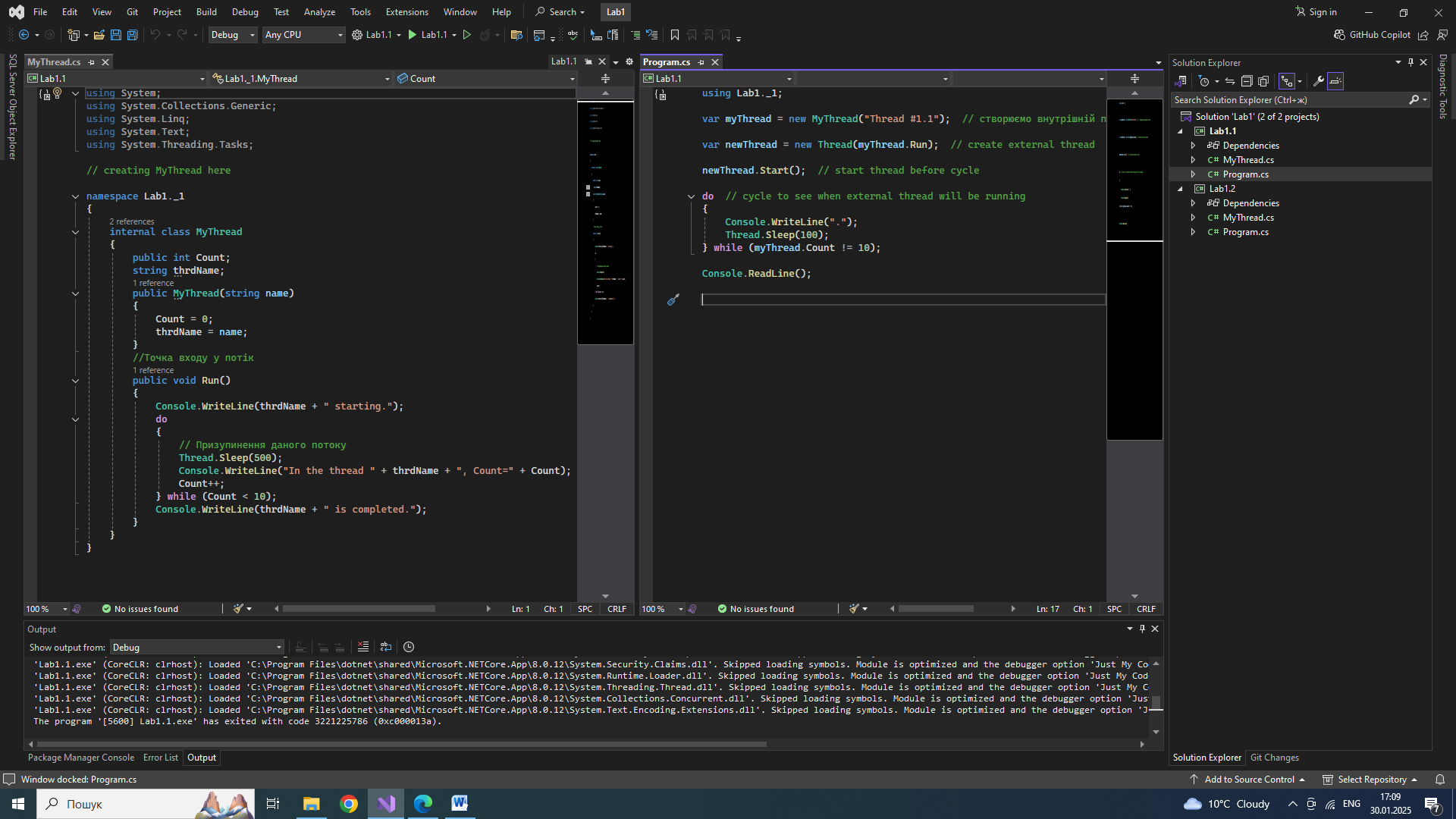
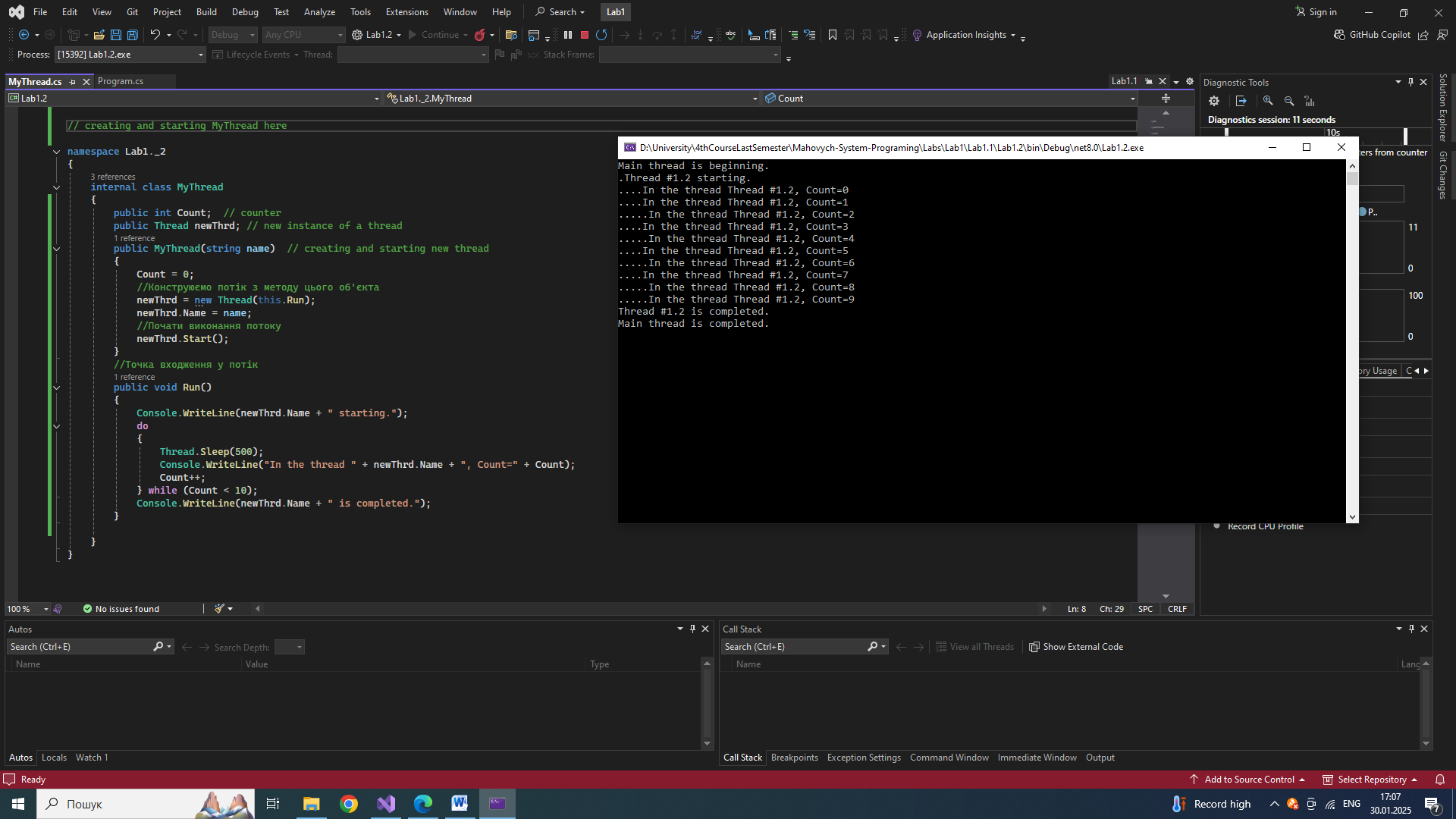
**Хід роботи**

