**Лабораторна робота №1**

**Процеси та потоки**

Мета: Вивчення роботи з процесами та потоками.

**Хід роботи**

**Завдання 1**

Необхідно написати дві програми (**три**), які будуть мати спільні дані  та одночасно до них звертатися.

Існує кілька механізмів реалізації спільного доступу до даних різних процесів.

Скористаємося одним з них, найбільш зручним - проектуванням файлу в пам'ять.

Одна програма буде сортувати дані у файлі, а інша відображати вміст цього файлу. Працювати обидва процеси будуть одночасно. **Третя програма** буде створювати (або заповнювати по новому) масив випадкових чисел.

Створіть файл data.dat. У ньому мають бути записані числа, згенеровані випадковим чином. Кількість чисел - 20-30 штук. Діапазон значень: від 10 до 100. (Це саме числа, а не символьні рядки зберігають ASCII коди цифр !!!)

**Програма №1. "Сортування даних" (консольна)**

Беремо за основу програму "Hello windows"  
Включаємо обробку події натискання клавіші, і відстежуємо в ньому натискання пробілу. Якщо користувач натиснув пробіл, значить починаємо сортування даних.  
Виконуємо проектування файлу в пам'ять. Використовуємо для цього створений файл data.dat. В результаті отримаємо доступ до даних як до звичайного одновимірного масиву.  
Виконуємо сортування масиву, будь-яким з методів сортування. Вставте 1-но секундну затримку для кожної ітерації сортування масиву, це дозволить потім наочніше побачити процес сортування.  
По закінченню сортування, програма виводить у вікно, рядок «Робота завершена».

**Програма №2. «Виведення файлу даних у вікно» (віконна)**  
Виконуємо проектування файлу в пам'ять. Використовуємо для цього створений файл data.dat. В результаті отримаємо доступ до даних як до звичайного одновимірного масиву. Цей же файл проектує в пам'ять попередня програма.  
Створюємо таймер на 0.5 секунди. При отриманні повідомлення від таймера, виконуємо висновок всього масиву в вікно. Передбачте коректний перевивід даних у вікно, без накладень. У вікно виводиться не числа з масиву, а рядки одного і того ж символу, наприклад «\*», в кількості, що дорівнює числу з масиву.

Запускаємо на виконання обидві програми одночасно. Коли друга програма запустилася і виконує висновок даних у вікно (виводить поки одну й ту ж саму картинку кожні пів секунди), натискаємо пробіл в першій програмі і вона починає сортувати масив. При цьому, так як вони дані беруть з одного і того ж файлу (обидві проектували його собі на згадку), то перша вносить зміни переставляючи дані при сортуванні, а друга виводить з себе у вікно і ми бачимо хід процесу сортування. Тимчасову затримку в першій програмі можна при потребі збільшити.

Ці дві програми демонструють можливість організації спільного доступу процесів до одних і тих самих даних. Так само демонструється механізм проектування файлу в пам'ять, як один з найкращих методів доступу до файлу.

**Завдання 2.**

Для коректної роботи зі спільними даними у цих двох програмах потрібно додати **синхронізацію потоків**, які можуть одночасно звертатися до спільних даних.

Для організації такої синхронізації потрібно використати об'єкт ядра ОС **mutex або semaphor**, або інший синхронізуючий об'єкт, а також **функції очікування** (наприклад, **WaitForSinglJbject()**).

Також обов'язковим є використання **обробки виняткових ситуацій** в роботі вище описаних трьох програм. Бо, некоректна робота будь якої з трьох, викличе неправильну роботу інших, через блокування спільних даних.

Для обробки виняткових ситуацій, необхідно правильно визначити **критичні секції коду** усіх написаних програм.

