**ЛАБОРАТОРНА РОБОТА № 1**

**Взаємодія між процесами.****Розподіл даних між процесами.   
Робота з файлами, які відображуються у пам'ять**

***Мета******:*** отримання базових навичок з взаємодії між процесами, розподілу даних між процесами. Навчитися працювати з файлами які відображуються у пам'ять.

**Хід роботи:**

Завдання №1: Необхідно написати дві програми, 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 !!!)

Лістинг програми (додаток 1):

using System;

using System.Diagnostics;

using System.IO;

using System.IO.MemoryMappedFiles;

using System.Text;

using System.Threading;

namespace Program1

{

class Program

{

static byte size = 30;

static void Main(string[] args)

{

Console.OutputEncoding = Encoding.Unicode;

Console.InputEncoding = Encoding.Unicode;

MemoryMappedFile memory = MemoryMappedFile.CreateFromFile(@"I:\Yaroslav\3kurs\Системне програмування\Lab01\data.dat",

FileMode.OpenOrCreate, "Numbers", size);

Semaphore semaphore = new Semaphore(3, 3, "NumbSem");

var accessor = memory.CreateViewAccessor();

var stream = memory.CreateViewStream();

byte[] numbersToWrite = new byte[size];

Random random = new Random();

string arr = "";

for (int i = 0; i < numbersToWrite.Length; i++)

{

numbersToWrite[i] = (byte)random.Next(10, 100);

arr += numbersToWrite[i] + " ";

Console.Write(numbersToWrite[i] + " ");

}

try

{

semaphore.WaitOne();

accessor.WriteArray(0, numbersToWrite, 0, numbersToWrite.Length);

}

finally

{

semaphore.Release();

}

Process proccess2 = new Process();

Process proccess3 = new Process();

proccess2.StartInfo.FileName = "I:\\Yarosalv\\3kurs\\Системне програмування\\Lab01\\Program2\\bin\\Debug\\Program2.exe";

proccess3.StartInfo.FileName = "I:\\Yarosalv\\3kurs\\Системне програмування\\Lab01\\Program3\\bin\\Debug\\Program3.exe";

proccess2.Start();

proccess3.Start();

Console.WriteLine("\nPush space");

while (Console.ReadKey().Key != ConsoleKey.Spacebar)

{

Console.WriteLine("Error");

}

Console.Clear();

Console.WriteLine(arr);

var handle = stream.SafeMemoryMappedViewHandle;

unsafe

{

byte\* pointer = null;

handle.AcquirePointer(ref pointer);

var d = size / 2;

while (d >= 1)

{

for (var i = d; i < size; i++)

{

try

{

semaphore.WaitOne();

var j = i;

while ((j >= d) && (\*(pointer + j - d) > \*(pointer + j)))

{

Swap(ref \*(pointer + j), ref \*(pointer + j - d));

j = j - d;

}

}

finally

{

semaphore.Release();

}

Thread.Sleep(100);

}

d = d / 2;

}

}

Console.WriteLine("Done!");

Console.ReadLine();

try

{

proccess2.Kill();

}

catch

{

Console.WriteLine("Program 2 disable");

}

try

{

proccess3.Kill();

}

catch

{

Console.WriteLine("Program 3 disable");

}

Console.WriteLine("Final");

}

static void Swap(ref byte a, ref byte b)

{

var t = a;

a = b;

b = t;

}

}

}

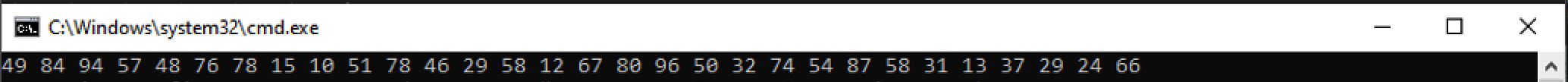


Рис. 1. Результат виконання програми

Лістинг програми (додаток 2):

using System;

using System.IO.MemoryMappedFiles;

using System.Threading;

using System.Windows.Forms;

namespace Program2

{

public partial class Form1 : Form

{

static byte size = 30;