**1 ПОВТОРИТИ ЗАДАЧІ**

**1.1 Створення і запуск потоку:**

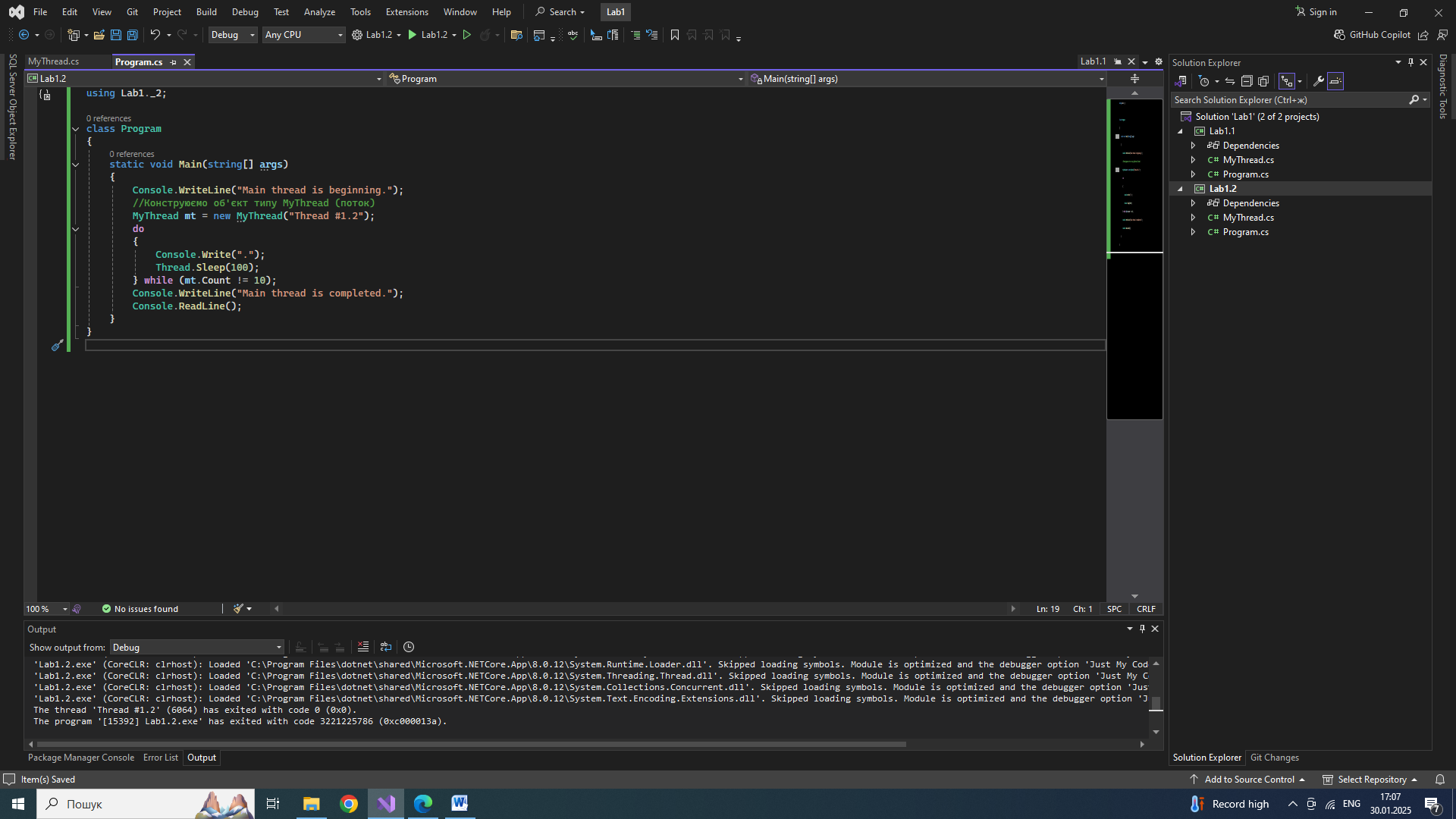
(результат виконання)