**Додаткове завдання.**

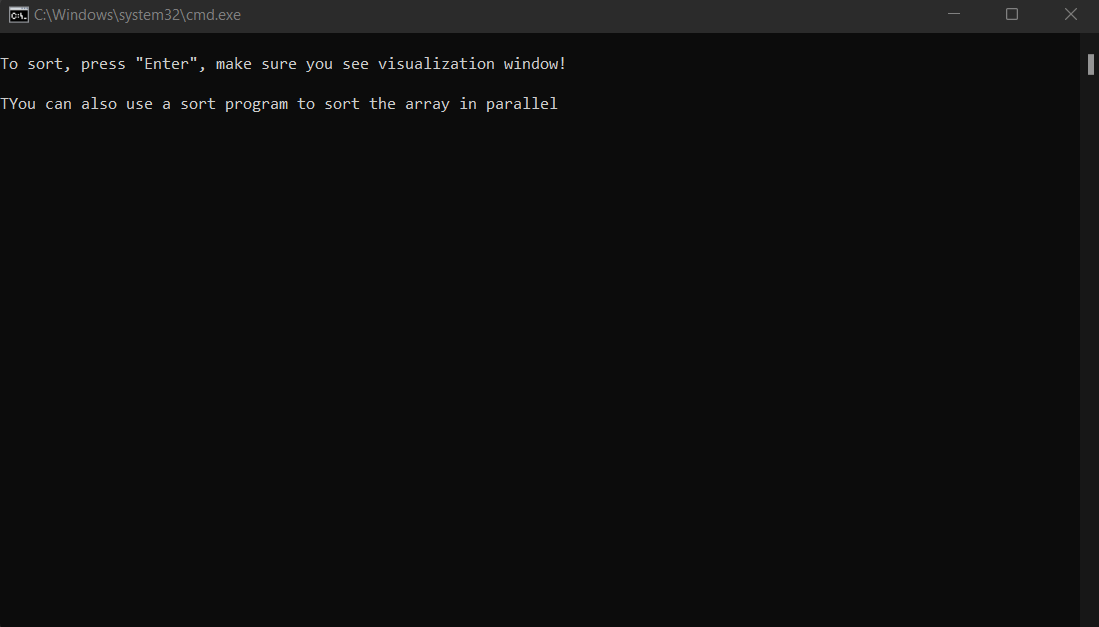
Написати **четверту програму** (консольну), яка буде одночасно працювати, та намагатися **відсортувати** той самий масив в **іншому напрямку** та **іншим** відомим **методом** **сортування**.

**Завдання підвищеної складності.**

Організувати роботу усіх чотирьох вищеописаних програм**на різних робочих станціях** локальної мережі але з **відображенням одного того самого файлу** даних. Для цього потрібно розібратися, як зробити видимим об'єкт "відображений файл" у локальній мережі.

Програма під назвою Create створює файл з назвою Data.data, створює процес і поток для доступу інших програми. Програма створює масив, заповнює його і записує його в файл:

Головне вікно програми:

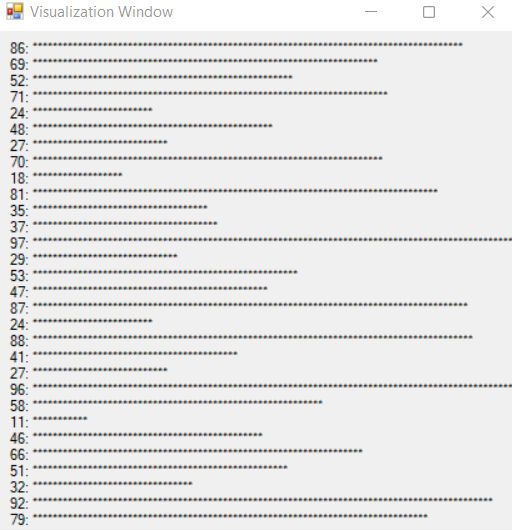


Лістинг програми:

|  |
| --- |
| namespace Create  {  internal class Program  {  static void Main(string[] args)  {  MemoryMappedFile mmf = MemoryMappedFile.CreateFromFile(@"d:\Data.data", FileMode.OpenOrCreate, "Numbers", 30 \* 4);  Mutex mut = new Mutex(false, "NumbMutex");  var myAccessor = mmf.CreateViewAccessor();  var stream = mmf.CreateViewStream();  int[] valueToWrite = new int[30];  Random rnd = new Random();  for (int i = 0; i < valueToWrite.Length; i++)  {  valueToWrite[i] = rnd.Next(10, 100);  }  try  {  mut.WaitOne();  myAccessor.WriteArray(0, valueToWrite, 0, valueToWrite.Length);  }  finally  {  mut.ReleaseMutex();  }  Process pr2 = new Process();  Process pr3 = new Process();  pr2.StartInfo.FileName = pr2.StartInfo.FileName = $@"G:\Study\3 grade\1 semen\SIS\Lab1\Visual\bin\Debug\task02.exe";  pr3.StartInfo.FileName = $@"G:\Study\3 grade\1 semen\SIS\Lab1\Sort\bin\Debug\Sort.exe";  pr2.Start();  pr3.Start();  Console.WriteLine("\nTo sort, press \"Enter\", make sure you see visualization window!");  Console.WriteLine("\nTYou can also use a sort program to sort the array in parallel");  Console.ReadLine();  Console.WriteLine("The sorting process is underway, you can view the process in the visualization window");  var handle = stream.SafeMemoryMappedViewHandle;  unsafe  {  byte\* pointer = null;  handle.AcquirePointer(ref pointer);  var size = 4 \* 30;  for (int i = 0; i < size - 4; i += 4)  {  for (int j = 4 + i; j < size; j += 4)  {  try  {  mut.WaitOne();  int first = \*(pointer + j),  second = \*(pointer + i);  if (first > second)  {  int temp;  temp = \*(pointer + j);  \*(pointer + j) = \*(pointer + i);  \*(pointer + i) = (byte)temp;  }  }  finally  {  mut.ReleaseMutex();  }  Thread.Sleep(100);  }  }  }  Console.WriteLine("Sorting complete to close all applications press Enter");  Console.ReadLine();  try  {  pr2.Kill();  }  catch  {  Console.WriteLine("Visualization window is closed");  }  try  {  pr3.Kill();  }  catch  {  Console.WriteLine("Sort program is closed");  }  Console.WriteLine("Work completed, you can close window: press \"Enter\"");  Console.ReadLine();  }  }  } |

