Міністерство освіти і науки України

Дніпровський національний університет ім. Олеся Гончара

Факультет фізики, електроніки та комп’ютерних систем

Кафедра ЕОМ

Звіт

до лабораторної роботи №4

з теми «Відображення файлу у пам’яті»

з дисципліни «Системне програмне забезпечення»

Виконав:

студент групи КІ-21-2

спеціальності

123 «Комп’ютерна інженерія»

Лебідь Олександр Павлович

Перевірив:

Герасимов Володимир Володимирович

Дніпро

2024

**Мета роботи:** ознайомитися з методами для роботи з постійно та непостійно зіставленими у памяті файлами, розглянути різницю у їх застосуванні.

**Хід роботи**

**Завдання**

1. Виконати усі приклади розглянутих програм.

2. Переробити приклади 4.1 і 4.2 так, щоб записувати і зчитувати

рядок.

**l\_4\_1.cs:**

using Microsoft.VisualBasic;

using System;

using System.Collections.Generic;

using System.IO;

using System.IO.MemoryMappedFiles;

namespace l\_4\_1

{

class Program

{

static void Main(string[] args)

{

string str;

int[] mas = new int[10];

Console.Write("Введiть рядок з 10 чисел: ");

str = Console.ReadLine()!;

string[] strArr = str.Split(' ');

mas = Array.ConvertAll(strArr, int.Parse);

Console.WriteLine();

List<int> inputMas = new List<int>();

Console.WriteLine("Масив.");

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

{

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

}

Console.WriteLine("\n Зiставлений у пам’ятi файл з файлу на диску.");

using (MemoryMappedFile mnf = MemoryMappedFile.CreateFromFile("a1.dta", FileMode.OpenOrCreate, "file", 1))

{

using (MemoryMappedViewStream stream = mnf.CreateViewStream())

{

BinaryWriter writer = new BinaryWriter(stream);

foreach (int i in mas)

{

writer.Write(i);

}

writer.Close();

Console.WriteLine("\n Файл на диску створений та закритий.");

}

using (MemoryMappedViewStream stream = mnf.CreateViewStream())

{

BinaryReader reader = new BinaryReader(stream);

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

{

inputMas.Add(reader.ReadInt32());

}

}

foreach (int i in inputMas)

{

Console.Write(i + " ");

}

Console.ReadKey();

}

}

}

}

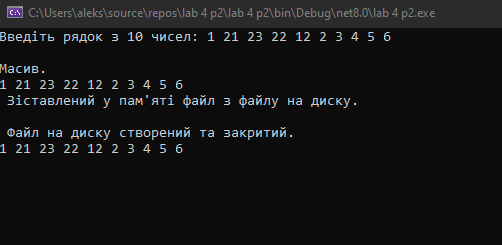


Рис. 1 – Результат прикладу 4.1

**l\_4\_2.cs:**

using Microsoft.VisualBasic;

using System;

using System.Collections.Generic;

using System.IO;

using System.IO.MemoryMappedFiles;

namespace l\_4\_2

{

class Program

{

static void Main(string[] args)

{

string str;

int[] mas = new int[10];

Console.Write("Введiть рядок з 10 чисел: ");

str = Console.ReadLine()!;

string[] strArr = str.Split(' ');

mas = Array.ConvertAll(strArr, int.Parse);

List<int> inputMas = new List<int>();

Console.WriteLine("Масив.");

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

{

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

}

Console.WriteLine("\n Непостiйний зiставлений у пам’ятi файл.");

using (MemoryMappedFile mnf = MemoryMappedFile.CreateNew("n\_file", 4096))

{

using (MemoryMappedViewStream stream = mnf.CreateViewStream())

{

BinaryWriter writer = new BinaryWriter(stream);

foreach (int i in mas)

writer.Write(i);

writer.Close();

}

using (MemoryMappedViewStream stream = mnf.CreateViewStream())

{

BinaryReader reader = new BinaryReader(stream);

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

{

inputMas.Add(reader.ReadInt32());

}

}

foreach (int i in inputMas)

{

Console.Write(i + " ");

}

Console.ReadKey();

}

}

}

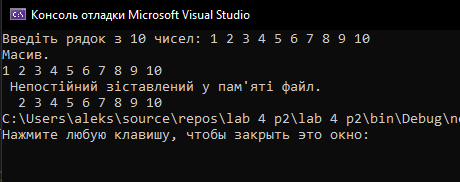
}  


Рис. 2 – Результат прикладу 4.2

3. Переробити приклад 4.3 так, щоб записувати у файл фразу “Hello,

World!” (кожен процес записує по слову).