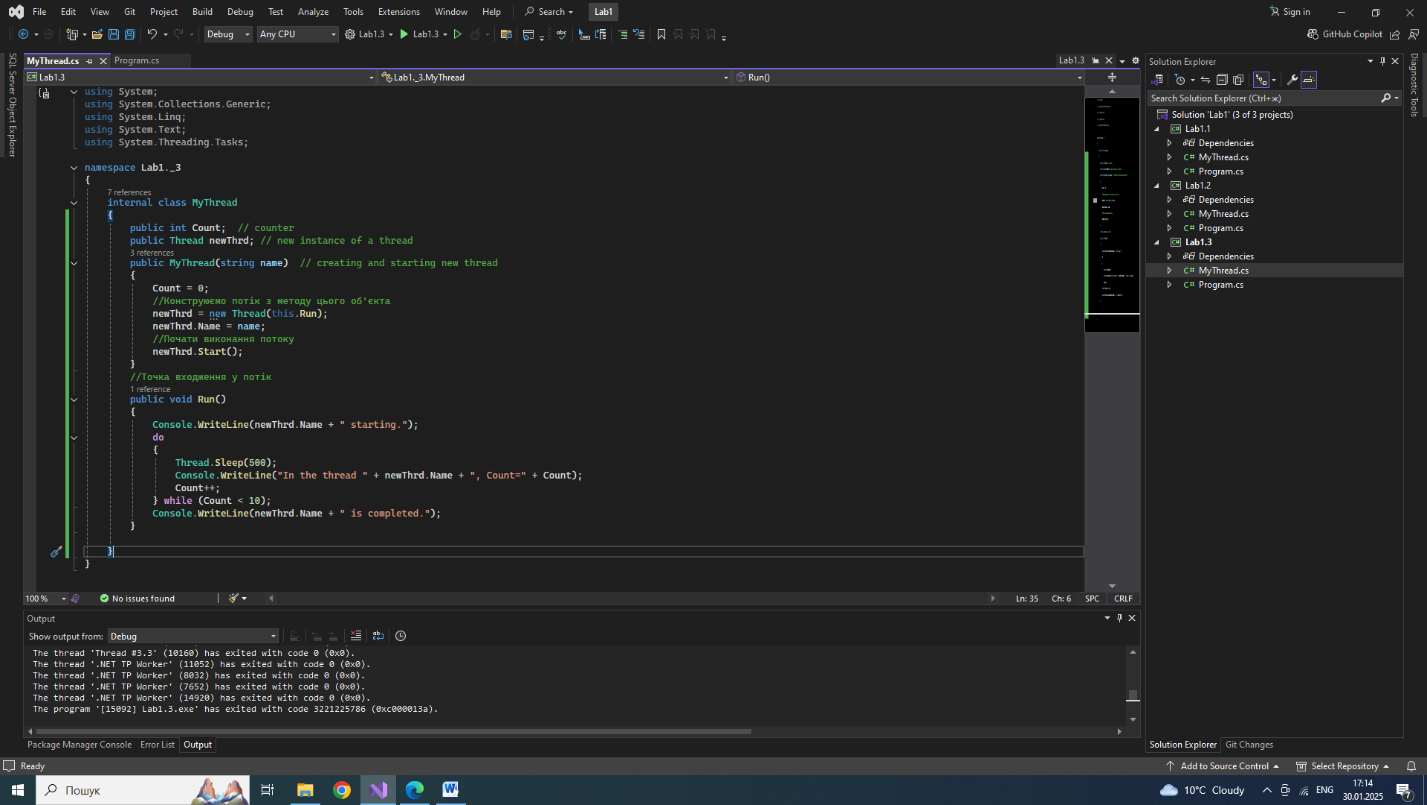
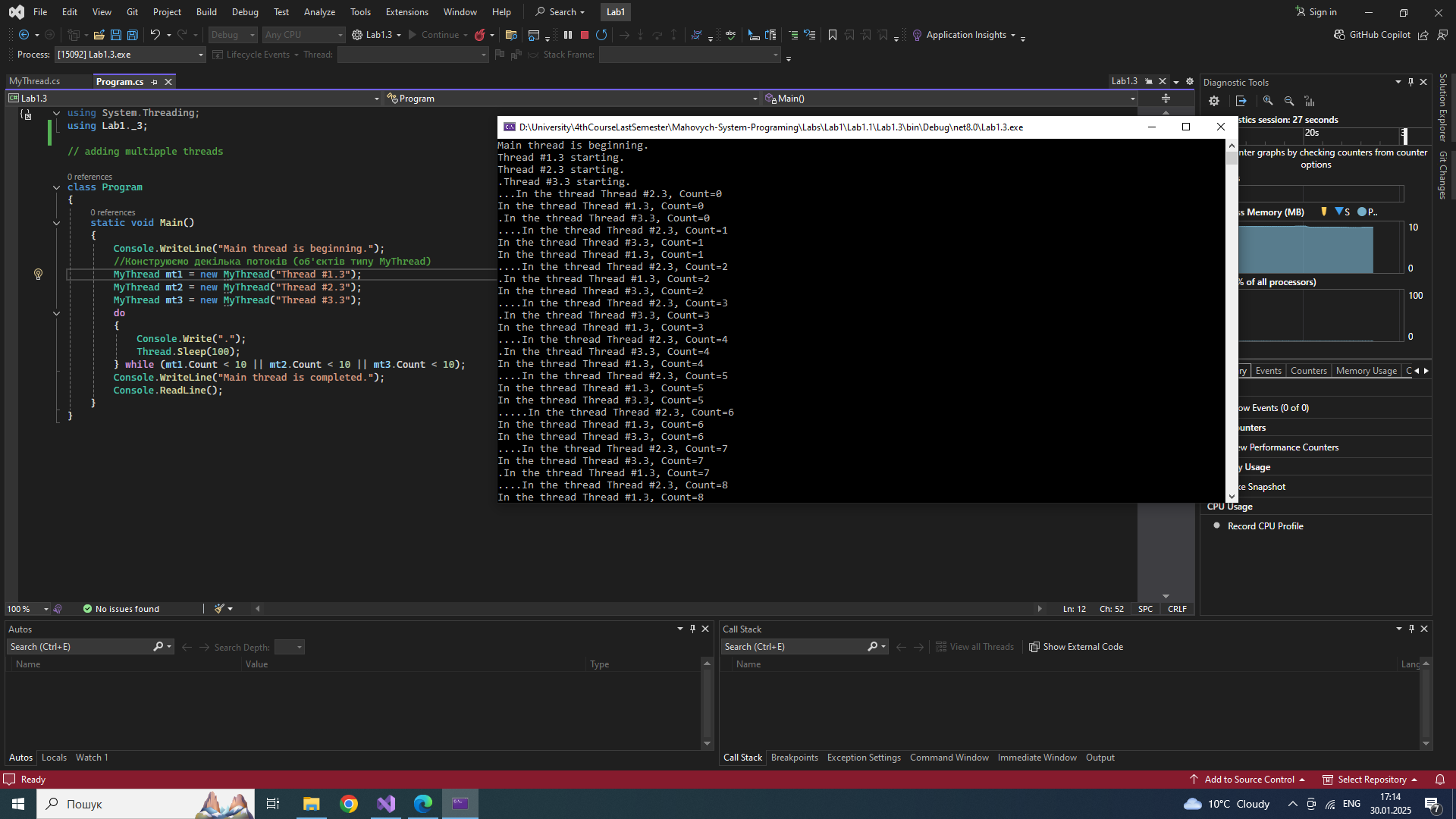
Точки позначають зовнішній потік, напис – внутрішній потік

(код)  
  
  
**1.2 Удосконалення багатопотокової програми:**

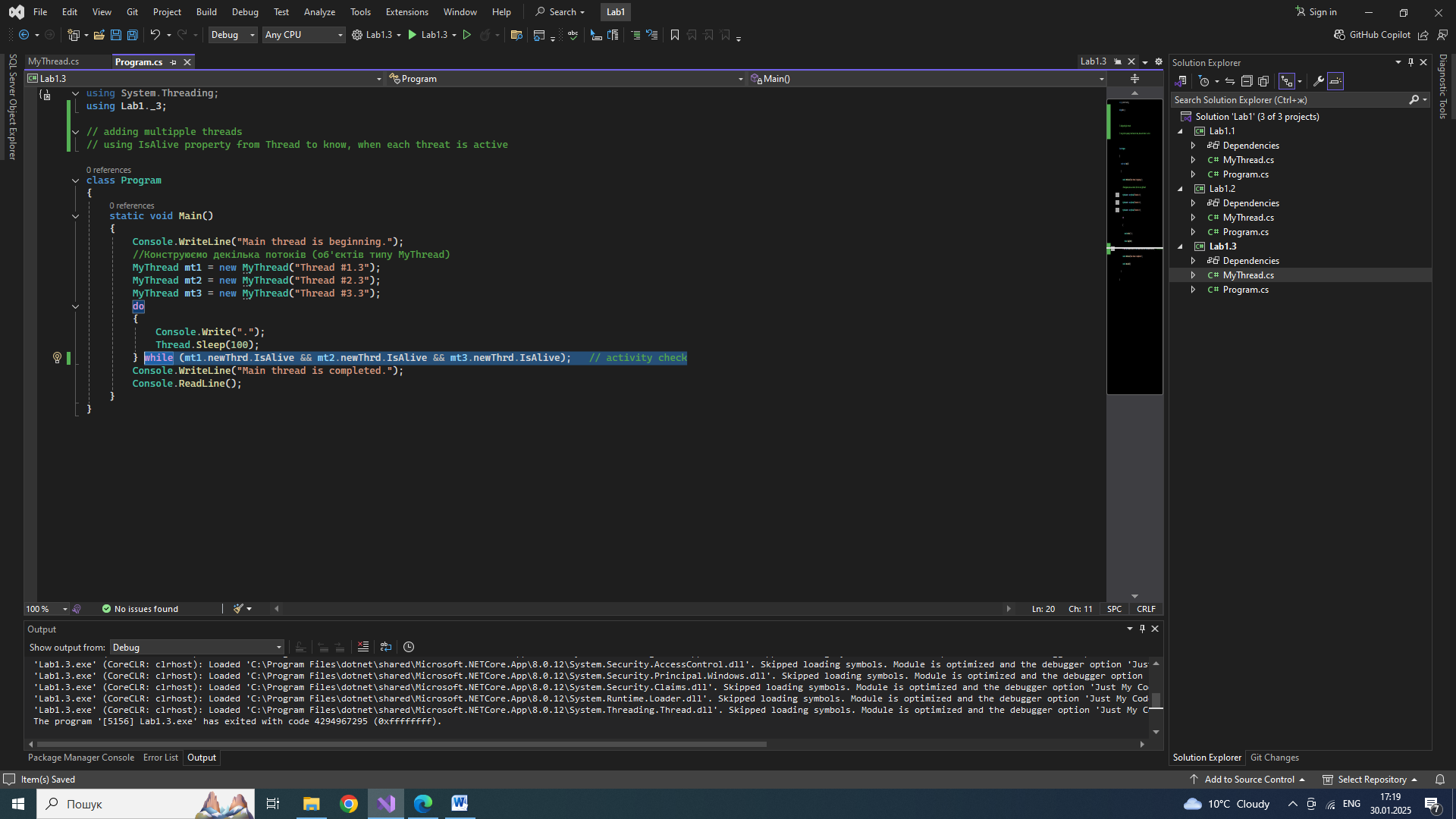
(результат виконання)