Друга програма Visual, її завдання виводит вміст файлу Data.data у реальному часі

Головне вікно програми:

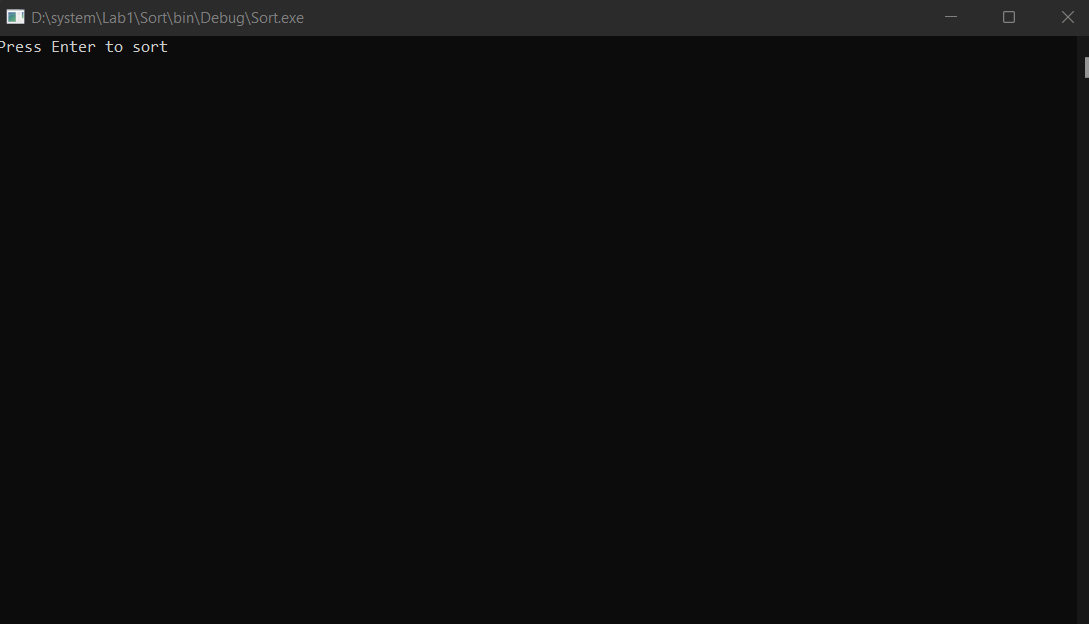


Лістинг програми:

|  |
| --- |
| namespace task02  {  public partial class Form1 : Form  {  public Form1()  {  InitializeComponent();  }  private void Form1\_Load(object sender, EventArgs e)  {  Visualize();  timer1.Enabled = true;  timer1.Start();  }  MemoryMappedFile mmf = MemoryMappedFile.OpenExisting("Numbers");  Mutex mut = Mutex.OpenExisting("NumbMutex");  public void Visualize()  {    try  {  mut.WaitOne();  string text = "";  var stream = mmf.CreateViewStream();  var handle = stream.SafeMemoryMappedViewHandle;  unsafe  {  byte\* pointer = null;  handle.AcquirePointer(ref pointer);  var size = 4 \* 30;  for (int i = 0; i < size; i += 4)  {  text += \*(pointer + i) + ": ";  for (int j = 0; j < \*(pointer + i); j++)  text += "\*";  text += "\n";  }  labelInf.Text = text;  }  }  finally  {  mut.ReleaseMutex();  }        }  private void timer1\_Tick(object sender, EventArgs e)  {  Visualize();  }  }  } |

Третя програма виконує функцію паралельного сортування масиву записаного в файл Data.data разом з програмою Create.