MemoryMappedFile memory = MemoryMappedFile.OpenExisting("Numbers");

Semaphore semaphore = Semaphore.OpenExisting("NumbSem");

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

Render();

timer1.Enabled = true;

timer1.Start();

}

public void Render()

{

try

{

semaphore.WaitOne();

string res = "";

var stream = memory.CreateViewStream();

var handle = memory.CreateViewStream().SafeMemoryMappedViewHandle;

unsafe

{

byte\* pointer = null;

handle.AcquirePointer(ref pointer);

for (int i = 0; i < size; i++)

{

for (int j = 0; j < \*(pointer + i); j++)

res += "\*";

res += $" {\*(pointer + i)}";

res += "\n";

}

label1.Text = res;

}

}

finally

{

semaphore.Release();

}

}

private void timer1\_Tick(object sender, EventArgs e)

{

Render();

}

}

}

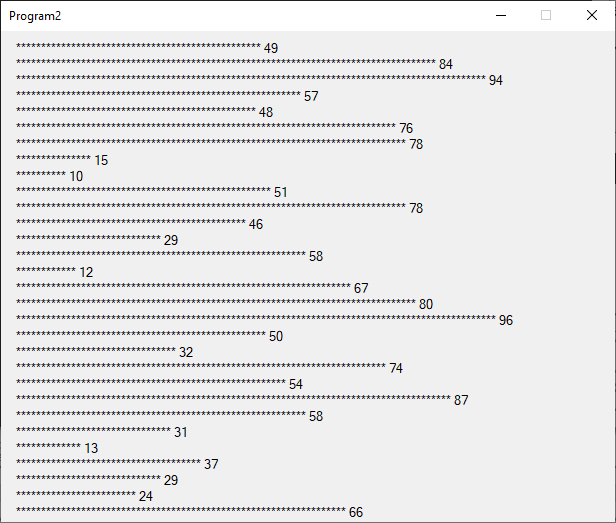


Рис. 2. Результат виконання програми (додаток 2)

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.

Лістинг програми (додаток 3):

using System;

using System.IO;

using System.IO.MemoryMappedFiles;

using System.Text;

using System.Threading;

namespace Program3

{

class Program

{

static byte size = 30;

static void Main(string[] args)

{

Console.OutputEncoding = Encoding.Unicode;

Console.InputEncoding = Encoding.Unicode;

try

{

MemoryMappedFile mmf = MemoryMappedFile.OpenExisting("Numbers");

Semaphore semaphore = Semaphore.OpenExisting("NumbSem");

Console.WriteLine("Push space");

while (Console.ReadKey().Key != ConsoleKey.Spacebar)

{

Console.WriteLine("Error");

}

Console.Clear();

var stream = mmf.CreateViewStream();

var handle = stream.SafeMemoryMappedViewHandle;

unsafe

{

byte\* pointer = null;

handle.AcquirePointer(ref pointer);

for (int i = 1; i < size; i++)

{

for (int j = 0; j < size - i; j++)

{

try

{

semaphore.WaitOne();

if (\*(pointer + j) < \*(pointer + j + 1))

{

Swap(ref \*(pointer + j), ref \*(pointer + j + 1));

}

}

finally

{

semaphore.Release();

}

Thread.Sleep(100);

}

}

}

Console.WriteLine("Done");

Console.ReadLine();

}

catch (FileNotFoundException)

{

Console.WriteLine("Error");

Console.ReadLine();

}

catch (WaitHandleCannotBeOpenedException)

{

Console.ReadLine();

}

}

static void Swap(ref byte a, ref byte b)

{

var t = a;

a = b;

b = t;