Створюємо та запускаємо потік з іншого місця програми, результат виконання такий самий як і в 1.1

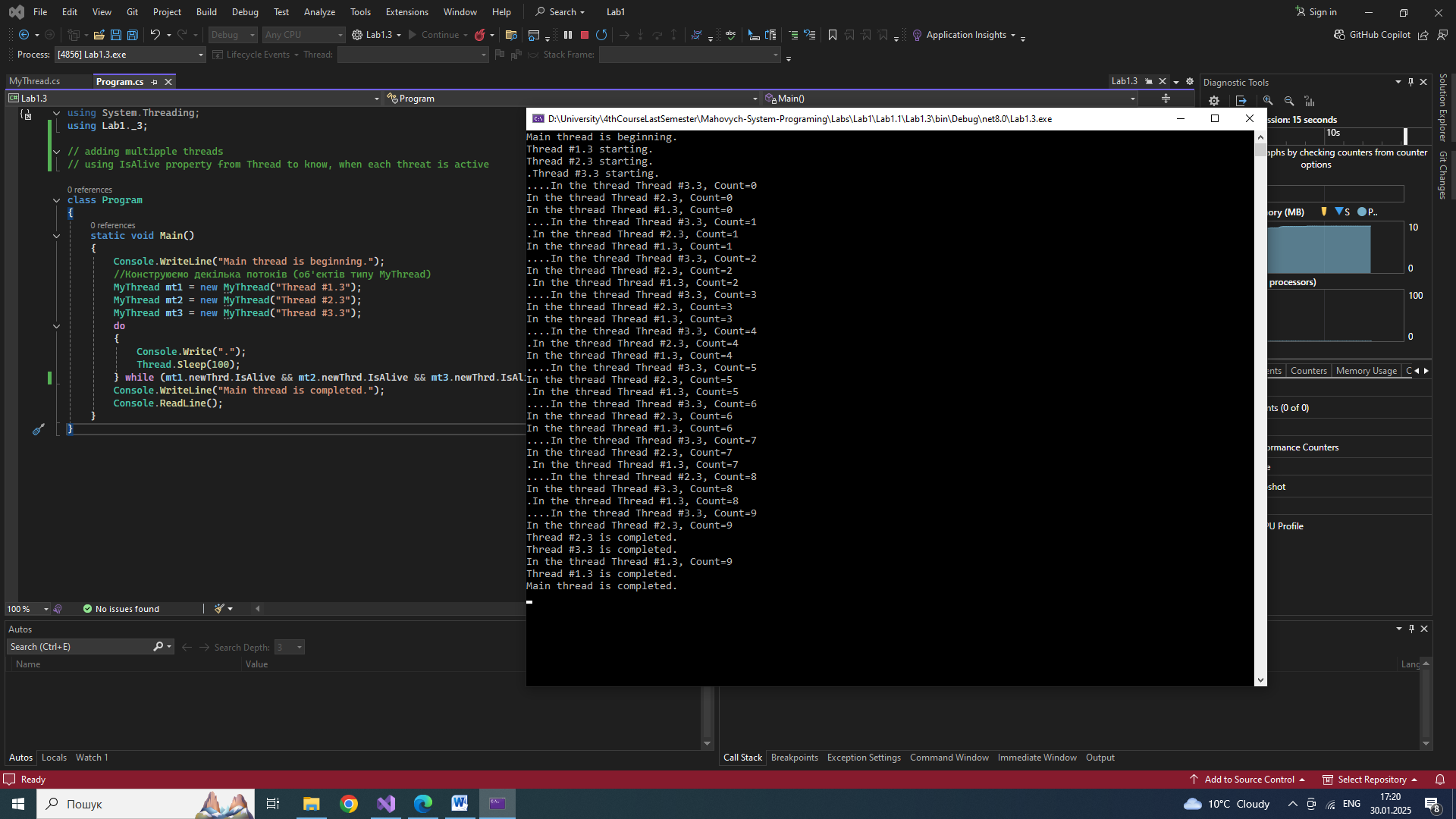
(файл Program.cs)

  
  
**1.3 Створення декількох потоків:**  
(створили додаткові потоки)

(не змінилося)