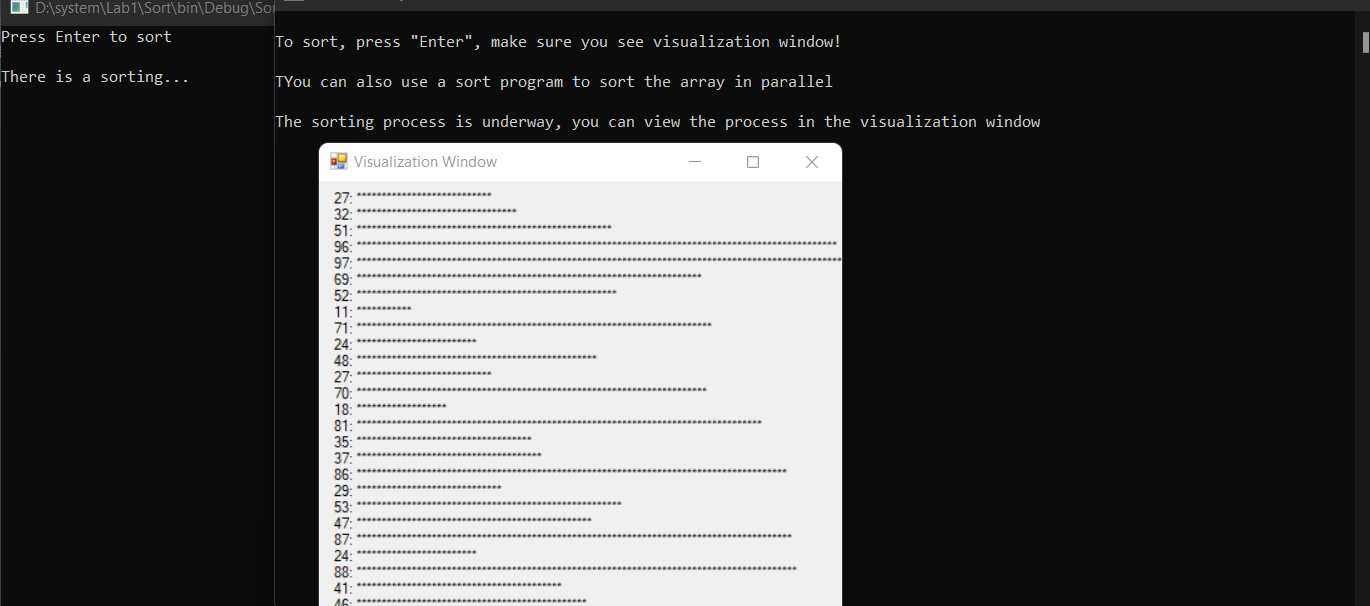
Головне вікно програми:



Лістинг програми:

|  |
| --- |
| namespace Sort  {  internal class Program  {  static void Main(string[] args)  {  while (true)  {  try  {  MemoryMappedFile mmf = MemoryMappedFile.OpenExisting("Numbers");  Mutex mut = Mutex.OpenExisting("NumbMutex");  Console.WriteLine("Press Enter to sort");  Console.ReadLine();  Console.WriteLine("There is a sorting...");  var stream = mmf.CreateViewStream();  var handle = stream.SafeMemoryMappedViewHandle;  unsafe  {  byte\* pointer = null;  handle.AcquirePointer(ref pointer);  var size = 4 \* 30;  for (int i = size - 4; i >= 0; i -= 4)  {  for (int j = size - 4; j >= 4; j -= 4)  {  try  {  mut.WaitOne();  if (\*(pointer + j) < \*(pointer + j - 4))  {  int temp;  temp = \*(pointer + j);  \*(pointer + j) = \*(pointer + j - 4);  \*(pointer + j - 4) = (byte)temp;  }  }  finally  {  mut.ReleaseMutex();  }  Thread.Sleep(100);  }  }  }  Console.WriteLine("Work completed, you can close window: press \"Enter\"");  Console.ReadLine();  }  catch (FileNotFoundException)  {  Console.WriteLine("Error, create program is not running");  Console.ReadLine();  }  catch (WaitHandleCannotBeOpenedException)  {  Console.WriteLine("Error, restart create program");  Console.ReadLine();  }  }  }  }  } |

Програми у робочому стані:



**Висновки:** набув навички роботи з процесами та потоками.