**l\_4\_3.cs:**

using System;

using System.Diagnostics;

using System.Diagnostics.Metrics;

using System.IO;

using System.IO.MemoryMappedFiles;

using System.Threading;

namespace l\_4\_3\_a

{

class Program

{

static void Main(string[] args)

{

using (MemoryMappedFile mmf = MemoryMappedFile.CreateNew("procfile", 10000))

{

bool mutexCreated;

Console.WriteLine("\t\tПроцес А.");

Console.WriteLine("Створюється м'ютекс.");

Mutex mutex = new Mutex(true, "testmapmutex", out mutexCreated);

Console.WriteLine("Непостiйний зiставлений у пам’ятi файл.");

using (MemoryMappedViewStream stream = mmf.CreateViewStream())

{

BinaryWriter writer = new BinaryWriter(stream);

Console.WriteLine("Запис даних.");

writer.Write("Hello,");

}

mutex.ReleaseMutex();

Process procB = new Process();

procB.StartInfo.FileName = @"C:\Users\aleks\source\repos\step1\step1\bin\Debug\net8.0\step1.exe";

procB.Start();

Console.WriteLine("Запуск процеса В. Для продовження натиснути ENTER.");

Console.ReadLine();

mutex.WaitOne();

using (MemoryMappedViewStream stream = mmf.CreateViewStream())

{

stream.Position = 0;

StreamReader reader = new StreamReader(stream);

Console.WriteLine("Процес A передав: {0}", reader.ReadLine());

Console.WriteLine("Процес B передав: {0}", reader.ReadLine());

}

mutex.ReleaseMutex();

Console.ReadKey();

}

}

}

}

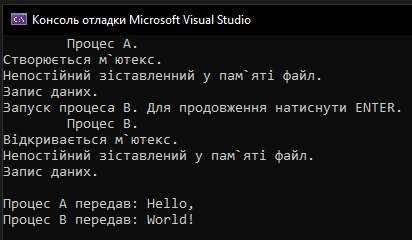


Рис. 3 – Результат прикладу 4.3

4. Виконати свій варіант по списку підгрупи в проекті C# Windows

Forms.

Варіант 5

Написати програму, в якій задається RGB-код кольору. У дочірньому

процесі колір фону вікна змінюється на заданий колір.

**Form1.cs:**  
using System.IO;

using System.Threading.Tasks;

namespace lab4p4

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

Task.Run(() => WriteToFile());

}

private async void WriteToFile()

{

string path = @"C:\Users\aleks\Desktop\color.txt"; // Замените на путь к вашему файлу

using (StreamWriter sw = new StreamWriter(path))

{

await sw.WriteAsync(this.textBox1.Text);

}

Task.Run(() => ReadFile());

}

private async void ReadFile()

{

string path = @"C:\Users\aleks\Desktop\color.txt"; // Замените на путь к вашему файлу

using (StreamReader sr = new StreamReader(path))

{

string line = await sr.ReadToEndAsync();

Invoke((MethodInvoker)delegate

{

string[] rgb = line.Split(' ');

if (rgb.Length == 3)

{

try

{

int r = int.Parse(rgb[0]);

int g = int.Parse(rgb[1]);

int b = int.Parse(rgb[2]);

Task.Run(() => ChangeBackground(r, g, b));

}

catch (Exception)

{

MessageBox.Show("Помилка при перетворенні RGB-коду на колір");

}

}

else

{

MessageBox.Show("RGB-код повинен містити три числа, розділені пробілами");

}

});

}

}

private void ChangeBackground(int r, int g, int b)

{

Invoke((MethodInvoker)delegate

{

this.BackColor = Color.FromArgb(r, g, b);

});

}

}

}

**Form1.Designer.cs:**

namespace lab4p4

{

partial class Form1

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

button1 = new Button();

textBox1 = new TextBox();

SuspendLayout();

//

// button1

//

button1.Location = new Point(333, 263);

button1.Name = "button1";

button1.Size = new Size(75, 23);

button1.TabIndex = 0;

button1.Text = "button1";

button1.UseVisualStyleBackColor = true;

button1.Click += button1\_Click;