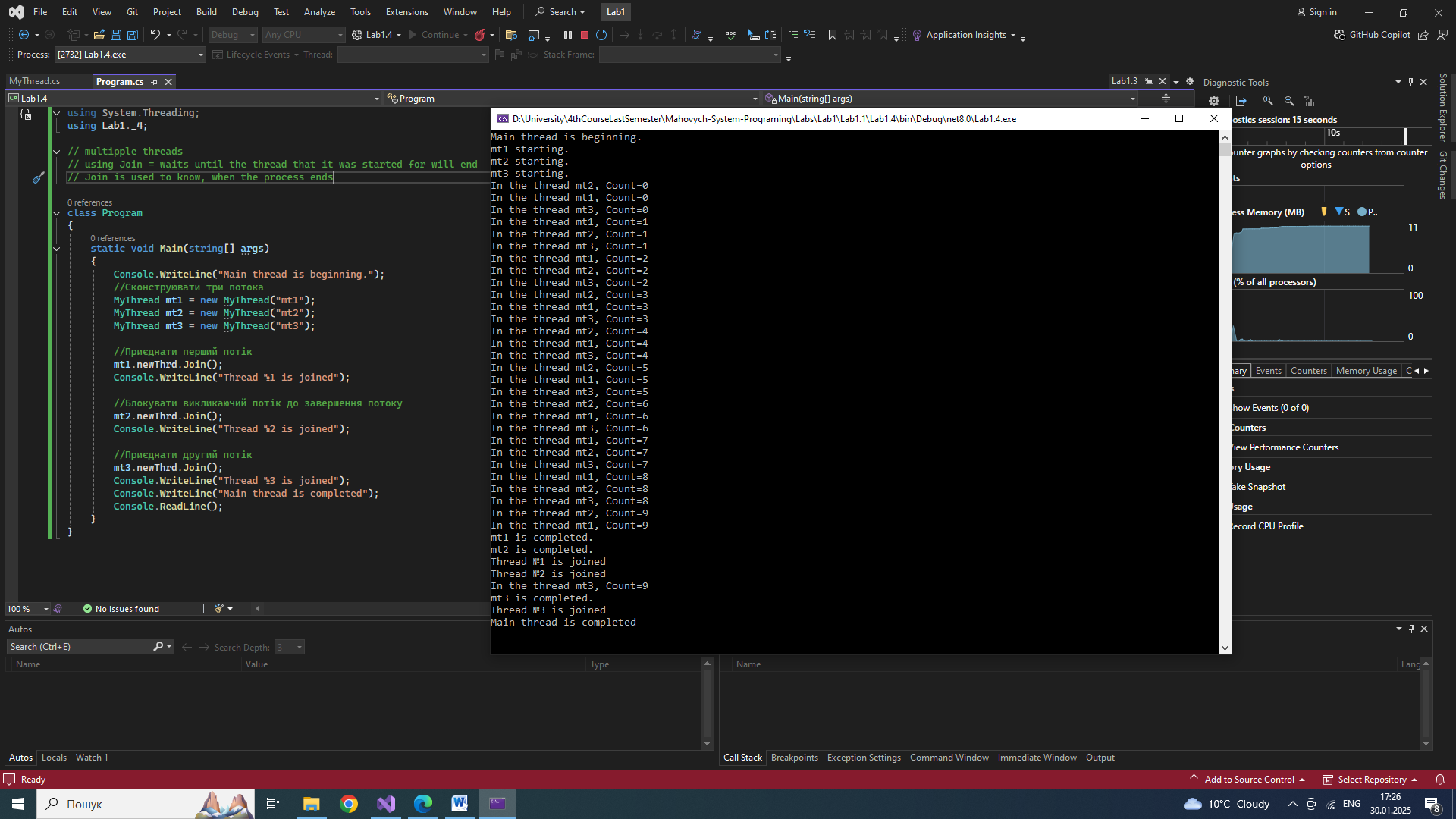
**1.4.1 Визначення моменту завершення потоку за допомогою IsAlive:**

(змінений код)

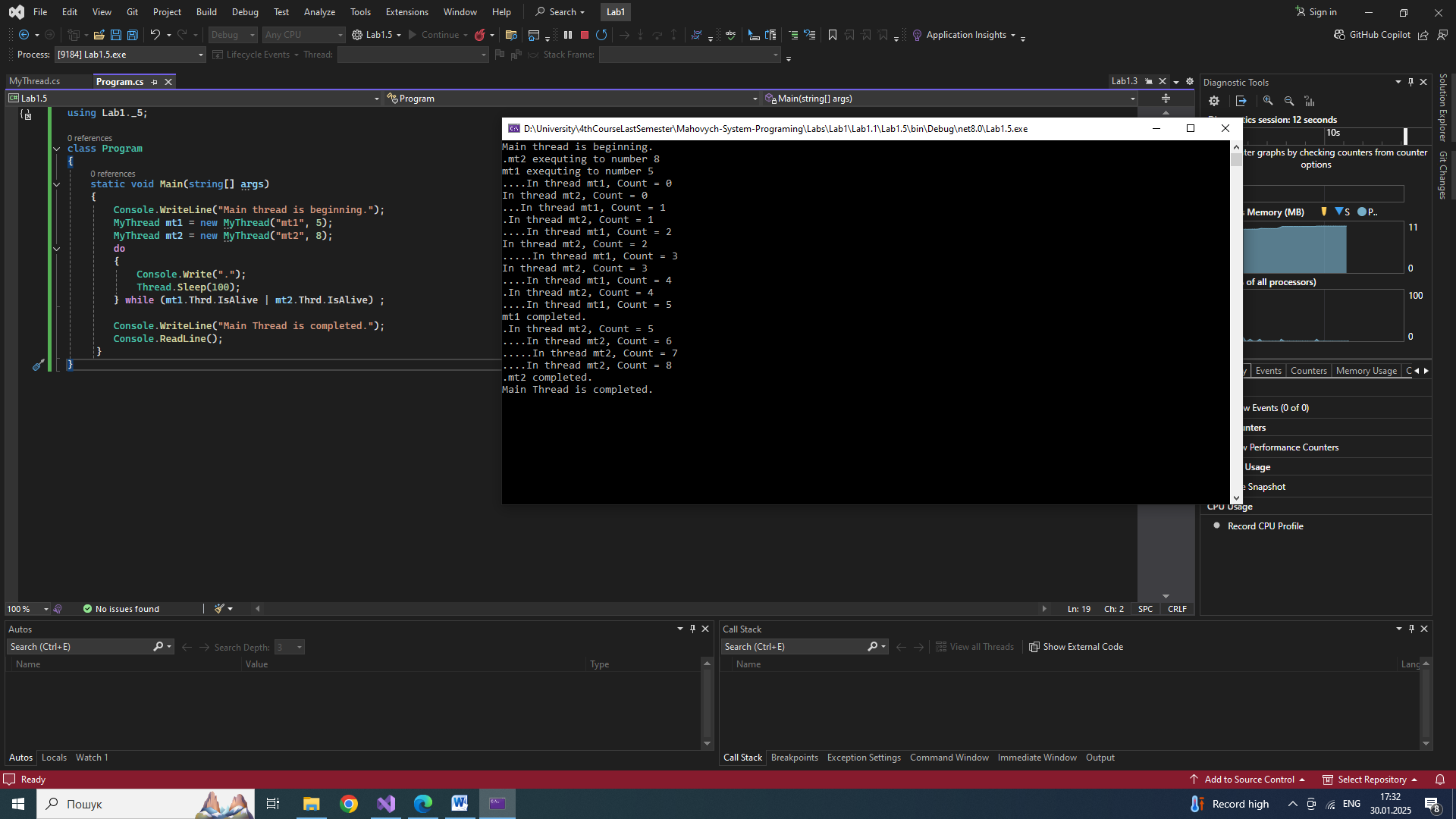


(виконання)

