**ЛАБОРАТОРНА РОБОТА № 5**

**Тема:** Динамічно зв'язувані бібліотеки.

**Мета:** Вивчення роботи з DLL.

**Завдання:**

1. Написати програмний продукт, який буде мати об'єктно-орієнтовану архітектуру. Обов'язкове створення та реалізація двох класів. Перший - інтерфейс, другий - обчислення.
2. Кожен клас має розташовуватися в окремій DLL.
3. DLL яка реалізує інтерфейс має завантажуватися разом з основною програмою.
4. DLL яка реалізує обчислення має завантажуватися пізніше, вже під час роботи програмного продукту, за вимогою користувача. А також після використання відвантажуватися з пам'яті. Перед завантаженням необхідно перевірити чи є вже бібліотека у пам'яті.
5. Підключити та використати будь які функції для прикладу, будь яку не власну бібліотеку.
6. При реалізації класу обчислень, використати алгоритм повного перебору, або сортування qsort, та використати породження багатьох потоків. Вивчити поведінку системи при граничному використанні пам'яті та породжених потоків. Показати графіки використання ресурсів. Проаналізувати залежності в них.

Результат виконання завдання:

Program.cs

namespace ConsoleApp1

{

class Program

{

static void Main(string[] args)

{

View.Interface();

}

}

}

View.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Reflection;

using System.Text;

using System.Threading.Tasks;

namespace DLLView

{

public class View

{

public static void Interface()

{

object operation;

do

{

operation = scanf();

if (toInt(operation) == 1)

{

Assembly asm = null;

try

{

asm = Assembly.LoadFrom(@"C:\Users\Yaroslav\Desktop\DLLWorkingSysProgramming\DLLcount\bin\Debug\DLLcount.dll");

if (asm == null) return;

}

catch (FileNotFoundException ex)

{

return;

}

Type tp;

try

{

tp = asm.GetType("DLLcount.Calculate");

object ob = Activator.CreateInstance(tp);

MethodInfo mi;

mi = tp.GetMethod("Array");

int[] a = new int[500];

Random rnd = new Random();

for (int i = 0; i < a.GetLength(0); i++)

{

a[i] = rnd.Next(-100, 200);

}

for (int i = 0; i < a.GetLength(0); i++)

{

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

}

Console.WriteLine();

object[] parametrs = new object[]{a, 0, 499};

int[] array;

array = (int[])mi.Invoke(ob, parametrs);

for (int i = 0; i < array.GetLength(0); i++)

{

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

}

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

} while (toInt(operation) != 0);

}

public static void printf(string message)

{

Console.WriteLine(message);

}

public static void printf(string message, object smth)

{

Console.WriteLine(message + smth.ToString());

}

public static object scanf()

{

return Console.ReadLine();

}

public static double toDouble(object value)

{

return Convert.ToDouble(value);

}

public static int toInt(object value)

{

try

{

return Convert.ToInt32(value);

}

catch

{

return 0;

}

}

}

}

Calculate.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

using System.Reflection;

using DLLView;

using System.Threading;

using System.Diagnostics;

namespace DLLcount

{

public class Calculate

{

public int[] Array(int[] array, int left, int right)

{

Structure a = new Structure();

a.arr = array;

a.left = left;

a.right = right;

Thread myThread = new Thread(new ParameterizedThreadStart(qSort));

myThread.Start(a);

Thread.Sleep(15000);

return array;

}

public static void Swap(ref int a, ref int b)

{

int tmp = a;

a = b;

b = tmp;

}

private static int Piece(int[] array, int left, int right)

{

int origin = array[right];

int i = (left - 1);

for (int j = left; j <= right - 1; j++)

if (array[j] <= origin)

{

i++;

Swap(ref array[i], ref array[j]);

}

Swap(ref array[i + 1], ref array[right]);

return i + 1;

}

public struct Structure

{

public int left;

public int right;

public int[] arr;

public int index;

}

public static void qSort(object ar)

{

Structure b = (Structure)ar;

if (b.left < b.right)

{

b.index = Piece(b.arr, b.left, b.right);

Structure buff1 = b;

Structure buff2 = b;

buff1.right = b.index - 1;

Thread t1 = new Thread(new ParameterizedThreadStart(qSort));

t1.Start(buff1);

buff2.left = b.index + 1;

Thread t2 = new Thread(new ParameterizedThreadStart(qSort));

t2.Start(buff2);

Console.WriteLine($"\n{Process.GetCurrentProcess().Threads.Count} ");

Thread.Sleep(10);

}

}

}

}

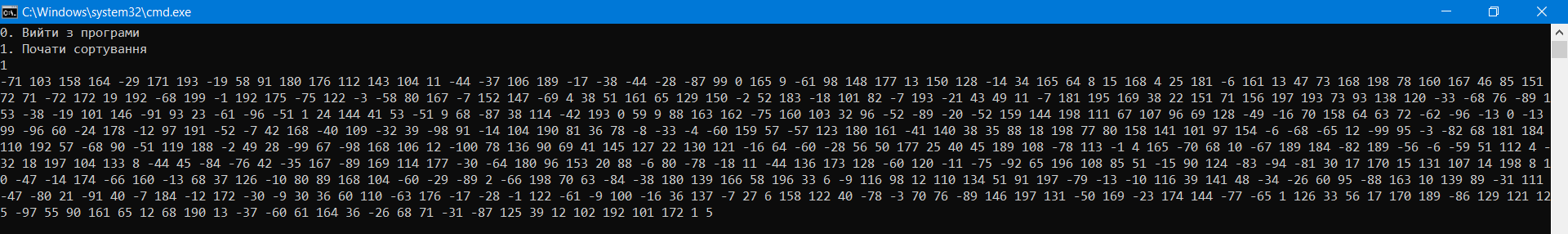


Рис. 5.1 Генерація чисел

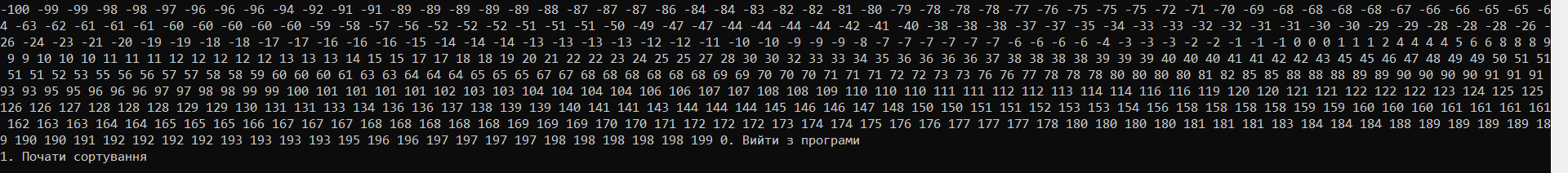


Рис. 5.2 Сортування чисел

**Висновок:** навчився працювати з DLL.