//

// textBox1

//

textBox1.Location = new Point(324, 204);

textBox1.Name = "textBox1";

textBox1.Size = new Size(100, 23);

textBox1.TabIndex = 1;

//

// Form1

//

AutoScaleDimensions = new SizeF(7F, 15F);

AutoScaleMode = AutoScaleMode.Font;

ClientSize = new Size(800, 450);

Controls.Add(textBox1);

Controls.Add(button1);

Name = "Form1";

Text = "Form1";

ResumeLayout(false);

PerformLayout();

}

#endregion

private Button button1;

private TextBox textBox1;

}

}

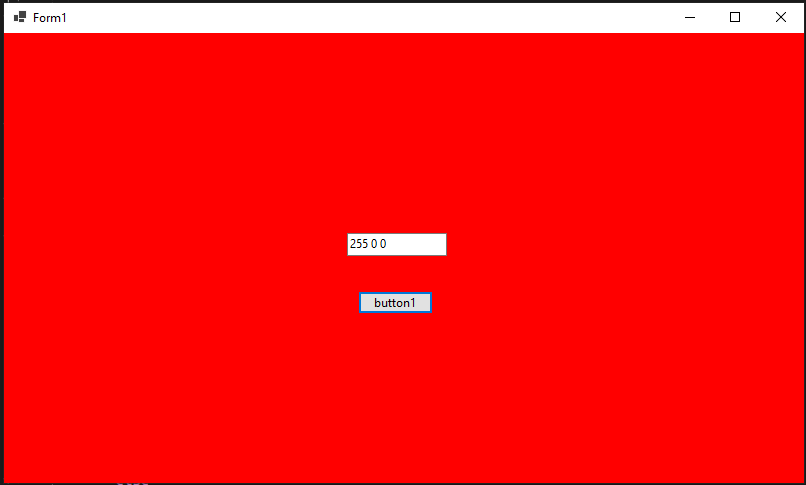


Рис. 4 – Перший результат

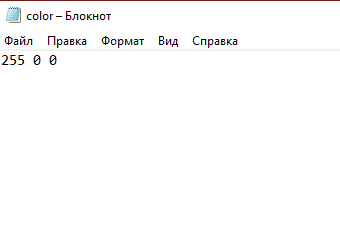


Рис. 4.1 – Результат у файлі



Рис. 5 – Другий результат

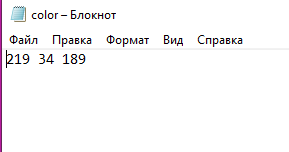
****

Рис. 5.1 – Результат у файлі

**Контрольні питання**

1. **Коротко описати концепцію механізму відображення файлів у пам’яті.**

**Відображення файлів у пам’яті** – це механізм, який дозволяє програмам працювати з файлами так, ніби вони є частиною оперативної пам’яті. Це забезпечує швидкий доступ до даних і може покращити продуктивність програм, особливо коли працюєте з великими файлами.

1. **Назвати два типи зіставлених у пам’яті файлів і їх відмінність.**

Існують два типи зіставлених у пам’яті файлів: постійні та непостійні.

* Постійні зіставлені файли пов’язані з реальним файлом на диску і зберігають свої дані між сеансами.
* Непостійні зіставлені файли, відомі також як анонімні, не пов’язані з файлом на диску і використовуються для обміну даними між процесами.

1. **Який метод використовується для отримання об’єкта, який представляє постійний зіставлений у пам’яті файл?**

Для отримання об’єкта, який представляє постійний зіставлений у пам’яті файл, використовується метод MemoryMappedFile.CreateFromFile().

1. **Який метод використовується для отримання об’єкта, який представляє непостійний зіставлений у пам’яті файл?**

Для отримання об’єкта, який представляє непостійний зіставлений у пам’яті файл, використовується метод MemoryMappedFile.CreateNew().

1. **Який метод використовується для отримання об’єкта представлення для послідовного і довільного доступу до зіставленого в пам’яті файлу?**

Для отримання об’єкта представлення для послідовного і довільного доступу до зіставленого в пам’яті файлу використовується метод MemoryMappedFile.CreateViewStream(). Цей метод повертає об’єкт MemoryMappedViewStream, який можна використовувати для читання та запису даних у файл.

**Висновок:** Під час виконання лабораторної роботи було ознайомлено з методами для роботи з постійно та непостійно зіставленими у памяті файлами, розглянуто різницю у їх застосуванні.