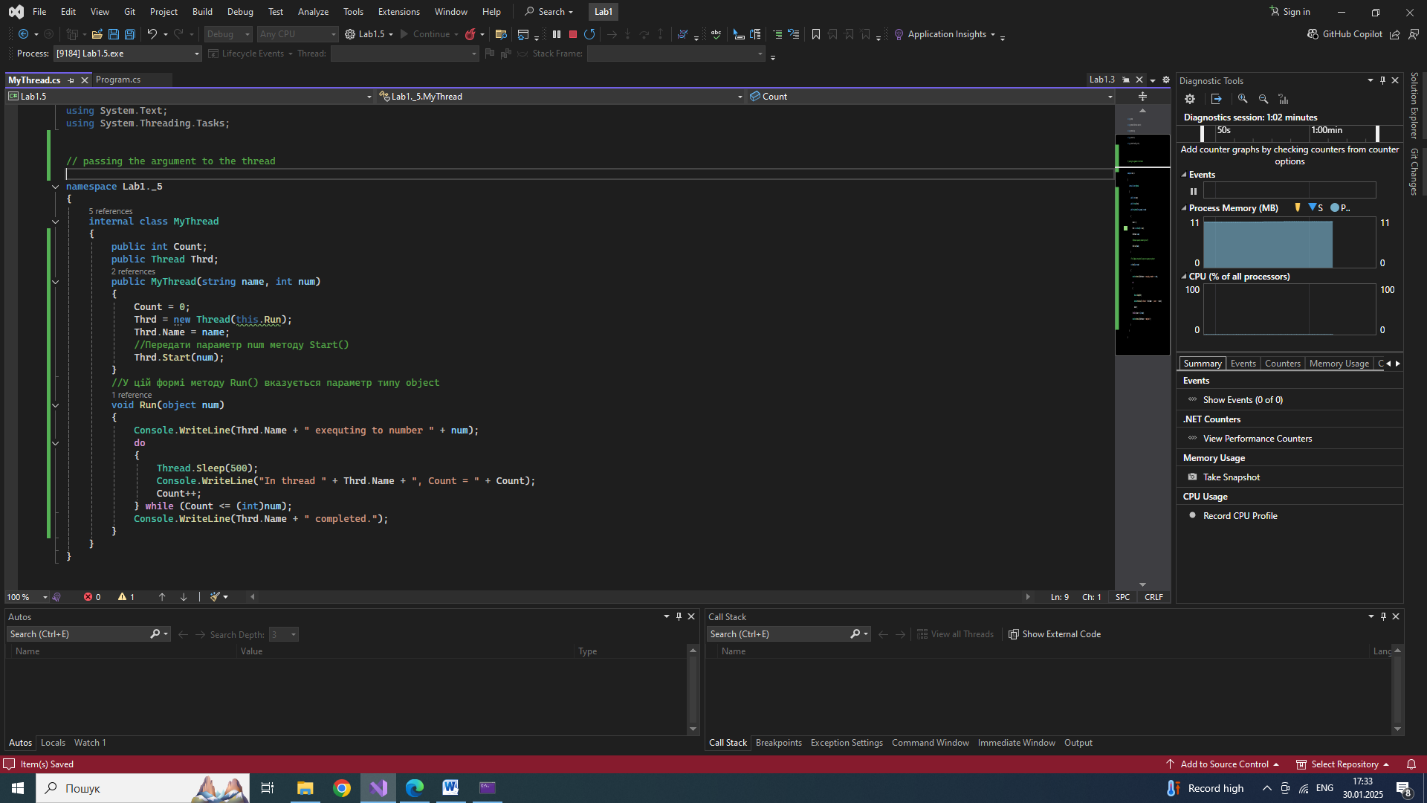
Бачимо, коли кожен потік завершив виконання

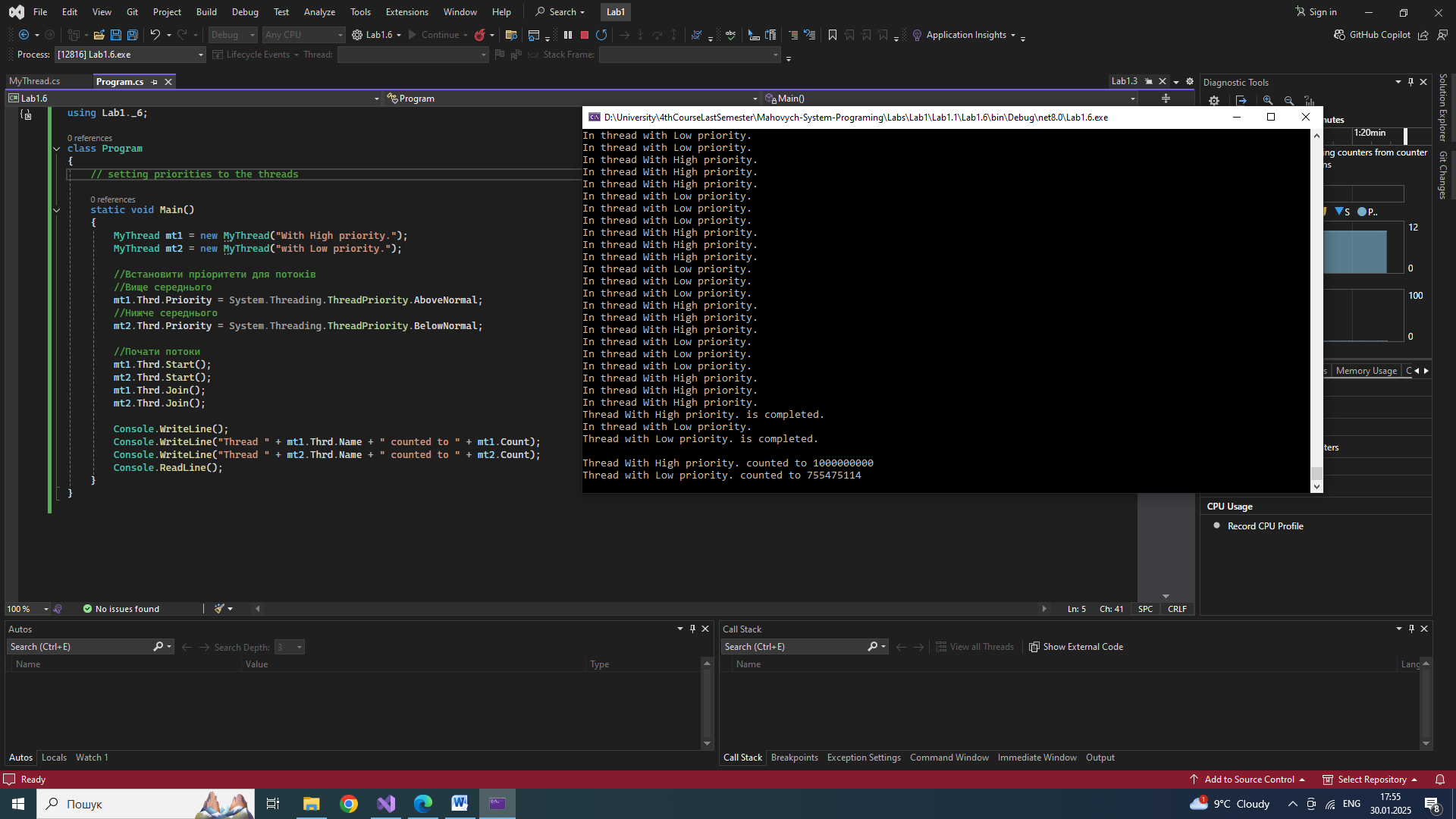
**1.4.2 Визначення моменту завершення потоку за допомогою Join**

(результат виконання)