}

}

}

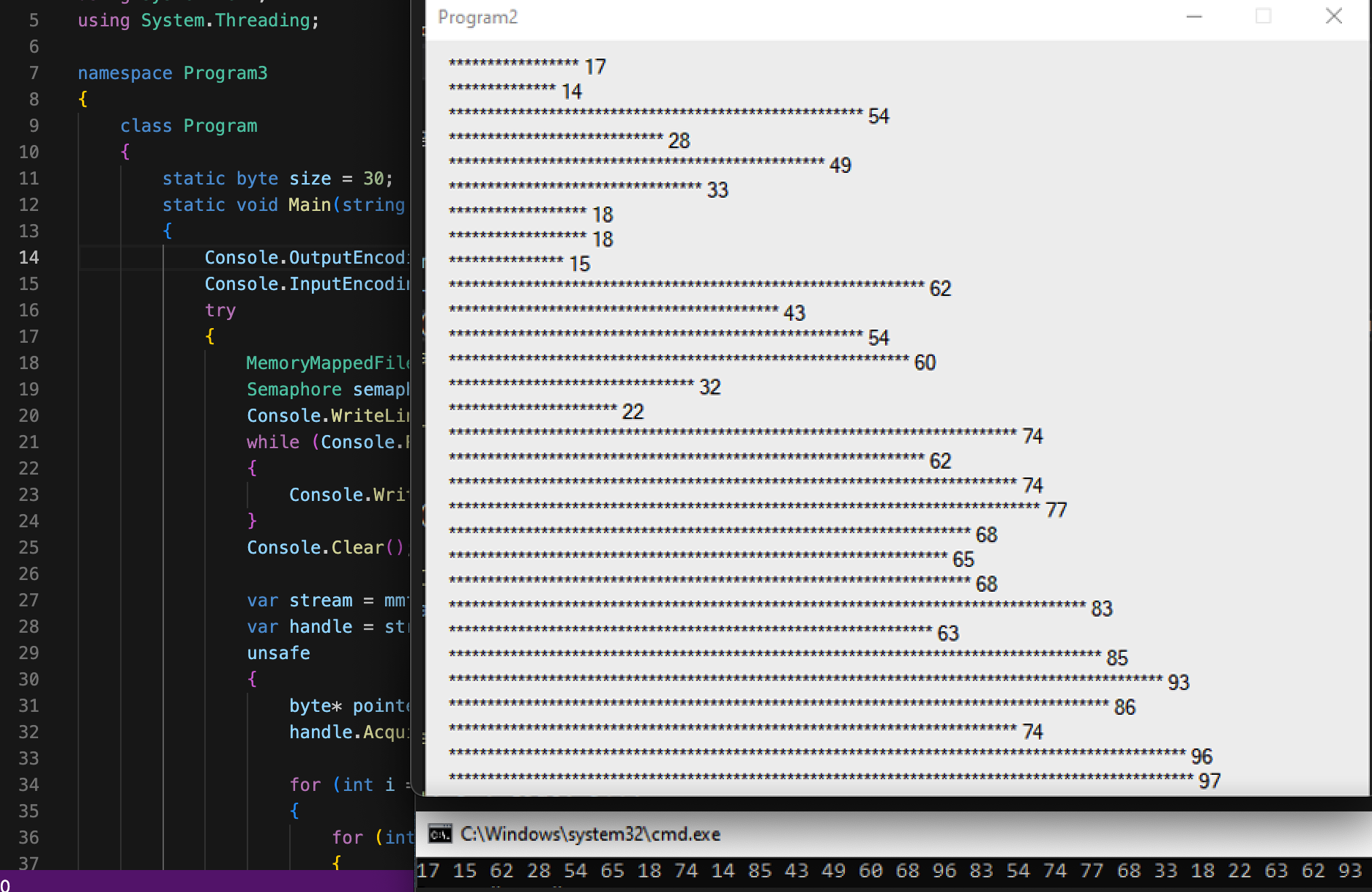


Рис. 3. Результат виконання програм

**Висновки:** було отримано базові навички з взаємодії між процесами, розподілу даних між процесами. Було вивчено основи роботи з файлами, які відображуються у пам'ять.