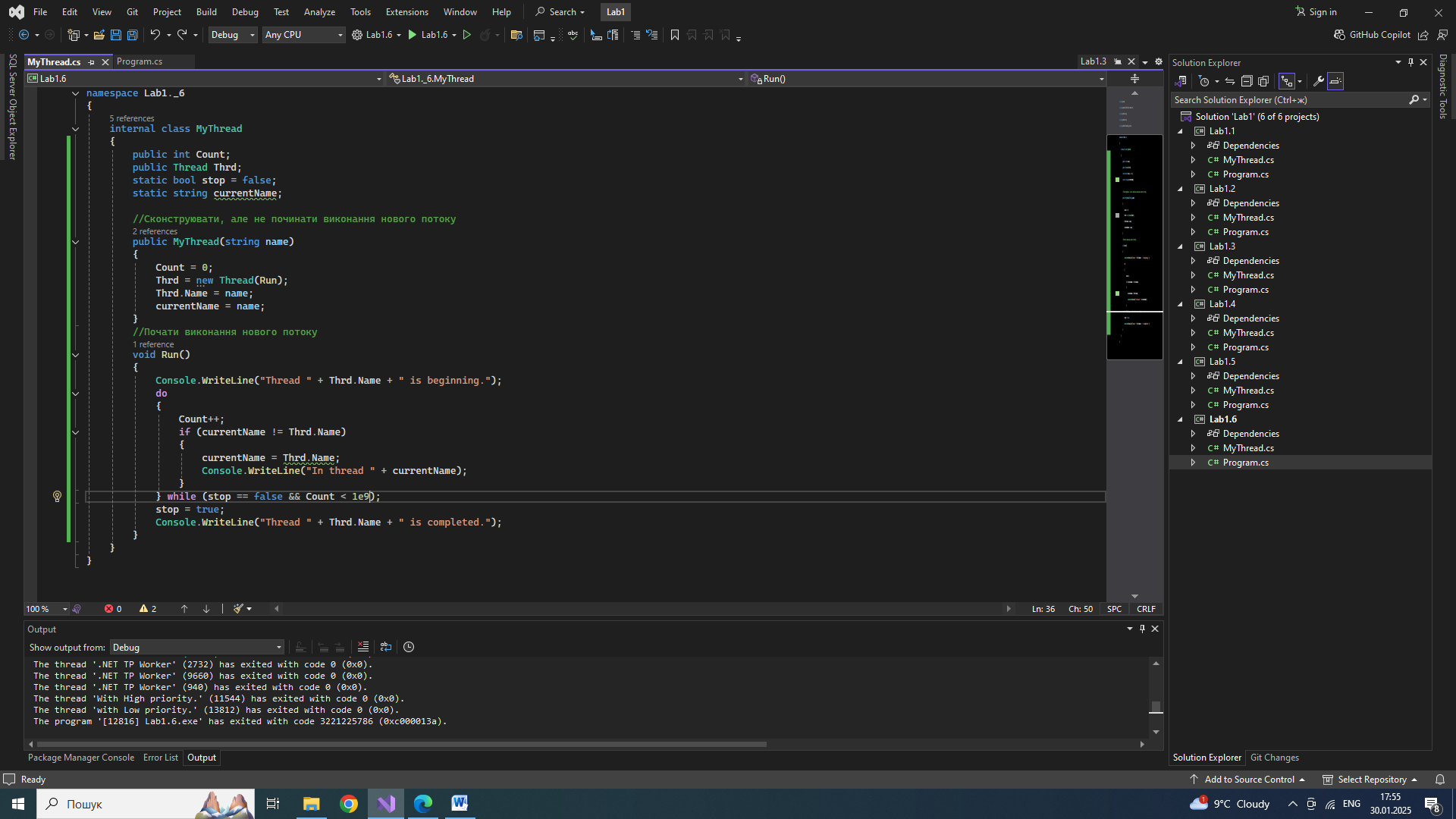
**1.5 Передача аргументу потоку, що визначає, коли настає зупинка:**

(виконання)

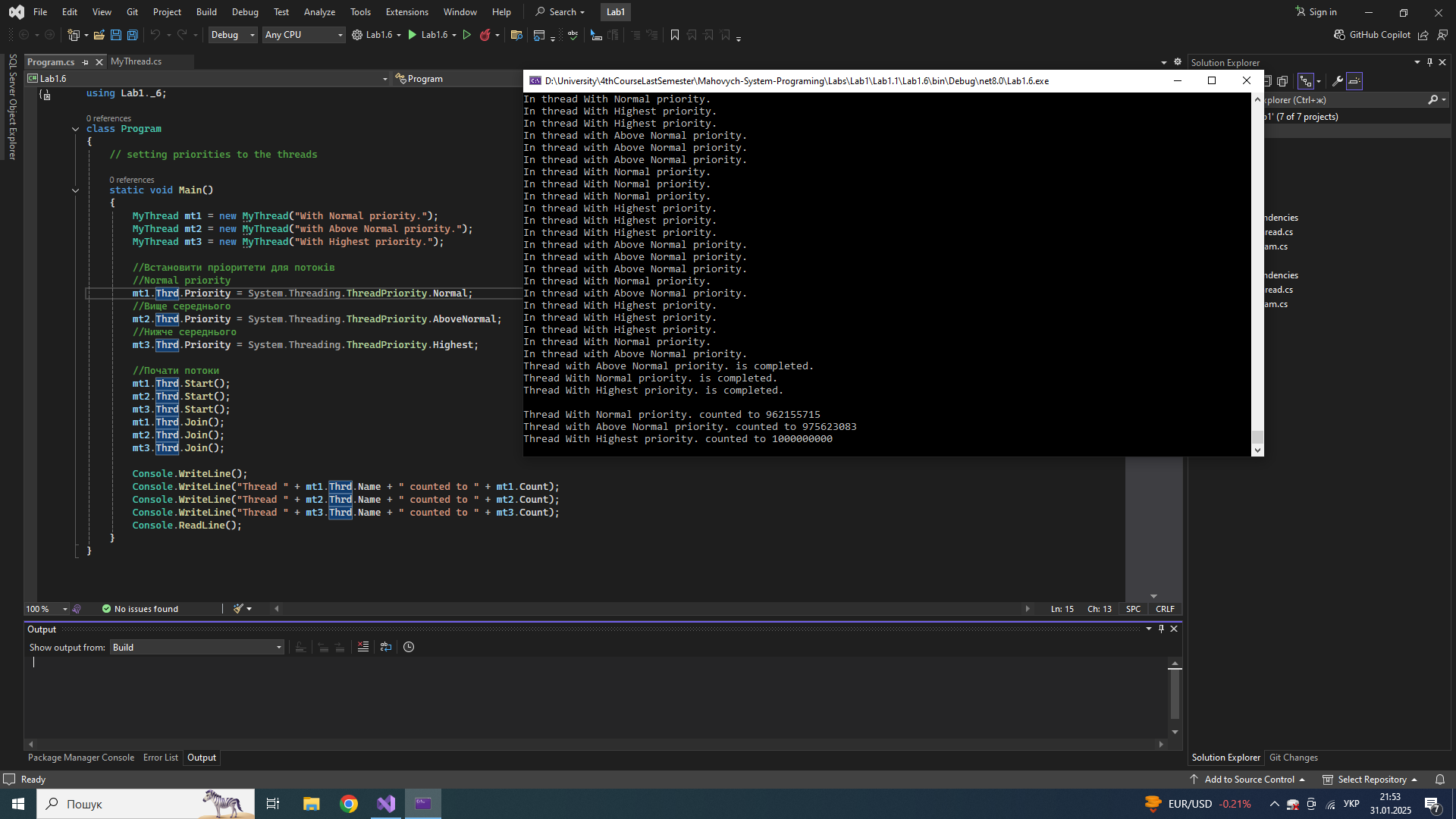
 (код MyThread.cs)

**1.6** **Пріоритети потоків:**

(виконання)

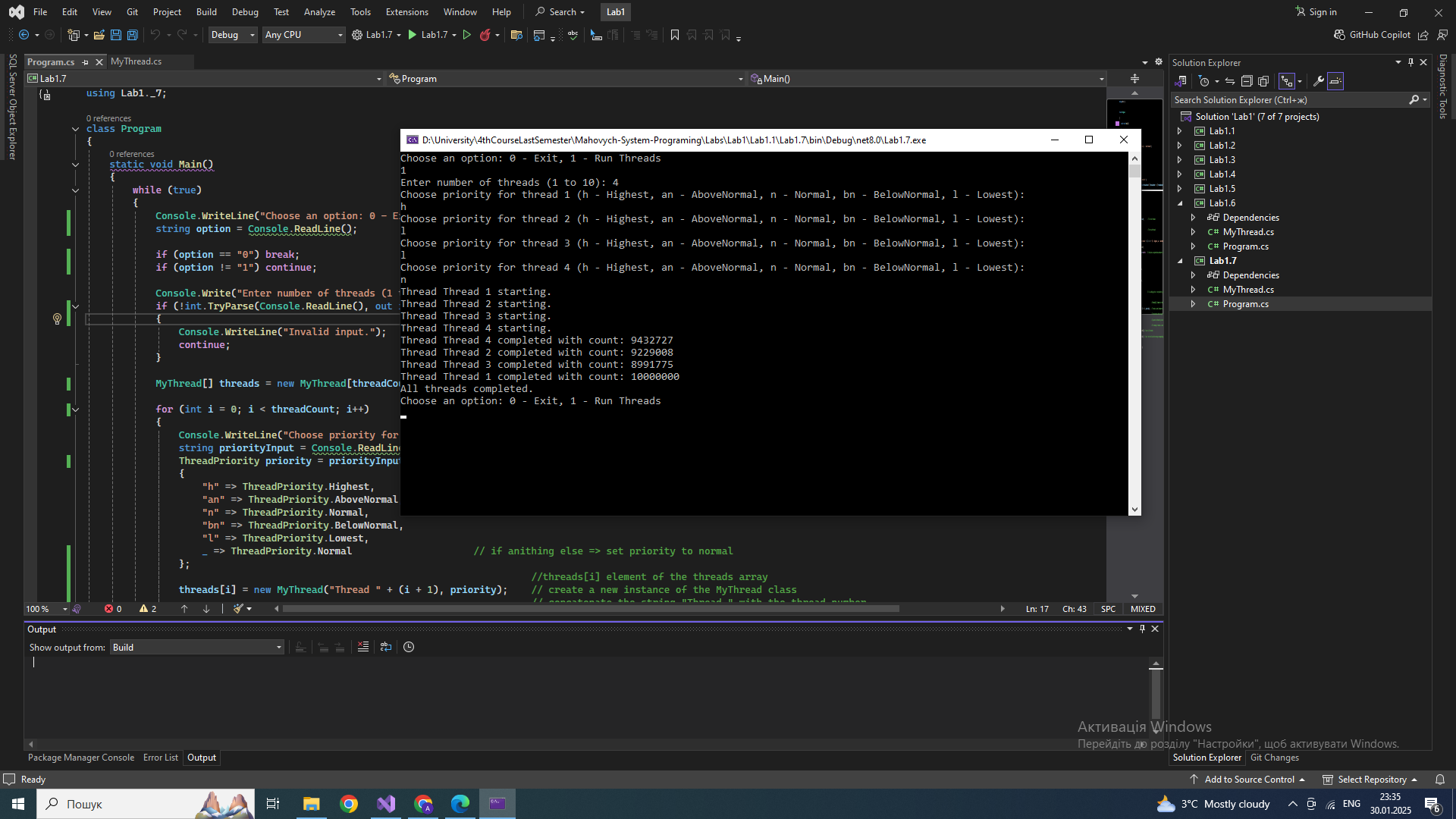
 (код MyThread.cs)

**2 БАГАТОПОТОКОВА ПРОГРАМА З РІЗНИМИ ПРІОРИТЕТАМИ**

 Варіант 1 = Потоків 3 шт. (пріоритети: середній, вище середнього, найвищий).

Розподіл процесорного часу між потоками у %:

* Normal 34.04%
* AboveNormal 33.21%
* Highest 34.04%

**3 ПРОГРАМА З ВИКОРИСТАННЯМ ГРАФІЧНОГО ІНТЕРФЕЙСУ, З МОЖЛИВІСТЮ ЗАДАННЯ КІЛЬКОСТІ ПОТОКІВ ТА ЇХНІ ПРІОРИТЕТИ**

(виконання)

Код файлу Program.cs:

using Lab1.\_7;

class Program

{

static void Main()

{

while (true)

{

Console.WriteLine("Choose an option: 0 - Exit, 1 - Run Threads");

string option = Console.ReadLine();

if (option == "0") break;

if (option != "1") continue;

Console.Write("Enter number of threads (1 to 10): ");

if (!int.TryParse(Console.ReadLine(), out int threadCount) || threadCount < 1 || threadCount > 10) // int.TryParse() attempts to convert a string

{ // out int threadCount = converted num stored in the threadCount

Console.WriteLine("Invalid input.");

continue;

}

MyThread[] threads = new MyThread[threadCount]; // list of threads

for (int i = 0; i < threadCount; i++) // for each thread

{

Console.WriteLine("Choose priority for thread " + (i + 1) + " (h - Highest, an - AboveNormal, n - Normal, bn - BelowNormal, l - Lowest): ");

string priorityInput = Console.ReadLine();

ThreadPriority priority = priorityInput switch // evaluate an expression and match it against cases

{

"h" => ThreadPriority.Highest,

"an" => ThreadPriority.AboveNormal,

"n" => ThreadPriority.Normal,

"bn" => ThreadPriority.BelowNormal,

"l" => ThreadPriority.Lowest,

\_ => ThreadPriority.Normal // if anithing else => set priority to normal

};

//threads[i] element of the threads array

threads[i] = new MyThread("Thread " + (i + 1), priority); // create a new instance of the MyThread class

// concatenate the string "Thread " with the thread number

} // parse the thread's priority

// to sum up: creates a new MyThread object for each thread

foreach (var thread in threads) thread.Start(); // start all threads

foreach (var thread in threads) thread.Join(); // wait for the finish for every coresponding thread

Console.WriteLine("All threads completed.");

}

}

}

Код файлу MyThread.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab1.\_7

{

internal class MyThread

{

private int count; // private counter

private Thread thread; // private field named thread of type Thread

private static bool stop = false; // stop flag; when one thread will reach the number -> all others will be stopped

private static string currentThreadName;

public MyThread(string name, ThreadPriority priority) // constructor for MyThread class with two parameters

{

count = 0; // counter within each thread

thread = new Thread(Run) { Name = name, Priority = priority }; // creates a new instance of the Thread class

}

public void Start() => thread.Start(); // method of MyThread class, starts the execution of the thread

public void Join() => thread.Join(); // method ..., waits until the thread that Join() was called on completes its work.

private void Run() // method ..., will be executed by each thread when it starts

{

Console.WriteLine("Thread " + thread.Name + " starting."); // thread was started

do // code inside will execute repeatedly until the condition after while is no longer true

{

count++;

if (currentThreadName != thread.Name) // used to track which thread is currently executing

{ // changes the name, if needed

currentThreadName = thread.Name;

//Console.WriteLine("In thread " + currentThreadName); // shows which thread is currently running

}

} while (!stop && count < 1e7); // exit when: this is false (if one of the threads reaches the end number)

// and

stop = true; // stop is set to true

Console.WriteLine("Thread " + thread.Name + " completed with count: " + count); // show with what count each thread was stopped

}

}

}

Висновок: у ході виконання цієї лабораторної роботи я створила багатопотокову програму, вдосконалила її та спостерігала, як пріоритет потоку впливає на час його виконання.